Package 'mfrmr'

June 12, 2026

Type Package

Title Estimation and Diagnostics for Many-Facet Measurement Models

Version 0.2.1

Description Native R implementation of many-facet ordered-response measurement models with arbitrary facet counts, rating-scale and partial-credit parameterizations, a bounded generalized partial-credit extension, and both marginal and joint maximum likelihood estimation. The package provides a fit / diagnose / report pipeline covering anchoring, linking, bias and differential-functioning screening, and publication-oriented reporting summaries, with reproducibility manifests for replay. See 'Andrich' (1978) <doi:10.1007/BF02293814>, 'Masters' (1982) <doi:10.1007/BF02296272>, and 'Muraki' (1992) <doi:10.1177/014662169201600206> for the underlying ordered-response models.

URL https://github.com/Ryuya-dot-com/mfrmr

BugReports https://github.com/Ryuya-dot-com/mfrmr/issues

License MIT + file LICENSE

Language en

Encoding UTF-8

LazyData true

Depends R (>= 4.1)

Imports cli, dplyr, grDevices, graphics, tidyr, tibble, purrr, stringr, psych, lifecycle, rlang, methods, stats, utils

LinkingTo cpp11

Suggests testthat (>= 3.0.0), covr, knitr, rmarkdown, igraph, lme4, digest, kableExtra, flextable, future.apply, shiny, mirt, TAM, eRm

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Ryuya Komuro [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-9205-0926>)

Maintainer Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp>

Repository CRAN

Date/Publication 2026-06-12 17:00:13 UTC

Contents

`mfrmr-package` *mfrmr: Many-Facet Ordered-Response Modeling in R*

Description

`mfrmr` provides estimation, diagnostics, and reporting utilities for many-facet ordered-response measurement models: the Rasch-family `RSM` / `PCM` route and the package's bounded `GPCM` extension where explicitly documented.

Details

Start with the following core workflow before branching into the longer GPCM, simulation, and planning notes:

1. Fit with `fit_mfrm()` using `method = "MML"`
2. For RSM / PCM, run `diagnose_mfrm()` with `diagnostic_mode = "both"`; for bounded GPCM, use the direct diagnostic route and read `gpcm_capability_matrix()`
3. Build a comprehensive first screen with `mfrm_results()`
4. Read `summary(fit)`, `summary(diag)`, and `summary(res)` before branching
5. Use `plot_qc_dashboard()` and `reporting_checklist()` as the first visual and reporting screens

Recommended workflow:

1. Fit model with `fit_mfrm()`
2. For RSM / PCM, compute diagnostics with `diagnose_mfrm()` and prefer `diagnostic_mode = "both"` when you want legacy residual continuity plus the newer strict marginal-fit screen
3. For RSM / PCM, run residual PCA with `analyze_residual_pca()` if needed
4. For RSM / PCM, or bounded GPCM with the documented screening caveat, estimate interactions with `estimate_bias()`
5. For RSM / PCM, choose a downstream branch: `reporting_checklist()` for manuscript/report preparation, or `build_misfit_casebook()` / `build_linking_review()` for operational misfit or anchor/drift review. After `build_misfit_casebook()`, inspect `casebook$group_view_index` before moving to source-specific plots.
6. For RSM / PCM, build narrative/report outputs with `build_apa_outputs()` and `build_visual_summaries()`
7. Treat bounded GPCM, prediction, and planning helpers as advanced scope after the basic RSM / PCM route is working cleanly.

Guide pages:

- `mfrmr_output_guide()` for the compact purpose-to-helper map
- mfrmr_workflow_methods
- mfrmr_visual_diagnostics
- mfrmr_reports_and_tables
- mfrmr_reporting_and_apa
- mfrmr_linking_and_dff
- gpcm_capability_matrix
- mfrmr_compatibility_layer

Companion vignettes:

- `vignette("mfrmr-workflow", package = "mfrmr")`
- `vignette("mfrmr-mml-and-marginal-fit", package = "mfrmr")`
- `vignette("mfrmr-visual-diagnostics", package = "mfrmr")`

- `vignette("mfrmr-reporting-and-apa", package = "mfrmr")`
- `vignette("mfrmr-linking-and-dff", package = "mfrmr")`

A two-page landscape cheatsheet of the public API ships at `system.file("cheatsheet", "mfrmr-cheatsheet.pdf", package = "mfrmr")` (pre-rendered) and `system.file("cheatsheet", "mfrmr-cheatsheet.Rmd", package = "mfrmr")` (source). Open the PDF directly for a printable reference card, or knit the source with `rmarkdown::render()` when you want a customised version.

First 5-minute route

Use this order before exploring the broader feature surface:

1. `fit_mfrm()` with `method = "MML"`
2. `diagnose_mfrm()` with `diagnostic_mode = "both"` for `RSM` / `PCM`; for bounded `GPCM`, keep diagnostics on the direct exploratory route
3. `mfrm_results()` for a FACETS-style first screen
4. `summary(fit)`, `summary(diag)`, and `summary(res)`
5. `plot_qc_dashboard()` for first-pass triage
6. Choose the next branch: `reporting_checklist()` for reporting, `build_weighting_review()` for Rasch-versus-`GPCM` weighting review, `build_misfit_casebook()` for operational case review, or `build_linking_review()` for operational linking review (`RSM` / `PCM`) or caveated bounded-`GPCM` linking synthesis

Advanced scope

After the basic route above:

- the package now includes a first-version latent-regression `MML` branch for ordered-response `RSM` / `PCM` models with a one-dimensional conditional-normal population model and explicit one-row-per-person covariates expanded through `stats::model.matrix()`
- bounded `GPCM` support is summarized by `gpcm_capability_matrix()`
- bounded `GPCM` supports the core fit/summary/scoring/information path, direct Wright/pathway/CCC plots, residual-PCA follow-up, and the residual-based diagnostics tables/plots as exploratory tools
- posterior-predictive computation, `MCMC` engines, and Docker-based advanced runtimes are future extensions rather than requirements for the current bounded `GPCM` route
- direct `GPCM` data generation through `build_mfrm_sim_spec()`, `extract_mfrm_sim_spec()`, and `simulate_mfrm_data()` is available when the specification carries both thresholds and slopes
- slope-aware `fair_average_table()` and `estimate_bias()` are available for bounded `GPCM` with explicit caveats; `build_apa_outputs()`, `build_visual_summaries()`, `run_qc_pipeline()`, `build_mfrm_manifest()`, `build_mfrm_replay_script()`, and `export_mfrm_bundle()` are available as caveated partial reporting/export surfaces; score-side FACETS compatibility and broader planning semantics remain validated for `RSM` / `PCM`
- `predict_mfrm_population()` remains a scenario-level forecast helper and should not be described as the latent-regression estimator itself

- the current simulation/planning layer remains role-based for two non-person facets rather than fully arbitrary-facet planning, with boundaries exposed through planner metadata such as `planning_scope`, `planning_constraints`, and `planning_schema`
- latent-class mixture models and response-time / careless-rating adjustment are not estimated by mfrmr; use residual, person-fit, local-dependence, and rater-drift diagnostics as screening layers rather than as mixture-model substitutes

Equal weighting versus bounded GPCM

The package's operational reference route is the Rasch-family `RSM` / `PCM` branch. That route enforces fixed discrimination and therefore preserves an equal-weighting scoring interpretation across observed ratings.

Bounded `GPCM` is supported because some users want a slope-aware model- comparison or sensitivity layer inside the same many-facet workflow. However, the package does not treat bounded `GPCM` as a universal replacement for the Rasch-family route. A better fit under `GPCM` should be read as evidence about discrimination-based reweighting, not as an automatic reason to discard the equal-weighting model.

Observation weights are a different concept again. Optional `Weight` columns change how observed rating events enter estimation and summaries, but they do not create a free-form facet-weighting scheme and do not alter the fixed-discrimination meaning of `RSM` / `PCM`.

Public entry map:

- First-screen results: `mfrm_results()`, `summary(res)$next_actions`, and `mfrmr_output_guide("entry")`
- Interactive exploration: `mfrm_results_interactive()` only when prompts are explicitly wanted at the console

Function families:

- Model fitting: `fit_mfrm()`, `summary.mfrm_fit()`, `plot.mfrm_fit()`
- Legacy-compatible workflow wrapper: `run_mfrm_facets()`, `mfrmRFacets()`
- Diagnostics: `diagnose_mfrm()`, `summary(diag)`, `analyze_residual_pca()`, `plot_residual_pca()`
- Bias and interaction: `estimate_bias()`, `estimate_all_bias()`, `summary(bias)`, `bias_interaction_report()`, `plot_bias_interaction()`
- Differential functioning: `analyze_dff()`, `analyze_dif()`, `dif_interaction_table()`, `plot_dif_heatmap()`, `dif_report()`
- Design simulation: `build_mfrm_sim_spec()`, `extract_mfrm_sim_spec()`, `simulate_mfrm_data()`, `evaluate_mfrm_recovery()`, `assess_mfrm_recovery()`, `evaluate_mfrm_design()`, `evaluate_mfrm_signal_det` `predict_mfrm_population()`, `predict_mfrm_units()`, `sample_mfrm_plausible_values()` (including fit-derived empirical / resampled / skeleton-based simulation specifications; fixed-calibration unit scoring supports `MML` fits directly, latent-regression `MML` fits through the fitted population model when scored units also provide one-row-per-person background data, and `JML` fits through a post hoc reference-prior EAP layer; fit-derived simulation specifications also support direct bounded `GPCM` data generation, recovery checks, role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers with documented caveats; curve reports, graph-only exports, fair-average tables, and bias screening are also available for bounded `GPCM` with documented caveats)

- Reporting: build_apa_outputs(), build_visual_summaries(), reporting_checklist(), apa_table() for the full RSM / PCM route; bounded GPCM currently stays on the checklist / direct-table / direct- plot / summary-appendix side instead of the narrative/QC layer
- Weighting review: compare_mfrm(), build_weighting_review(), build_model_choice_review(), compute_information(), plot_information()
- Case review: build_misfit_casebook(), plot_unexpected(), plot_displacement(), plot_marginal_fit(), plot_marginal_pairwise()
- Linking and scale maintenance: review_mfrm_anchors(), detect_anchor_drift(), build_equating_chain(), build_linking_review(), plot_anchor_drift()
- Dashboards: facet_quality_dashboard(), plot_facet_quality_dashboard()
- Export / reproducibility: build_mfrm_manifest(), build_mfrm_replay_script(), build_conquest_overlap_bun normalize_conquest_overlap_files(), normalize_conquest_overlap_tables(), review_conquest_overlap(export_mfrm_bundle() for the diagnostics-compatible Rasch-family route; bounded GPCM remains outside the current fit-based manifest/replay/bundle layer but can use export_summary_appendix() for documented direct outputs
- Equivalence: analyze_facet_equivalence(), plot_facet_equivalence()
- Data and anchors: describe_mfrm_data(), review_mfrm_anchors(), make_anchor_table(), load_mfrmr_data()

Data interface:

- Input analysis data is long format (one row per observed rating).
- Required columns are one person column, one ordered score column, and one or more non-person facet columns named in facets = c(...).
- Score values should be ordered integer categories. Binary 0/1 or 1/2 input is supported as the two-category Rasch-family special case; by contrast, fractional score values should be recoded before fitting rather than relying on automatic coercion.
- If keep_original = FALSE, unused intermediate categories are collapsed to a contiguous internal scale and the mapping is stored in fit$prep$score_map.
- If the intended scale has unused boundary categories, such as a 1-5 scale with only 2-5 observed, set rating_min = 1, rating_max = 5 so the zero-count boundary category remains in the fitted support. If unused intermediate categories should also remain in the original scale, set keep_original = TRUE.
- summary(describe_mfrm_data(...)) reports retained zero-count categories in Notes, printed Caveats, and $caveats; summary(fit) carries full structured rows into printed Caveats and $caveats, with Key warnings as a short triage subset. Summary-table exports route those rows through score_category_caveats or analysis_caveats. Treat adjacent thresholds as weakly identified when an intermediate category is unobserved.
- Optional columns such as Subset, Weight, and Group support linking, weighted analysis, and fairness-focused follow-up workflows.
- Packaged simulation data is available via load_mfrmr_data() or data().

Interpreting output

Core object classes are:

- `mfrm_fit`: fitted model parameters and metadata.
- `mfrm_diagnostics`: fit, facet-level reliability, and flag diagnostics, plus inter-rater agreement when one facet is treated as a rater facet.
- `mfrm_bias`: interaction bias estimates.
- `mfrm_dff` / `mfrm_dif`: differential-functioning contrasts and screening summaries.
- `mfrm_population_prediction`: scenario-level forecast summaries for one future design.
- `mfrm_unit_prediction`: posterior summaries for future or partially observed persons under the fitted scoring basis.
- `mfrm_plausible_values`: posterior draws for future or partially observed persons under the fitted scoring basis.
- `mfrm_bundle` families: summary/report bundles and draw-free plot data.

Typical workflow

1. Prepare long-format data.
2. Fit with `fit_mfrm()`.
3. For `RSM` / `PCM`, diagnose with `diagnose_mfrm()` and prefer `diagnostic_mode = "both"` for final `MML` runs.
4. For `RSM` / `PCM`, run `analyze_dff()` or `estimate_bias()` when fairness or interaction questions matter; bounded `GPCM` also supports `estimate_bias()` as a conditional screening review.
5. For `RSM` / `PCM`, report with `build_apa_outputs()` and `build_visual_summaries()`.
6. For design planning, move to `build_mfrm_sim_spec()`, `evaluate_mfrm_design()`, `mfrm_generalizability()`, `mfrm_d_study()`, and `predict_mfrm_population()`. Bounded `GPCM` also supports direct simulation via `extract_mfrm_sim_spec()` / `simulate_mfrm_data()`, but not the broader planning helpers. Those helpers assume two non-person facet roles even though the estimation core supports arbitrary facet counts. Treat `evaluate_mfrm_design()` as Monte Carlo design evaluation, and use `mfrm_d_study()` for analytic G/Phi design projections. `predict_mfrm_population()` remains the scenario-level forecast helper, not the latent-regression estimator.
7. For future-unit scoring, retain an `MML` calibration when you want the fitted marginal model directly, use an active latent-regression `MML` fit when scored units also provide one-row-per-person background data, or use a `JML` calibration when a post hoc fixed-calibration EAP layer is acceptable; then score with `predict_mfrm_units()` or `sample_mfrm_plausible_values()`.
8. For bounded `GPCM`, use `summary.mfrm_fit()`, `diagnose_mfrm()`, `analyze_residual_pca()`, `predict_mfrm_units()`, `sample_mfrm_plausible_values()`, `compute_information()`, `plot_qc_dashboard()`, `plot.mfrm_fit()`, `category_structure_report()`, `category_curves_report()`, graph-only `facets_output_file_bundle()`, direct simulation-spec generation/data generation, recovery checks, `fair_average_table()`, `estimate_bias()`, and the residual-based table helpers with their documented caveats. Caveated APA/QC/export bundles and exploratory linking review plus role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers are available, while full score-side FACETS review, posterior-predictive checks, and heavy backends remain outside the bounded `GPCM` route. Use `gpcm_capability_matrix()` as the formal boundary statement.

Model formulation

The Rasch-family branch used by `RSM` and `PCM` extends the basic Rasch model by incorporating multiple measurement facets into a single additive linear predictor on the log-odds scale.

RSM/PCM adjacent-category equation

For an observation where person n with ability θ_n is rated by rater j with severity δ_j on criterion i with difficulty β_i, the probability of observing category k (out of K ordered categories) is:

$$P(X_{nij} = k \mid \theta_n, \delta_j, \beta_i, \tau) = \frac{\exp\left[\sum_{s=1}^{k}(\theta_n - \delta_j - \beta_i - \tau_s)\right]}{\sum_{c=0}^{K}\exp\left[\sum_{s=1}^{c}(\theta_n - \delta_j - \beta_i - \tau_s)\right]}$$

where τ_s are the Rasch-Andrich threshold (step) parameters in the `RSM` reference case and $\sum_{s=1}^{0}(\cdot) \equiv 0$ by convention. Additional facets enter as additive terms in the linear predictor $\eta = \theta_n - \delta_j - \beta_i - \ldots$.

This additive predictor generalises to any number of facets; the `facets` argument to `fit_mfrm()` accepts an arbitrary-length character vector.

Rating Scale Model (RSM)

Under the RSM (Andrich, 1978), all levels of the step facet share a single set of threshold parameters $\tau_1, \ldots, \tau_K$.

Partial Credit Model (PCM)

Under the PCM (Masters, 1982), each level of the designated `step_facet` has its own threshold vector on the package's common observed score scale. In the current implementation, threshold locations may vary by step-facet level, but the fitted score range is defined by one global category set taken from the observed data.

Bounded Generalized Partial Credit Model (GPCM)

Under bounded `GPCM` (Muraki, 1992), the same adjacent-category partial-credit kernel is multiplied by a positive slope α_g for the designated slope-facet level g:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k-1)} = \alpha_g(\theta_n - \delta_j - \beta_i - \tau_{gk}).$$

The current implementation requires `slope_facet == step_facet` and identifies slopes on the log scale with geometric mean 1. This makes bounded `GPCM` a slope-aware sensitivity/extension route, not a replacement for the equal-weighting `RSM`/`PCM` interpretation.

Ordered-response scope

The implemented response-model scope is ordered categorical only. Binary responses are the $K = 1$ special case of the same formulation, so they are handled through the ordinary ordered-score interface. This means `mfrmr` supports ordered binary and ordered polytomous data under `RSM` and `PCM`, plus a narrow bounded `GPCM` branch with one designated `slope_facet` that currently must equal `step_facet`. Unordered nominal/multinomial response models are not yet implemented.

Estimation methods

Marginal Maximum Likelihood (MML)

MML integrates over the person ability distribution using Gauss-Hermite quadrature, in the broader marginal-likelihood framework introduced by Bock & Aitkin (1981) for IRT:

$$L = \prod_n \int P(\mathbf{X}_n \mid \theta, \boldsymbol{\delta})\, \phi(\theta)\, d\theta \approx \prod_n \sum_{q=1}^{Q} w_q\, P(\mathbf{X}_n \mid \theta_q, \boldsymbol{\delta})$$

where $\phi(\theta)$ is the assumed normal prior and (θ_q, w_q) are quadrature nodes and weights. Person estimates are obtained post-hoc via Expected A Posteriori (EAP):

$$\hat{\theta}_n^{\mathrm{EAP}} = \frac{\sum_q \theta_q\, w_q\, L(\mathbf{X}_n \mid \theta_q)}{\sum_q w_q\, L(\mathbf{X}_n \mid \theta_q)}$$

MML avoids the incidental-parameter problem and is generally preferred for smaller samples.

Note: Bock & Aitkin (1981) is the canonical citation for the Gauss-Hermite-quadrature MML *framework*. The default mfrmr engine (`mml_engine = "direct"`) optimises this marginal log-likelihood by direct gradient methods (BFGS / L-BFGS-B), not by Bock & Aitkin's signature EM algorithm. The `"em"` and `"hybrid"` engines do follow the EM template but use a BFGS M-step rather than B&A's probit IRLS, because the target is the polytomous Rasch family rather than B&A's 2PL probit model.

Joint Maximum Likelihood (JML)

JML estimates all person and facet parameters simultaneously as fixed effects by maximising the joint log-likelihood $\ell(\boldsymbol{\theta}, \boldsymbol{\delta} \mid \mathbf{X})$ directly. It does not assume a parametric person distribution, which can be advantageous when the population shape is strongly non-normal, but parameter estimates are known to be biased when the number of persons is small relative to the number of items (Neyman & Scott, 1948). The package still accepts `"JMLE"` as a backward-compatible alias, but user-facing summaries and documentation use `"JML"` as the public label.

See `fit_mfrm()` for practical guidance on choosing between the two.

Strict marginal diagnostics

For `RSM` / `PCM`, `diagnose_mfrm(..., diagnostic_mode = "both")` returns two complementary targets: the `legacy` residual / EAP diagnostics and a `marginal_fit` layer whose expected counts and pairwise summaries are integrated over the posterior quadrature bundle rather than plugged in at the EAP point. The screen is structured as limited-information evidence (Orlando & Thissen, 2000; Haberman & Sinharay, 2013; Sinharay & Monroe, 2025), not as an omnibus accept / reject test, and it complements rather than replaces separation / reliability and inter-rater agreement summaries. The full derivation, with notation and pairwise local-dependence events, lives in `vignette("mfrmr-mml-and-marginal-fit", package = "mfrmr")`.

Statistical background

Key statistics reported throughout the package:

Infit (Information-Weighted Mean Square)

Weighted average of squared standardized residuals, where weights are the model-based variance of each observation:

$$\text{Infit}_j = \frac{\sum_i Z_{ij}^2 \, \text{Var}_i \, w_i}{\sum_i \text{Var}_i \, w_i}$$

Expected value is 1.0 under model fit. Values below 0.5 suggest overfit (Mead-style responses); values above 1.5 suggest underfit (noise or misfit). Infit is most sensitive to unexpected patterns among on-target observations (Wright & Masters, 1982).

Note: The 0.5–1.5 range is the general "productive for measurement" band given by Linacre (2002, *RMT* 16(2), 878). Context-specific bands come from Wright & Linacre (1994, *RMT* 8(3), 370): 0.8–1.2 for high-stakes MCQ, 0.7–1.3 for run-of-the-mill MCQ, 0.6–1.4 for rating-scale surveys, 0.5–1.7 for clinical observation, and 0.4–1.2 for judged performance. See also Bond & Fox (2015) for textbook summaries of these conventions.

Outfit (Unweighted Mean Square)

Simple average of squared standardized residuals:

$$\text{Outfit}_j = \frac{\sum_i Z_{ij}^2 \, w_i}{\sum_i w_i}$$

Same expected value and flagging thresholds as Infit, but more sensitive to extreme off-target outliers (e.g., a high-ability person scoring the lowest category).

ZSTD (Standardized Fit Statistic)

Wilson-Hilferty (1931) cube-root transformation that converts the mean-square chi-square ratio to an approximate standard normal deviate:

$$\text{ZSTD} = \frac{\text{MnSq}^{1/3} - (1 - 2/(9\,df))}{\sqrt{2/(9\,df)}}$$

Values near 0 indicate expected fit; $|\text{ZSTD}| > 2$ flags potential misfit at the 5\ 1\ ZSTD is reported alongside every Infit and Outfit value. ZSTD is withheld (`NA`) when the applicable df falls below 1, where the Wilson-Hilferty transformation is numerically unstable; FACETS/Winsteps under `WHEXACT` can continue with a linear approximation on such cells.

Residual basis under MML vs JMLE engines

For `method = "MML"` fits, the standardized residuals behind Infit, Outfit, and ZSTD are evaluated at EAP person measures, which are shrunken toward the population mean. JMLE engines such as FACETS evaluate the same formulas at unshrunken JMLE estimates, so MnSq and ZSTD values are not numerically interchangeable across the two residual bases, most visibly for extreme-scoring persons. Use `method = "JML"` when an external FACETS fit comparison requires a JMLE-style residual basis, and see `facets_fit_df_guide()` for the separate standardization-side df/ZSTD conventions.

PTMEA (Point-Measure Correlation)

Pearson correlation between observed scores and estimated person measures within each facet level. Positive values indicate that scoring aligns with the latent trait dimension; negative values suggest reversed orientation or scoring errors.

Separation

Package-reported separation is the ratio of adjusted true standard deviation to root-mean-square measurement error:

$$G = \frac{\mathrm{SD}_{\mathrm{adj}}}{\mathrm{RMSE}}$$

where $\mathrm{SD}_{\mathrm{adj}} = \sqrt{\mathrm{ObservedVariance} - \mathrm{ErrorVariance}}$. Higher values indicate the facet discriminates more statistically distinct levels along the measured variable. In `mfrmr`, `Separation` is the model-based value and `RealSeparation` provides a more conservative companion based on `RealSE`.

Reliability

$$R = \frac{G^2}{1 + G^2}$$

Analogous to Cronbach's alpha or KR-20 for the reproducibility of element ordering. In `mfrmr`, `Reliability` is the model-based value and `RealReliability` gives the conservative companion based on `RealSE`. For `MML`, these are anchored to observed-information `ModelSE` estimates for non-person facets; `JML` keeps them as exploratory summaries.

For the person facet under `MML`, the same G and R formulas are applied to EAP person measures with posterior SDs in the error slot. EAP measures are shrunken, so their observed variance is already deflated (approximately the true variance times the reliability), and subtracting the mean posterior variance deflates it again. The reported MML person separation/reliability is therefore a conservative summary: it is systematically lower than the IRT empirical-reliability convention $\mathrm{Var}(\mathrm{EAP})/(\mathrm{Var}(\mathrm{EAP}) + \overline{\mathrm{PSD}^2})$ and is not numerically comparable to JMLE-based person separation reliability from FACETS. The gap is small when measurement is precise and grows as precision drops. Person rows can still carry the model-based precision tier because posterior SDs are model-based quantities; that tier describes the SE source, not FACETS comparability. Use `method = "JML"` when a FACETS-style person separation table is required, and treat MML person rows as conservative summaries.

This is a Rasch/FACETS-style separation reliability on the fitted logit scale, not an intra-class correlation. Use `compute_facet_icc()` only when you want the complementary random-effects variance-share view on the observed-score scale; for non-person facets, large ICC values indicate systematic facet variance rather than desirable measurement reliability.

Strata

Number of statistically distinguishable groups of elements:

$$H = \frac{4G + 1}{3}$$

Three or more strata are commonly used as a practical target (Wright & Masters, 1982), but in this package the estimate inherits the same approximation limits as the separation index.

Key references

- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43, 561–573.
- Bond, T. G., & Fox, C. M. (2015). *Applying the Rasch model* (3rd ed.). Routledge.
- Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, 46, 443–459.
- Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed.). Springer. (AIC / BIC weights and Delta-IC bands used by `compare_mfrm()`.)
- Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology*, 38(1), 67–86. (Source for the `lz` person-fit statistic implemented in `compute_person_fit_indices()`.)
- Haberman, S. J., & Sinharay, S. (2013). Generalized residuals for general models for contingency tables with application to item response theory. *Journal of the American Statistical Association*, 108, 1435–1444.
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly*, 2, 197–221.
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, 16(2), 159–176. (Source for the bounded `GPCM` extension used in `fit_mfrm(model = "GPCM")`, `fair_average_table()`, and `estimate_bias()`.)
- Muraki, E. (1993). Information functions of the generalized partial credit model. *Applied Psychological Measurement*, 17(4), 351–363. (Companion paper to Muraki 1992 that derives the GPCM item information identity $I_j(\theta) = D^2 a_j^2 \mathrm{Var}(T \mid \theta)$ via Samejima's (1974) polytomous information formula. This is the canonical reference for `compute_information()` under bounded `GPCM`.)
- Samejima, F. (1974). Normal ogive model on the continuous response level in the multidimensional latent space. *Psychometrika*, 39, 111–121. (General polytomous information formula that Muraki 1993 specializes to the GPCM.)
- Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika*, 66(3), 331–342. (Source for the `lz_star` correction in `compute_person_fit_indices()` when person estimates come from the JML/fixed-effect route. MML/EAP person scores are left uncorrected because EAP does not satisfy the Snijders estimating-equation setup.)
- Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.
- Linacre, J. M. (2002). What do Infit and Outfit, mean-square and standardized mean? *Rasch Measurement Transactions*, 16(2), 878.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47, 149–174.
- Orlando, M., & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous item response theory models. *Applied Psychological Measurement*, 24, 50–64.
- Orlando, M., & Thissen, D. (2003). Further investigation of the performance of S-X2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement*, 27, 289–298.

- Sinharay, S., Johnson, M. S., & Stern, H. S. (2006). Posterior predictive assessment of item response theory models. *Applied Psychological Measurement*, 30, 298–321.
- Sinharay, S., & Monroe, S. (2025). Assessment of fit of item response theory models: A critical review of the status quo and some future directions. *British Journal of Mathematical and Statistical Psychology*, 78, 711–733.
- Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press.
- Wright, B. D., & Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions*, 8(3), 370.
- Wilson, E. B., & Hilferty, M. M. (1931). The distribution of chi-square. *Proceedings of the National Academy of Sciences of the United States of America*, 17(12), 684-688.

Model selection

RSM vs PCM

The Rating Scale Model (RSM; Andrich, 1978) assumes all levels of the step facet share identical threshold parameters. The Partial Credit Model (PCM; Masters, 1982) allows each level of the `step_facet` to have its own set of thresholds on the package's shared observed score scale. Use RSM when the rating rubric is identical across all items/criteria; use PCM when category boundaries are expected to vary by item or criterion. In the current implementation, PCM assumes one common observed score support across the fitted data, so it should not be described as a fully mixed-category model with arbitrary item-specific category counts.

MML vs JML

Marginal Maximum Likelihood (MML) integrates over the person ability distribution using Gauss-Hermite quadrature and does not directly estimate person parameters; person estimates are computed post-hoc via Expected A Posteriori (EAP). Joint Maximum Likelihood (JML) estimates all person and facet parameters simultaneously as fixed effects; `"JMLE"` remains a backward-compatible alias.

MML is generally preferred for smaller samples because it avoids the incidental-parameter problem of JML. JML does not assume a normal person distribution and can be lighter computationally in some settings, which may be an advantage when the population shape is strongly non-normal.

See `fit_mfrm()` for usage.

Fixed-calibration scoring after fitting

`predict_mfrm_units()` and `sample_mfrm_plausible_values()` score future or partially observed persons on a quadrature grid under the fitted scoring basis. For ordinary `MML` fits, these summaries inherit the fitted marginal calibration directly. For latent-regression `MML` fits, they use the fitted one-dimensional conditional normal population model and therefore require one-row-per-person background data for the scored units when the fitted population model includes covariates. Intercept-only latent-regression fits (`population_formula = ~ 1`) can reconstruct that minimal person table from the scored person IDs. For `JML` fits, `mfrmr` uses the fitted facet and step parameters together with a standard normal reference prior introduced only for the post hoc scoring layer. This is useful for practical fixed-scale scoring, but it should still be described as a limited approximation rather than as full ConQuest-style population modeling.

Current ConQuest overlap

The package now includes a first-version latent-regression `MML` branch, but the overlap with ConQuest should still be described conservatively. The documented overlap is: ordered-response `RSM`

/ PCM, one latent dimension, a conditional-normal person population model, and person covariates supplied through an explicit one-row-per-person table and expanded through the package-built model matrix. Categorical person covariates carry fitted levels and contrasts into scoring. This is a scoped overlap, not a claim of broad ConQuest numerical equivalence for arbitrary imported design matrices, multidimensional models, imported design specifications, or the full plausible-values workflow.

Author(s)

Maintainer: Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp> (ORCID) [copyright holder]

Authors:

- Ryuya Komuro <ryuya.komuro.c4@tohoku.ac.jp> (ORCID) [copyright holder]

See Also

Useful links:

- https://github.com/Ryuya-dot-com/mfrmr
- Report bugs at https://github.com/Ryuya-dot-com/mfrmr/issues

Examples

```
mfrm_threshold_profiles()
list_mfrmr_data()


toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  model = "RSM",
  quad_points = 7
)
diag <- diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")
summary(diag)
```

---

analyze_dff *Differential facet functioning analysis*

Description

Tests whether the difficulty of facet levels differs across a grouping variable (e.g., whether rater severity differs for male vs. female examinees, or whether item difficulty differs across rater subgroups).

`analyze_dif()` is retained for compatibility with earlier package versions. In many-facet workflows, prefer `analyze_dff()` as the primary entry point.

Usage

```
analyze_dff(
  fit,
  diagnostics,
  facet,
  group,
  data = NULL,
  focal = NULL,
  method = c("residual", "refit"),
  min_obs = 10,
  p_adjust = "holm"
)

analyze_dif(...)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`.
`facet`	Character scalar naming the facet whose elements are tested for differential functioning (for example, `"Criterion"` or `"Rater"`).
`group`	Character scalar naming the column in the data that defines the grouping variable (e.g., `"Gender"`, `"Site"`).
`data`	Optional data frame containing at least the group column and the same person/facet/score columns used to fit the model. If `NULL` (default), mfrmr tries to recover the data from `fit$prep$data`. That slot only holds the columns that `fit_mfrm()` actually modelled; if the grouping column was not among them (common for DIF screening), pass the original data frame via `data = <df>` explicitly. The same applies when the fit object has been serialized without the prep slot.
`focal`	Optional character vector of group levels to treat as focal. If `NULL` (default), all pairwise group comparisons are performed.
`method`	Analysis method: `"residual"` (default) uses the fitted model's residuals without re-estimation; `"refit"` re-estimates the model within each group subset. The residual method is faster and avoids convergence issues with small subsets.
`min_obs`	Minimum number of observations per cell (facet-level x group). Cells below this threshold are flagged as sparse and their statistics set to `NA`. Default `10`.

p_adjust	Method for multiple-comparison adjustment, passed to stats::p.adjust(). Default is "holm".
...	Passed directly to analyze_dff().

Details

Differential facet functioning (DFF) occurs when the difficulty or severity of a facet element differs across subgroups of the population, after controlling for overall ability. In an MFRM context this generalises classical DIF (which applies to items) to any facet: raters, criteria, tasks, etc.

Differential functioning is a threat to measurement fairness: if Criterion 1 is harder for Group A than Group B at the same ability level, the measurement scale is no longer group-invariant.

Two methods are available:

Residual method (method = "residual"): Uses the existing fitted model's observation-level residuals. For each facet-level $\times$ group cell, the observed and expected score sums are aggregated and a standardized residual is computed as:

$$z = \frac{\sum(X_{obs} - E_{exp})}{\sqrt{\sum \mathrm{Var}}}$$

Pairwise contrasts between groups compare the mean observed-minus-expected difference for each facet level, with uncertainty summarized by a Welch/Satterthwaite approximation. This method is fast, stable with small subsets, and does not require re-estimation. Because the resulting contrast is not a logit-scale parameter difference, the residual method is treated as a screening procedure rather than an ETS-style classifier.

Refit method (method = "refit"): Subsets the data by group, refits the MFRM model within each subset, anchors all non-target facets back to the baseline calibration when possible, and compares the resulting facet-level estimates using a Welch t-statistic:

$$t = \frac{\hat{\delta}_1 - \hat{\delta}_2}{\sqrt{SE_1^2 + SE_2^2}}$$

This provides group-specific parameter estimates on a common scale when linking anchors are available, but is slower and may encounter convergence issues with small subsets. ETS categories are reported only for contrasts whose subgroup calibrations retained enough linking anchors to support a common-scale interpretation and whose subgroup precision remained on the package's model-based MML path.

When facet refers to an item-like facet (for example Criterion), this recovers the familiar DIF case. When facet refers to raters or prompts/tasks, the same machinery supports DRF/DPF-style analyses.

For the refit method only, effect size is classified following the ETS (Educational Testing Service) DIF guidelines when subgroup calibrations are both linked and eligible for model-based inference:

- **A (Negligible)**: $|\Delta| < 0.43$ logits
- **B (Moderate)**: $0.43 \leq |\Delta| < 0.64$ logits
- **C (Large)**: $|\Delta| \geq 0.64$ logits

Multiple comparisons are adjusted using Holm's step-down procedure by default, which controls the family-wise error rate without assuming independence. Alternative methods (e.g., "BH" for false discovery rate) can be specified via p_adjust.

Value

An object of class `mfrm_dff` (with compatibility class `mfrm_dif`) with:

- `dif_table`: data.frame of differential-functioning contrasts.
- `cell_table`: (residual method) per-cell detail table.
- `summary`: counts by screening or ETS classification.
- `group_fits`: (refit method) per-group facet estimates.
- `gpcm_boundary`: for bounded `GPCM` fits, a capability-boundary table.
- `config`: list with facet, group, method, min_obs, p_adjust settings.

Choosing a method

In most first-pass DFF screening, start with `method = "residual"`. It is faster, reuses the fitted model, and is less fragile in smaller subsets. Use `method = "refit"` when you specifically want group-specific parameter estimates and can tolerate extra computation. Both methods should yield similar conclusions when sample sizes are adequate ($N \geq 100$ per group is a useful guideline for stable differential-functioning detection).

Interpreting output

- `$dif_table`: one row per facet-level x group-pair with contrast, SE, t-statistic, p-value, adjusted p-value, effect metric, and method-appropriate classification. Includes `Method`, `N_Group1`, `N_Group2`, `EffectMetric`, `ClassificationSystem`, `ContrastBasis`, `SEBasis`, `StatisticLabel`, `ProbabilityMetric`, `DFBasis`, `ReportingUse`, `PrimaryReportingEligible`, and `sparse` columns.
- `$cell_table`: (residual method only) per-cell detail with N, ObsScore, ExpScore, ObsExpAvg, StdResidual.
- `$summary`: counts by screening result (`method = "residual"`) or ETS category plus linked-screening and insufficient-linking rows (`method = "refit"`).
- `$group_fits`: (refit method only) list of per-group facet estimates and subgroup linking diagnostics.
- `$gpcm_boundary`: for bounded `GPCM` fits, a capability-boundary table marking the DFF/DIF output as caveated screening evidence.

GPCM boundary

For bounded `GPCM`, DFF/DIF rows are available as slope-aware screening evidence over the fitted expected-score and residual scale. Keep residual-method contrasts and interaction cells in screening language. Refit contrasts require explicit subgroup linking and precision support before stronger subgroup-comparison language is used.

Typical workflow

1. Fit a model with `fit_mfrm()`. For `RSM` / `PCM` fairness review, prefer `method = "MML"`.
2. Run `diagnose_mfrm()` and, for `RSM` / `PCM`, prefer `diagnostic_mode = "both"` so legacy and strict marginal screens remain visible together.

3. Run `analyze_dff(fit, diagnostics, facet = "Criterion", group = "Gender", data = my_data)`.
4. Inspect `$dif_table` for flagged levels and `$summary` for counts.
5. Use `dif_interaction_table()` when you need cell-level diagnostics.
6. Use `plot_dif_heatmap()` or `dif_report()` for communication.

See Also

`fit_mfrm()`, `estimate_bias()`, `compare_mfrm()`, `dif_interaction_table()`, `plot_dif_heatmap()`, `dif_report()`, `subset_connectivity_report()`, mfrmr_linking_and_dff

Examples

```
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
dff <- analyze_dff(fit, diag, facet = "Rater", group = "Group", data = toy)
dff$summary
# Look for: a small `FlaggedPairs` count relative to `Pairs`. Under
#   method = "residual", `ClassificationSystem` is "screening", not
#   ETS. "Screen positive" rows are prompts for substantive review.
head(dff$dif_table[, c("Level", "Group1", "Group2", "Contrast",
                       "Classification", "ClassificationSystem")])
# The residual contrast is an observed-minus-expected average contrast
# between groups. It is useful for screening, but it is not an ETS
# A/B/C logit-delta classification.
dff_refit <- analyze_dff(fit, diag, facet = "Rater", group = "Group",
                         data = toy, method = "refit")
unique(dff_refit$dif_table$ClassificationSystem)
# Look for: "ETS" only when subgroup calibration, linking, and precision
#   checks all support a common-scale model-based contrast.
sc <- subset_connectivity_report(fit, diagnostics = diag)
plot(sc, type = "design_matrix", draw = FALSE)
if ("ScaleLinkStatus" %in% names(dff_refit$dif_table)) {
  unique(dff_refit$dif_table$ScaleLinkStatus)
}
# Look for: "linked" in `ScaleLinkStatus` confirms the focal and
#   reference groups share enough common elements for a comparable
#   contrast; "demoted_*" rows lose linking under the refit branch
#   and should be read as exploratory.
```

`analyze_facet_equivalence` *Analyze practical equivalence within a facet*

Description

Analyze practical equivalence within a facet

Usage

```
analyze_facet_equivalence(
  fit,
  diagnostics = NULL,
  facet = NULL,
  equivalence_bound = 0.5,
  ci_level = 0.95,
  conf_level = NULL
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are computed with residual_pca = "none".
facet	Character scalar naming the non-person facet to evaluate. If NULL, the function prefers a rater-like facet and otherwise uses the first model facet.
equivalence_bound	Practical-equivalence bound in logits. Default 0.5 is a moderate bound intended as a starting point, not a universal threshold. The TOST/ROPE result depends on both the bound *and* the per-level standard errors, so in small or high-variance designs the test may fail to reject non-equivalence simply because the SEs are wide. Choose equivalence_bound based on the smallest difference that would be practically meaningful in your assessment context (commonly 0.3 to 0.5 logits for rater-mediated designs) and check $summary for per-level SE magnitude before drawing conclusions.
ci_level	Confidence level used for the forest-style interval view. Default 0.95.
conf_level	Deprecated alias for ci_level, retained for backward compatibility. Supplying a non-NULL value overrides ci_level and emits a one-time deprecation warning. Will be removed in a future release. Default 0.95.

Details

This function tests whether facet elements (e.g., raters) are similar enough to be treated as practically interchangeable, rather than merely testing whether they differ significantly. This is the key distinction from a standard chi-square heterogeneity test: absence of evidence for difference is not evidence of equivalence.

The function uses existing facet estimates and their standard errors from diagnostics$measures; no re-estimation is performed.

The bundle combines four complementary views:

1. **Fixed chi-square test**: tests H_0: all element measures are equal. A non-significant result is *necessary but not sufficient* for interchangeability. It is reported as context, not as direct evidence of equivalence.

2. **Pairwise TOST (Two One-Sided Tests)**: for each pair of elements, tests whether the difference falls within $\pm$`equivalence_bound`. The TOST procedure (Schuirmann, 1987) rejects the null hypothesis of *non-equivalence* when both one-sided tests are significant at level α. A pair is declared "Equivalent" when the TOST p-value < 0.05.
3. **BIC-based Bayes-factor heuristic**: an approximate screening tool (not full Bayesian inference) that compares the evidence for a common-facet model (all elements equal) against a heterogeneity model (elements differ) via $\mathrm{BF}_{01} \approx \exp((\mathrm{BIC}_{H_1} - \mathrm{BIC}_{H_0})/2)$ (Kass & Raftery, 1995). Values > 3 favour the common-facet model; < 1/3 favour heterogeneity.
4. **ROPE-style grand-mean proximity**: the proportion of each element's normal-approximation confidence distribution that falls within $\pm$`equivalence_bound` of the weighted grand mean. This is a descriptive proximity summary, not a Bayesian ROPE decision rule around a prespecified null value.

Choosing `equivalence_bound`: the default of 0.5 logits is a moderate criterion. For high-stakes certification, 0.3 logits may be appropriate; for exploratory or low-stakes contexts, 1.0 logits may suffice. The bound should reflect the smallest difference that would be practically meaningful in your application.

Value

A named list with class `mfrm_facet_equivalence`.

What this analysis means

`analyze_facet_equivalence()` is a practical-interchangeability screen. It asks whether facet levels are close enough, under a user-defined logit bound, to be treated as practically similar for the current use case.

What this analysis does not justify

- A non-significant chi-square result is not evidence of equivalence.
- Forest/ROPE displays are descriptive and do not replace the pairwise TOST decision rule.
- The BIC-based Bayes-factor summary is a heuristic screen, not a full Bayesian equivalence analysis.

Interpreting output

Start with `summary$Decision`, which is a conservative summary of the pairwise TOST results. Then use the remaining tables as context:

- `chi_square`: is there broad heterogeneity in the facet?
- `pairwise`: which specific pairs meet the practical-equivalence bound?
- `rope` / `forest`: how close is each level to the facet grand mean?

Smaller `equivalence_bound` values make the criterion stricter. If the decision is `"partial_pairwise_equivalence"`, that means some pairwise contrasts satisfy the practical-equivalence bound but not all of them do.

Decision rule

The final Decision is a pairwise TOST summary rather than a global equivalence proof. If all pairwise contrasts satisfy the practical- equivalence bound, the facet is labeled "all_pairs_equivalent". If at least one, but not all, pairwise contrasts are equivalent, the facet is labeled "partial_pairwise_equivalence". If no pairwise contrasts meet the practical-equivalence bound, the facet is labeled "no_pairwise_equivalence_establishe The chi-square, Bayes-factor, and grand-mean proximity summaries are reported as descriptive context.

How to read the main outputs

- summary: one-row pairwise-TOST decision summary and aggregate context.
- pairwise: pair-level TOST detail; use this for the primary inferential read.
- chi_square: broad heterogeneity screen.
- rope / forest: level-wise proximity to the weighted grand mean.

Recommended next step

If the result is borderline or high-stakes, re-run the analysis with a tighter or looser equivalence_bound, then inspect pairwise and plot_facet_equivalence() before deciding how strongly to claim interchangeability.

Typical workflow

1. Fit a model with fit_mfrm().
2. Run analyze_facet_equivalence() for the facet you want to screen.
3. Read summary and chi_square first.
4. Use plot_facet_equivalence() to inspect which levels drive the result.

Output

The returned bundle has class mfrm_facet_equivalence and includes:

- summary: one-row overview with convergent decision
- chi_square: fixed chi-square / separation summary
- pairwise: pairwise TOST detail table
- rope: element-wise ROPE probabilities around the weighted grand mean
- forest: element-wise estimate, confidence interval, and ROPE status
- settings: applied facet and threshold settings

References

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association, 90*(430), 773-795.

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics, 15*(6), 657-680.

See Also

facets_chisq_table(), fair_average_table(), plot_facet_equivalence()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
eq <- analyze_facet_equivalence(fit, facet = "Rater")
eq$summary[, c("Facet", "Elements", "Decision", "MeanROPE")]
head(eq$pairwise[, c("ElementA", "ElementB", "Equivalent")])
```

---

analyze_hierarchical_structure

Analyze the hierarchical structure of a rating design

Description

One-stop review that combines the nesting, cross-tabulation, ICC, and design-effect reports into a single object. Designed to be reused by the publication-workflow surface: its summary feeds into `reporting_checklist()`, and its tables are picked up by `build_mfrm_manifest()` for reproducibility bundles.

Usage

```
analyze_hierarchical_structure(
  data,
  facets = NULL,
  person = "Person",
  score = "Score",
  compute_icc = TRUE,
  ci_method = c("none", "profile", "boot"),
  ci_level = 0.95,
  ci_boot_reps = 1000L,
  ci_boot_seed = NULL,
  igraph_layout = TRUE,
  icc_ci_method = NULL,
  icc_ci_level = NULL,
  icc_ci_boot_reps = NULL,
  icc_ci_boot_seed = NULL
)
```

Arguments

`data`	Data frame in long format, or an `mfrm_fit` (its `prep$data` is used).
`facets`	Character vector of facet column names. When `data` is an `mfrm_fit`, defaults to `fit$prep$facet_names`.

person	Person column name. Defaults to "Person".
score	Score column name. Defaults to "Score".
compute_icc	Logical; if TRUE and lme4 is available, adds ICC and design-effect tables.
ci_method	ICC confidence-interval method passed through to compute_facet_icc(). One of "none" (default, point estimate only), "profile", or "boot". Deprecated alias: icc_ci_method (kept for backward compatibility, emits a lifecycle warning).
ci_level	Confidence level when ci_method != "none"; default 0.95. Deprecated alias: icc_ci_level.
ci_boot_reps	Number of bootstrap replicates when ci_method = "boot". Default 1000. Deprecated alias: icc_ci_boot_reps.
ci_boot_seed	Optional RNG seed for reproducible bootstrap CIs. Deprecated alias: icc_ci_boot_seed.
igraph_layout	Logical; if TRUE and igraph is available, adds a connectivity component summary using a bipartite graph over person x facet levels.
icc_ci_method, icc_ci_level, icc_ci_boot_reps, icc_ci_boot_seed	Deprecated spellings of the ci_* arguments above, retained for one release. Supplying a non-NULL value routes through lifecycle::deprecate_warn() and overrides the canonical ci_* argument.

Value

A list of class mfrm_hierarchical_structure with:

- nesting: output of detect_facet_nesting().
- crosstabs: list of pairwise observation-count data.frames (long format, suitable for heatmap plotting).
- icc: output of compute_facet_icc() when requested.
- design_effect: output of compute_facet_design_effect() when requested.
- connectivity: named list with bipartite-graph component summary when igraph is available.
- summary: one-row summary used by downstream reporting helpers.
- facets: character vector of facet names that were reviewed (echoed for downstream reporting helpers that need to label rows by review scope).

Interpreting output

- nesting: a detect_facet_nesting() object with every facet pair classified as Crossed / Partially / Near-perfectly / Fully nested.
- crosstabs: list of (LevelA, LevelB, N) long-format tables, one per facet pair. Plot via plot(x, type = "crosstab", pair = "FacetA__FacetB").
- icc: per-facet variance shares. See compute_facet_icc() for the two-scale interpretation.
- design_effect: Kish (1965) Deff and EffectiveN.
- connectivity: number of bipartite components linking Person x facet levels. A single component is required for a common measurement scale; multiple components indicate a disconnected design.

Typical workflow

1. Optional: fit the MFRM with `fit_mfrm()`.
2. Call `analyze_hierarchical_structure(fit)` (or on the raw data).
3. Read `summary(x)` for the condensed view.
4. Feed the object to `reporting_checklist()` and `build_mfrm_manifest()` to record the review in publication bundles. `build_apa_outputs()` uses the fit-level `FacetSampleSizeFlag` to add a Methods sentence automatically.

References

McEwen, M. R. (2018). *The effects of incomplete rating designs on results from many-facets-Rasch model analyses* (Doctoral thesis, Brigham Young University). https://scholarsarchive.byu.edu/etd/6689/

Linacre, J. M. (2026). *A User's Guide to FACETS, Version 4.5.0*. Winsteps.com. https://www.winsteps.com/facets.htm

Kish, L. (1965). *Survey Sampling*. New York: Wiley.

Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine, 15*(2), 155-163.

See Also

`detect_facet_nesting()`, `facet_small_sample_review()`, `compute_facet_icc()`, `compute_facet_design_effect()`, `reporting_checklist()`, `build_mfrm_manifest()`, `fit_mfrm()`.

Examples

```
toy <- load_mfrmr_data("example_core")
hs <- analyze_hierarchical_structure(toy,
                                     facets = c("Rater", "Criterion"),
                                     compute_icc = FALSE,
                                     igraph_layout = FALSE)
summary(hs)


# Full review when lme4 and igraph are available.
if (requireNamespace("lme4", quietly = TRUE) &&
    requireNamespace("igraph", quietly = TRUE)) {
  hs_full <- analyze_hierarchical_structure(toy,
                                            facets = c("Rater", "Criterion"))
  summary(hs_full)
  plot(hs_full, type = "icc")
}
```

`analyze_residual_pca` *Run exploratory residual PCA summaries*

Description

Legacy-compatible residual diagnostics can be inspected in two ways:

1. overall residual PCA on the person x combined-facet matrix
2. facet-specific residual PCA on person x facet-level matrices

Usage

```
analyze_residual_pca(
  diagnostics,
  mode = c("overall", "facet", "both"),
  facets = NULL,
  pca_max_factors = 10L,
  parallel = FALSE,
  parallel_reps = 200L,
  parallel_quantile = 0.95,
  parallel_method = c("residual_permutation"),
  seed = NULL
)
```

Arguments

`diagnostics`	Output from `diagnose_mfrm()` or `fit_mfrm()`.
`mode`	`"overall"`, `"facet"`, or `"both"`.
`facets`	Optional subset of facets for facet-specific PCA.
`pca_max_factors`	Maximum number of retained components.
`parallel`	Logical; if `TRUE`, add residual-permutation parallel analysis to the PCA tables.
`parallel_reps`	Number of residual permutations used when `parallel = TRUE`.
`parallel_quantile`	Upper null quantile used as the exploratory comparison cutoff. The default (`0.95`) follows the common parallel analysis convention.
`parallel_method`	Parallel-analysis null method. Currently `"residual_permutation"` is implemented: standardized residuals are permuted within each residual column, preserving each column's residual distribution and missingness pattern while breaking residual association.
`seed`	Optional integer seed for reproducible residual permutations.

Details

The function works on standardized residual structures derived from `diagnose_mfrm()`. When a fitted object from `fit_mfrm()` is supplied, diagnostics are computed internally.

Conceptually, this follows the Rasch residual-PCA tradition of examining structure in model residuals after the primary Rasch dimension has been extracted. In `mfrmr`, however, the implementation is an **exploratory many-facet adaptation**: it works on standardized residual matrices built as person x combined-facet or person x facet-level layouts, rather than reproducing FACETS/Winsteps residual-contrast tables one-to-one.

Residual PCA should therefore be reported as residual-structure evidence, not as a formal proof of unidimensionality. It also should not be described as DIMTEST or UNIDIM: those essential-unidimensionality tests require a separate item-response-layer definition that is not uniquely determined by a many-facet long data set. In applied MFRM reporting, residual PCA is best triangulated with global residual fit, element fit, and Q3-style local-dependence screens.

Output tables use:

- `Component`: principal-component index (1, 2, ...)
- `Eigenvalue`: eigenvalue for each component
- `Proportion`: component variance proportion
- `Cumulative`: cumulative variance proportion

When `parallel = TRUE`, the variance tables additionally include data-driven null summaries:

- `ParallelMean`: mean permuted-residual eigenvalue
- `ParallelCutoff`: `parallel_quantile` cutoff of permuted eigenvalues
- `ExcessOverParallelCutoff`: observed eigenvalue minus the cutoff
- `ExceedsParallelCutoff`: whether the observed eigenvalue exceeds the permutation cutoff

The default `parallel_reps = 200` is intended as a practical review setting. For stable final reporting of the 95% cutoff, use a larger value when the residual matrix size makes that computationally reasonable.

For `mode = "facet"` or `"both"`, `by_facet_table` additionally includes a `Facet` column.

`summary(pca)` is supported through `summary()`. `plot(pca)` is dispatched through `plot()` for class `mfrm_residual_pca`. Available types include `"overall_scree"`, `"facet_scree"`, `"overall_parallel_scree"`, `"facet_parallel_scree"`, `"overall_parallel_excess"`, `"facet_parallel_excess"`, `"overall_loadings"`, and `"facet_loadings"`.

Value

A named list with:

- `mode`: resolved mode used for computation
- `facet_names`: facets analyzed
- `overall`: overall PCA bundle (or `NULL`)
- `by_facet`: named list of facet PCA bundles
- `overall_table`: variance table for overall PCA

- by_facet_table: stacked variance table across facets
- parallel_settings, parallel_overall_table, parallel_by_facet_table, and parallel_status: returned for every call; the parallel tables are populated when parallel = TRUE
- errors: named list of any per-facet PCA errors that were caught and turned into NA_real_ rows in the variance tables (e.g., psych::principal() failure on a near-singular residual matrix). The list is empty when every facet PCA succeeded.
- warnings: named list of non-fatal PCA warnings captured from the underlying PCA engine. These indicate exploratory boundary conditions, not confirmatory evidence.

Interpreting output

Use overall_table first:

- early components with noticeably larger eigenvalues or proportions suggest stronger residual structure that may deserve follow-up. Small early components can be described as evidence consistent with the specified one-dimensional facet structure only when fit and local-dependence screens tell the same story.

Then inspect by_facet_table:

- helps localize which facet contributes most to residual structure.

Finally, inspect loadings via plot_residual_pca() to identify which variables/elements drive each component.

References

The residual-PCA idea follows the Rasch residual-structure literature, especially Linacre's discussions of principal components of Rasch residuals. The current mfrmr implementation should be interpreted as an exploratory extension for many-facet workflows rather than as a direct reproduction of a single FACETS/Winsteps output table.

The optional parallel analysis follows Horn's data-driven eigenvalue comparison logic and later recommendations to compare observed eigenvalues with high quantiles of an empirical null distribution. Because mfrmr applies it to standardized Rasch-family residual matrices, the null distribution is generated by within-column residual permutation rather than by simulating raw item scores.

- Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30, 179-185.
- Glorfeld, L. W. (1995). An improvement on Horn's parallel analysis methodology for selecting the correct number of factors to retain. *Educational and Psychological Measurement*, 55, 377-393.
- Hayton, J. C., Allen, D. G., & Scarpello, V. (2004). Factor retention decisions in exploratory factor analysis: A tutorial on parallel analysis. *Organizational Research Methods*, 7, 191-205.
- Timmerman, M. E., & Lorenzo-Seva, U. (2011). Dimensionality assessment of ordered polytomous items with parallel analysis. *Psychological Methods*, 16, 209-220.
- Linacre, J. M. (1998). *Structure in Rasch residuals: Why principal components analysis (PCA)?* Rasch Measurement Transactions, 12(2), 636.

- Linacre, J. M. (1998). *Detecting multidimensionality: Which residual data-type works best?* Journal of Outcome Measurement, 2(3), 266-283.
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly*, 2(3), 197-221.
- Yamashita, T. (2024). An application of many-facet Rasch measurement to evaluate automated essay scoring: A case of ChatGPT-4.0. *Research Methods in Applied Linguistics*, 3(3), 100133.
- Uto, M. (2021). A multidimensional generalized many-facet Rasch model for rubric-based performance assessment. *Behaviormetrika*, 48(2), 425-457.
- Aryadoust, V., Ng, L. Y., & Sayama, H. (2021). A comprehensive review of Rasch measurement in language assessment: Recommendations and guidelines for research. *Language Testing*, 38(1), 6-40.
- Tseng, W.-T. (2016). Measuring English vocabulary size via computerized adaptive testing. *Computers & Education*, 97, 69-85.

Typical workflow

1. Fit model and run `diagnose_mfrm()` with `residual_pca = "none"` or `"both"`.
2. Call `analyze_residual_pca(..., mode = "both")`.
3. Review `summary(pca)`, then plot scree/loadings.
4. Cross-check with fit/misfit diagnostics before conclusions.

See Also

`diagnose_mfrm()`, `plot_residual_pca()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "both")
pca <- analyze_residual_pca(diag, mode = "both")
pca2 <- analyze_residual_pca(fit, mode = "both")
summary(pca)
p <- plot_residual_pca(pca, mode = "overall", plot_type = "scree", draw = FALSE)
p$data$plot
head(p$data)
pca_pa <- analyze_residual_pca(diag, mode = "overall", parallel = TRUE, parallel_reps = 10)
head(pca_pa$overall_table)
head(pca$overall_table)
```

anchor_to_baseline *Fit new data anchored to a baseline calibration*

Description

Re-estimates a fitted many-facet model on new data while holding selected facet parameters fixed at the values from a previous (baseline) calibration. This is the standard workflow for placing new data onto an existing scale, linking test forms, or carrying a baseline calibration across administration windows. For bounded GPCM, treat this as direct exploratory anchor/drift support rather than as the package's formal linking-synthesis route.

Usage

```
anchor_to_baseline(
  new_data,
  baseline_fit,
  person,
  facets,
  score,
  anchor_facets = NULL,
  include_person = FALSE,
  weight = NULL,
  model = NULL,
  method = NULL,
  anchor_policy = "warn",
  ...
)

## S3 method for class 'mfrm_anchored_fit'
print(x, ...)

## S3 method for class 'mfrm_anchored_fit'
summary(object, ...)

## S3 method for class 'summary.mfrm_anchored_fit'
print(x, ...)
```

Arguments

new_data	Data frame in long format (one row per rating).
baseline_fit	An mfrm_fit object from a previous calibration.
person	Character column name for person/examinee.
facets	Character vector of facet column names.
score	Character column name for the rating score.
anchor_facets	Character vector of facets to anchor (default: all non-Person facets).

include_person	If TRUE, also anchor person estimates.
weight	Optional character column name for observation weights.
model	Scale model override; defaults to baseline model.
method	Estimation method override; defaults to baseline method.
anchor_policy	How to handle anchor issues: "warn", "error", "silent".
...	Ignored.
x	An mfrm_anchored_fit object.
object	An mfrm_anchored_fit object (for summary).

Details

This function automates the baseline-anchored calibration workflow:

1. Extracts anchor values from the baseline fit using make_anchor_table().
2. Re-estimates the model on new_data with those anchors fixed via fit_mfrm(..., anchors = anchor_table).
3. Runs diagnose_mfrm() on the anchored fit.
4. Computes element-level differences (new estimate minus baseline estimate) for every common element.

The model and method arguments default to the baseline fit's settings so the calibration framework remains consistent. Elements present in the anchor table but absent from the new data are handled according to anchor_policy: "warn" (default) emits a message, "error" stops execution, and "silent" ignores silently.

The returned drift table is best interpreted as an anchored consistency check. When a facet is fixed through anchor_facets, those anchored levels are constrained in the new run, so their reported differences are not an independent drift analysis. For genuine cross-wave drift monitoring, fit the waves separately and use detect_anchor_drift() on the resulting fits.

Element-level differences are calculated for every element that appears in both the baseline and the new calibration:

$$\Delta_e = \hat{\delta}_{e,\text{new}} - \hat{\delta}_{e,\text{base}}$$

An element is **flagged** when $|\Delta_e| > 0.5$ logits or $|\Delta_e / SE_{\Delta_e}| > 2.0$, where $SE_{\Delta_e} = \sqrt{SE^2_{\text{base}} + SE^2_{\text{new}}}$.

Value

Object of class mfrm_anchored_fit with components:

fit The anchored mfrm_fit object.

diagnostics Output of diagnose_mfrm() on the anchored fit.

baseline_anchors Anchor table extracted from the baseline.

drift Tibble of element-level drift statistics.

Which function should I use?

- Use `anchor_to_baseline()` when you have one new dataset and want to place it directly on a baseline scale.
- Use `detect_anchor_drift()` when you already have multiple fitted waves and want to compare their stability.
- Use `build_equating_chain()` when you need cumulative offsets across an ordered series of waves.

Interpreting output

- `$drift`: one row per common element with columns `Facet`, `Level`, `Baseline`, `New`, `Drift`, `SE_Baseline`, `SE_New`, `SE_Diff`, `Drift_SE_Ratio`, and `Flag`. Read this as an anchored consistency table. Small absolute differences indicate that the anchored re-fit stayed close to the baseline scale. Flagged rows warrant review, but they are not a substitute for a separate drift study on unanchored common elements.
- `$fit`: the full anchored `mfrm_fit` object, usable with `diagnose_mfrm()`, `measurable_summary_table()`, etc.
- `$diagnostics`: pre-computed diagnostics for the anchored calibration.
- `$baseline_anchors`: the anchor table fed to `fit_mfrm()`, useful for reviewing which elements were constrained.

Typical workflow

1. Fit the baseline model: `fit1 <- fit_mfrm(...)`.
2. Collect new data (e.g., a later administration).
3. Call `res <- anchor_to_baseline(new_data, fit1, ...)`.
4. Inspect `summary(res)` to confirm the anchored run remains close to the baseline scale.
5. For multi-wave drift monitoring, fit waves separately and pass the fits to `detect_anchor_drift()` or `build_equating_chain()`.

See Also

`fit_mfrm()`, `make_anchor_table()`, `detect_anchor_drift()`, `diagnose_mfrm()`, `build_equating_chain()`, mfrmr_linking_and_dff

Examples

```
d1 <- load_mfrmr_data("study1")
keep1 <- unique(d1$Person)[1:15]
d1 <- d1[d1$Person %in% keep1, , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
d2 <- load_mfrmr_data("study2")
keep2 <- unique(d2$Person)[1:15]
d2 <- d2[d2$Person %in% keep2, , drop = FALSE]
res <- anchor_to_baseline(d2, fit1, "Person",
                          c("Rater", "Criterion"), "Score",
                          anchor_facets = "Criterion")
```

```
summary(res)
head(res$drift[, c("Facet", "Level", "Drift", "Flag")])
res$baseline_anchors[1:3, ]
```

---

apa_table *Build APA-style table output using base R structures*

Description

Build APA-style table output using base R structures

Usage

```
apa_table(
  x,
  which = NULL,
  diagnostics = NULL,
  digits = 2,
  caption = NULL,
  note = NULL,
  bias_results = NULL,
  context = list(),
  whexact = FALSE,
  branch = c("apa", "facets")
)
```

Arguments

x	A data.frame, mfrm_fit, summary() output supported by build_summary_table_bundle(), an mfrm_summary_table_bundle, diagnostics list, or bias-result list.
which	Optional table selector when x has multiple tables.
diagnostics	Optional diagnostics from diagnose_mfrm() (used when x is mfrm_fit and which targets diagnostics tables).
digits	Number of rounding digits for numeric columns.
caption	Optional caption text.
note	Optional note text.
bias_results	Optional output from estimate_bias() used when auto-generating APA metadata for fit-based tables.
context	Optional context list forwarded when auto-generating APA metadata for fit-based tables.
whexact	Logical forwarded to APA metadata helpers.
branch	Output branch: "apa" for manuscript-oriented labels, "facets" for FACETS-aligned labels.

Details

This helper avoids styling dependencies and returns a reproducible base `data.frame` plus metadata.

Supported `which` values:

- For `mfrm_fit`: `"summary"`, `"person"`, `"facets"`, `"steps"`
- For `summary()` outputs or `mfrm_summary_table_bundle`: names listed in `build_summary_table_bundle(x)$table_`
- For diagnostics list: `"overall_fit"`, `"measures"`, `"fit"`, `"reliability"`, `"facets_chisq"`, `"bias"`, `"interactions"`, `"interrater_summary"`, `"interrater_pairs"`, `"obs"`
- For bias-result list: `"table"`, `"summary"`, `"chi_sq"`

Value

A list of class `apa_table` with fields:

- `table` (`data.frame`)
- `which`
- `caption`
- `note`
- `digits`
- `branch`, `style`

Interpreting output

- `table`: plain data.frame ready for export or further formatting.
- `which`: source component that produced the table.
- `caption`/`note`: manuscript-oriented metadata stored with the table.

Typical workflow

1. Build table object with `apa_table(...)`.
2. Inspect quickly with `summary(tbl)`.
3. Render base preview via `plot(tbl, ...)` or export `tbl$table`.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `build_apa_outputs()`, `reporting_checklist()`, mfrmr_reporting_and_apa

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
tbl <- apa_table(fit, which = "summary", caption = "Model summary", note = "Toy example")
tbl_facets <- apa_table(fit, which = "summary", branch = "facets")
fit_bundle <- build_summary_table_bundle(summary(fit))
tbl_from_summary <- apa_table(fit_bundle, which = "facet_overview")
summary(tbl)
```

```
p <- plot(tbl, draw = FALSE)
p_facets <- plot(tbl_facets, type = "numeric_profile", draw = FALSE)
p$data$plot
p_facets$data$plot
if (interactive()) {
  plot(
    tbl,
    type = "numeric_profile",
    main = "APA Table Numeric Profile (Customized)",
    palette = c(numeric_profile = "#2b8cbe", grid = "#d9d9d9"),
    label_angle = 45
  )
}
tbl$note
```

`apply_empirical_bayes_shrinkage` *Apply empirical-Bayes shrinkage to fitted non-person facet estimates*

Description

Post-hoc shrinkage helper that augments an `mfrm_fit` with James-Stein / empirical-Bayes shrunk estimates for each non-person facet. The shrinkage variance $\hat{\tau}^2$ is estimated by method of moments from the facet-level point estimates and their standard errors:

$$\hat{\tau}^2 = \max\left(0, \frac{1}{K}\sum_{j=1}^{K}\hat{\delta}_j^2 - \overline{\mathrm{SE}^2}\right),$$

where the first term is the population variance of the facet point estimates around their *known* mean of zero (the mfrmr sum-to-zero identification pins the facet mean exactly at 0, so no degree of freedom is consumed by mean estimation). The shrinkage factor is $B_j = \mathrm{SE}_j^2/(\hat{\tau}^2 + \mathrm{SE}_j^2)$, and the shrunk point / standard error are $\hat{\delta}_j^{EB} = (1 - B_j)\hat{\delta}_j$ and $\mathrm{SE}_j^{EB} = \sqrt{(1 - B_j)\mathrm{SE}_j^2}$. The posterior SE form treats $\hat{\tau}^2$ as known; it omits the Morris (1983, eqs. 4.1-4.2, p. 51) confidence-interval correction $v \cdot \hat{\delta}_j^2$ with $v = 2B_j^2/(K - r - 2)$, where r is the number of regression coefficients used to model the prior mean (under mfrmr's sum-to-zero pinning, $r = 0$, so the divisor is $K - 2$). This correction adds variance proportional to the squared deviation $\hat{\delta}_j^2$, accounting for uncertainty in $\hat{\tau}^2$. Under the equal-variance assumption $\hat{\delta}_j^2 \approx \hat{\tau}^2$, the omitted variance is on the order of $2/(K - 2)$ times the reported posterior variance $V(1 - B_j)$, so the true SE is approximately $\sqrt{1 + 2/(K - 2)}$ times the reported `ShrunkSE`. Magnitudes: SE understated by ~73\ at $K = 8$, ~7\ `ShrunkSE` as a lower bound rather than a calibrated posterior SE.

Usage

```
apply_empirical_bayes_shrinkage(
  fit,
  facet_prior_sd = NULL,
  shrink_person = FALSE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm() with a non-empty facets$others table.
facet_prior_sd	Optional numeric scalar. When supplied, the shrinkage variance is fixed at facet_prior_sd^2 instead of being estimated from the data. Useful when a prior is elicited from expert knowledge or a previous fit.
shrink_person	Logical. When TRUE, the same empirical-Bayes shrinkage is also applied to fit$facets$person. Default FALSE, since MML person estimates already reflect a N(0, sigma^2) prior.

Details

fit$facets$others gains ShrunkEstimate, ShrunkSE, and ShrinkageFactor columns, and fit$shrinkage_report records the per-facet $\hat{\tau}^2$, mean shrinkage, and effective degrees of freedom ($\text{EffectiveDF}_f = \sum_j (1 - B_j)$, which matches the "effective number of parameters" defined by Efron & Morris, 1973). The original Estimate / SE columns are preserved.

Value

The same mfrm_fit, with augmented columns and a new shrinkage_report list entry, and with fit$config$facet_shrinkage set to "empirical_bayes".

Typical workflow

1. Fit the model as usual with fit_mfrm().
2. Call apply_empirical_bayes_shrinkage(fit) when small-N facets are present (see facet_small_sample_review(
3. Report both the original and shrunk estimates in the manuscript, citing Efron & Morris (1973). build_apa_outputs() will add the sentence automatically when fit$config$facet_shrinkage is set.

References

Efron, B., & Morris, C. (1973). Combining possibly related estimation problems. *Journal of the Royal Statistical Society: Series B, 35*(3), 379-402.

Efron, B. (2021). *Empirical Bayes: Concepts and methods* (Technical report). Department of Statistics, Stanford University. https://efron.ckirby.su.domains/papers/2021EB-concepts-methods.pdf

Morris, C. N. (1983). Parametric empirical Bayes inference: Theory and applications. *Journal of the American Statistical Association, 78*(381), 47-55.

See Also

fit_mfrm() (which accepts facet_shrinkage directly), facet_small_sample_review(), compute_facet_icc().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
fit_eb <- apply_empirical_bayes_shrinkage(fit)
fit_eb$shrinkage_report
# Look for:
# - `Tau2` is the estimated between-level prior variance per facet.
#   `Tau2 = 0` means the data did not justify any pooling and the
#   shrunken estimates equal the raw estimates (`MeanShrinkage = 0`).
# - `MeanShrinkage` near 0 = little movement, near 1 = heavy pooling
#   toward 0. Small-N facets typically pull values further than
#   well-identified ones.
# - `EffectiveDF` is the implied "effective number of parameters"
#   (Efron & Morris 1973); EffectiveDF much smaller than the row
#   count of the facet means most levels were pooled together.
head(fit_eb$facets$others[, c("Facet", "Level", "Estimate",
                              "ShrunkEstimate", "ShrinkageFactor")])
# Look for: rows where `ShrinkageFactor` is large (close to 1) had
#   their estimates pulled most strongly toward the facet mean (0).
```

---

as.data.frame.mfrm_fit

Convert mfrm_fit to a tidy data.frame

Description

Returns all facet-level estimates (person and others) in a single tidy data.frame. Useful for quick interactive export: `write.csv(as.data.frame(fit), "results.csv")`.

Usage

```
## S3 method for class 'mfrm_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An `mfrm_fit` object from `fit_mfrm`.
`row.names`	Ignored (included for S3 generic compatibility).
`optional`	Ignored (included for S3 generic compatibility).
`...`	Additional arguments (ignored).

Details

This method returns four columns (`Facet`, `Level`, `Estimate`, `Extreme`) so that the result is easy to inspect, join, or write to disk.

Value

A data.frame with columns `Facet`, `Level`, `Estimate`, and `Extreme`. The `Extreme` column is populated for person rows from the extreme-score flag added in 0.1.6 (`"Min"` / `"Max"` / `NA`); non-person facet rows carry `NA` in that column by design.

Interpreting output

Person estimates are returned with `Facet = "Person"`. All non-person facets are stacked underneath in the same schema.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Convert with `as.data.frame(fit)` for a compact long-format export.
3. Join additional diagnostics later if you need SE or fit statistics.

See Also

`fit_mfrm`, `export_mfrm`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
head(as.data.frame(fit))
```

---

assess_mfrm_recovery *Assess whether recovery-simulation results are ready to use*

Description

Converts the numerical output from `evaluate_mfrm_recovery()` into a reviewer-facing adequacy checklist. The goal is not to impose one universal pass/fail rule; it is to make the main user questions explicit: Did the runs finish? Did the fitted models converge? Are uncertainty summaries available? Are coverage and Monte Carlo precision plausible? If practical RMSE or bias limits are supplied, which parameter groups need follow-up? For bounded `GPCM`, which slope-regime generator condition frames the recovery evidence?

Usage

```
assess_mfrm_recovery(
  x,
  min_reps = 30,
  min_success_rate = 0.95,
  min_convergence_rate = 0.95,
  min_se_available = 0.8,
```

```
  coverage_target = 0.95,
  coverage_tolerance = 0.05,
  max_mcse_rmse_ratio = 0.25,
  max_rmse = NULL,
  max_abs_bias = NULL,
  top_n = 6,
  digits = 3,
  ...
)

## S3 method for class 'mfrm_recovery_assessment'
plot(
  x,
  y = NULL,
  type = c("status", "metrics"),
  metric = c("rmse", "bias", "coverage", "se_available", "mcse_rmse"),
  draw = TRUE,
  ...
)
```

Arguments

`x`	For `assess_mfrm_recovery()`, output from `evaluate_mfrm_recovery()`. For `plot.mfrm_recovery_assessment()`, output from `assess_mfrm_recovery()`.
`min_reps`	Minimum replication count expected before treating the simulation as more than a smoke check.
`min_success_rate`	Minimum acceptable proportion of replications that generated data and produced a fitted model.
`min_convergence_rate`	Minimum acceptable proportion of replications whose fitted model reported convergence.
`min_se_available`	Minimum acceptable proportion of recovery rows with standard errors in each parameter group. Set to `NULL` to skip this check.
`coverage_target`	Nominal coverage target, usually `0.95`.
`coverage_tolerance`	Absolute tolerance around `coverage_target`.
`max_mcse_rmse_ratio`	Maximum acceptable Monte Carlo SE of RMSE divided by RMSE. Set to `NULL` to skip this precision check.
`max_rmse`	Optional practical RMSE limit. Use a scalar for all parameter groups or a named vector/list with names such as `"facet"`, `"step"`, `"slope"`, `"Rater"`, or `"facet:Rater:logit"`.
`max_abs_bias`	Optional practical absolute-bias limit. Naming follows `max_rmse`.
`top_n`	Number of next-action lines retained in the compact output.

digits	Digits used by the print method.
...	Reserved for future extensions.
y	Reserved for S3 generic compatibility.
type	Assessment plot route. "status" summarizes checklist status counts; "metrics" plots a parameter-group assessment metric colored by its status.
metric	Metric used when type = "metrics". Supported values are "rmse", "bias", "coverage", "se_available", and "mcse_rmse".
draw	If TRUE, draw with base graphics. If FALSE, return an mfrm_plot_data object with reusable plot tables and metadata.

Details

RMSE and bias adequacy depends on the substantive scale and the use case, so the function does not mark them as failed unless the user supplies max_rmse or max_abs_bias. Without those limits, the corresponding rows are marked not_assessed and the next action asks the user to set practical thresholds when a decision depends on the metric.

The condition_review table is generator metadata for interpreting the recovery run. For bounded GPCM, GPCMSlopeRegime, StressLevel, and generated score-category support describe the data-generating condition; they are not model-fit tests and they are not literature-derived adequacy cut points. condition_reporting_notes turns those generator conditions into reporter-facing caveats, such as high-dispersion slope stress or sparse generated score support.

The optional diagnostic_review table is available when evaluate_mfrm_recovery() was called with include_diagnostics = TRUE. It summarizes fit and separation operating characteristics as diagnostic context only. Its availability fields do not mean that fit or separation values are adequate, and those rows do not enter the recovery adequacy status. diagnostic_reporting_notes should be read first when drafting fit/separation language because it separates zero separation/reliability, absolute fit-ZSTD flags, and df-sensitive ZSTD flags from recovery gates.

plot.mfrm_recovery_assessment() is a user-facing review aid. Use type = "status" first to see where checklist attention is needed, then type = "metrics" to inspect the parameter groups behind RMSE, bias, coverage, standard-error availability, or Monte Carlo precision statuses. The intended reading order is summary(recovery_review), then condition_reporting_notes and condition_review, then diagnostic_reporting_notes and diagnostic_review, then the status plot, then the metric plot, then the row-level recovery table for the parameter groups that need follow-up. When draw = FALSE, the plot data also include reading_order, guidance, condition/diagnostic handoff tables, and user-facing plot tables such as section_status for status plots and metric_review for metric plots.

Value

An object of class mfrm_recovery_assessment with:

- overview: compact run-level status.
- checklist: reviewer-facing adequacy checks.
- condition_review: generator-condition metadata, including bounded GPCM slope-regime interpretation and generated score-category support when available.

- `condition_reporting_notes`: reporter-facing generator-condition caveats separated from recovery metrics and release-gate decisions.
- `diagnostic_reporting_notes`: reporter-facing fit/separation caveats retained as diagnostic context rather than recovery gates.
- `diagnostic_review`: optional fit/separation operating-characteristic context when retained by `evaluate_mfrm_recovery()`.
- `metric_review`: parameter-group metric checks.
- `uncertainty_review`: compact coverage / SE availability interpretation.
- `reading_order`: recommended first-read order for the summary, condition, plot, and row-level recovery outputs.
- `next_actions`: short action list sorted by severity.
- `thresholds`: thresholds used for the assessment.

See Also

`evaluate_mfrm_recovery()`, `plot.mfrm_recovery_simulation()`

Examples

```
rec <- evaluate_mfrm_recovery(
  n_person = 12,
  n_rater = 2,
  n_criterion = 2,
  reps = 1,
  maxit = 30,
  seed = 123
)
assess_mfrm_recovery(rec, min_reps = 1, max_rmse = 1)

# Read the bounded-GPCM generator condition separately from recovery adequacy.
gpcm_spec <- build_mfrm_sim_spec(
  n_person = 14,
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 2,
  model = "GPCM",
  step_facet = "Criterion",
  slope_facet = "Criterion",
  slopes = c(0.85, 1.15),
  assignment = "crossed"
)
gpcm_rec <- suppressWarnings(evaluate_mfrm_recovery(
  sim_spec = gpcm_spec,
  reps = 1,
  fit_method = "MML",
  quad_points = 5,
  maxit = 12,
  include_diagnostics = TRUE,
  include_person = FALSE,
```

```
  seed = 456
))
gpcm_review <- assess_mfrm_recovery(
  gpcm_rec,
  min_reps = 1,
  max_rmse = c(slope = 2),
  max_abs_bias = c(slope = 1),
  min_se_available = NULL,
  max_mcse_rmse_ratio = NULL
)
gpcm_review$condition_reporting_notes[, c(
  "ConditionArea", "ReportingAttention", "ConditionFinding"
)]
gpcm_review$condition_review[, c(
  "Model", "GPCMSlopeRegime", "StressLevel", "ScoreSupportStatus"
)]
gpcm_review$diagnostic_reporting_notes[, c(
  "Facet", "ReportingAttention", "DiagnosticFinding"
)]
summary(gpcm_review)$reading_order
```

`as_flextable` *Generic for converting objects to a* `flextable`

Description

Generic for converting objects to a `flextable`

Usage

```
as_flextable(x, ...)
```

Arguments

`x`	Object to convert.
`...`	Passed to methods.

Value

A `flextable` object (concrete return type from the underlying method, e.g. `[as_flextable.apa_table()]` returns a `flextable` ready for `flextable::save_as_docx()`).

See Also

`as_flextable.apa_table()` for the `apa_table` method; `as_kable()` for a `knitr::kable`-targeted alternative; `apa_table()` for constructing an `apa_table` in the first place.

as_flextable.apa_table *Convert an* apa_table *to a* flextable

Description

Produces a Word / PowerPoint-friendly `flextable` with the caption and note wired in. Requires `flextable` (in Suggests).

Usage

```
## S3 method for class 'apa_table'
as_flextable(x, ...)
```

Arguments

x	An apa_table object from apa_table().
...	Additional arguments reserved for future use.

Value

A `flextable` object, or a message when `flextable` is unavailable.

See Also

as_kable.apa_table(), apa_table().

as_kable *Generic for converting objects to a* knitr::kable

Description

Generic for converting objects to a `knitr::kable`

Usage

```
as_kable(x, ...)
```

Arguments

x	Object to convert.
...	Passed to methods.

Value

A knitr::kable object (concrete return type from the underlying method, e.g. [as_kable.apa_table()] returns a kableExtra object when the package is installed).

See Also

as_kable.apa_table() for the apa_table method; as_flextable() for a flextable-targeted alternative; apa_table() for constructing an apa_table in the first place.

---

as_kable.apa_table *Convert an* apa_table *to a* knitr::kable() *object*

Description

Renders the table data for direct inclusion in RMarkdown, Quarto, or HTML reports, wiring the caption and note slots into the standard APA placement (caption above, note below). When kableExtra is installed the note is attached as a footer; otherwise the note is appended as a knitr::asis_output() block.

Usage

```
## S3 method for class 'apa_table'
as_kable(x, format = c("pipe", "html", "latex"), digits = 3L, ...)
```

Arguments

x	An apa_table object from apa_table().
format	One of "pipe" (default, Markdown), "html", or "latex", passed through to knitr::kable().
digits	Numeric; passed to knitr::kable().
...	Additional arguments forwarded to knitr::kable().

Value

A knitr_kable object ready to be printed inline in a report, or a message when knitr is unavailable.

See Also

as_flextable.apa_table(), apa_table().

`bias_count_table` *Build a bias-cell count report*

Description

Build a bias-cell count report

Usage

```
bias_count_table(
  bias_results,
  min_count_warn = 10,
  branch = c("original", "facets"),
  fit = NULL
)
```

Arguments

`bias_results`	Output from `estimate_bias()`.
`min_count_warn`	Minimum count threshold for flagging sparse bias cells.
`branch`	Output branch: `"facets"` keeps legacy manual-aligned naming, `"original"` returns compact QC-oriented names.
`fit`	Optional `fit_mfrm()` result used to attach run context metadata.

Details

This helper summarizes how many observations contribute to each bias-cell estimate and flags sparse cells.

Branch behavior:

- `"facets"`: keeps legacy manual-aligned column labels (`Sq`, `Observd Count`, `Obs-Exp Average`, `Model S.E.`) for side-by-side comparison with external workflows.
- `"original"`: keeps compact field names (`Count`, `BiasSize`, `SE`) for custom QC workflows and scripting.

Value

A named list with:

- `table`: cell-level counts with low-count flags
- `by_facet`: named list of counts aggregated by each interaction facet
- `by_facet_a`, `by_facet_b`: first two facet summaries (legacy compatibility)
- `summary`: one-row summary
- `thresholds`: applied thresholds
- `branch`, `style`: output branch metadata
- `fit_overview`: optional one-row fit metadata when `fit` is supplied

Interpreting output

- `table`: cell-level contribution counts and low-count flags.
- `by_facet`: sparse-cell structure by each interaction facet.
- `summary`: overall low-count prevalence.
- `fit_overview`: optional run context (when `fit` is supplied).

Low-count cells should be interpreted cautiously because bias-size estimates can become unstable with sparse support.

Typical workflow

1. Estimate bias with `estimate_bias()`.
2. Build `bias_count_table(...)` in desired branch.
3. Review low-count flags before interpreting bias magnitudes.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `table` data.frame contains, in the legacy-compatible branch:

FacetA, FacetB Interaction facet level identifiers; generic names for the two interaction facets.

Sq Sequential row number.

Observd Count Number of observations for this cell.

Obs-Exp Average Observed minus expected average for this cell.

Model S.E. Standard error of the bias estimate.

Infit, Outfit Fit statistics for this cell.

LowCountFlag Logical; `TRUE` when count < `min_count_warn`.

The `summary` data.frame contains:

InteractionFacets Names of the interaction facets.

Cells, TotalCount Number of cells and total observations.

LowCountCells, LowCountPercent Number and share of low-count cells.

See Also

`estimate_bias()`, `unexpected_after_bias_table()`, `build_fixed_reports()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t11 <- bias_count_table(bias)
t11_facets <- bias_count_table(bias, branch = "facets", fit = fit)
summary(t11)
p <- plot(t11, draw = FALSE)
p2 <- plot(t11, type = "lowcount_by_facet", draw = FALSE)
if (interactive()) {
  plot(
    t11,
    type = "cell_counts",
    draw = TRUE,
    main = "Bias Cell Counts (Customized)",
    palette = c(count = "#2b8cbe", low = "#cb181d"),
    label_angle = 45
  )
}
```

---

bias_interaction_report

Build a bias-interaction plot-data bundle (FACETS Table 13: ranked bias list)

Description

Bundles the **ranked flagged-cells** view of a bias-interaction run for downstream printing and plotting. The three sibling reports in this family are intentionally distinct:

- `bias_interaction_report()` (this one) = FACETS Table 13: a ranked list of interaction cells with `t`, `bias size`, and screening tail area – use when reviewing which `(facet_a, facet_b)` cells deserve follow-up.
- `bias_iteration_report()` = iteration history / convergence trace for the bias recalibration (FACETS Table 9 territory) – use when diagnosing whether the bias run itself stabilised.
- `bias_pairwise_report()` = pairwise contrast table for a target facet (FACETS Table 14 territory) – use when comparing levels within a facet while controlling for the other.

Usage

```
bias_interaction_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
```

```
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  top_n = 50,
  abs_t_warn = 2,
  abs_bias_warn = 0.5,
  p_max = 0.05,
  sort_by = c("abs_t", "abs_bias", "prob")
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
max_abs	Bound for absolute bias size when estimating from fit.
omit_extreme	Omit extreme-only elements when estimating from fit.
max_iter	Iteration cap for bias estimation when x is fit.
tol	Convergence tolerance for bias estimation when x is fit.
top_n	Maximum number of ranked rows to keep.
abs_t_warn	Warning cutoff for absolute t statistics.
abs_bias_warn	Warning cutoff for absolute bias size.
p_max	Warning cutoff for p-values.
sort_by	Ranking key: "abs_t", "abs_bias", or "prob".

Details

Preferred bundle API for interaction-bias diagnostics. The function can:

- use a precomputed bias object from estimate_bias(), or
- estimate internally from mfrm_fit + facet specification.

Value

A named list with bias-interaction plotting/report components. Class: mfrm_bias_interaction.

Interpreting output

Focus on ranked rows where multiple screening criteria converge:

- large absolute t statistic
- large absolute bias size
- small screening tail area

The bundle is optimized for downstream `summary()` and `plot_bias_interaction()` views.

Typical workflow

1. Run `estimate_bias()` (or provide `mfrm_fit` here).
2. Build `bias_interaction_report(...)`.
3. Review `summary(out)` and visualize with `plot_bias_interaction()`.

See Also

`estimate_bias()`, `build_fixed_reports()`, `plot_bias_interaction()`

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
out <- bias_interaction_report(bias, top_n = 10)
summary(out)
p_bi <- plot(out, draw = FALSE)
p_bi$data$plot
```

---

bias_iteration_report *Build a bias-iteration report (FACETS Table 9: iteration / convergence trace)*

Description

This report is NOT an alias of `bias_interaction_report()` despite the similar name. It focuses on the **recalibration path** of a bias run: iteration table, convergence summary, and orientation review. Use this to confirm that the bias recalibration itself converged; use `bias_interaction_report()` to review the ranked flagged cells from the converged run.

Usage

```
bias_iteration_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  top_n = 10
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
max_abs	Bound for absolute bias size when estimating from fit.
omit_extreme	Omit extreme-only elements when estimating from fit.
max_iter	Iteration cap for bias estimation when x is fit.
tol	Convergence tolerance for bias estimation when x is fit.
top_n	Maximum number of iteration rows to keep in preview-oriented summaries. The full iteration table is always returned.

Details

This report focuses on the recalibration path used by estimate_bias(). It provides a package-native counterpart to legacy iteration printouts by exposing the iteration table, convergence summary, and orientation review in one bundle.

Value

A named list with:

- table: iteration history
- summary: one-row convergence summary
- orientation_review: interaction-facet sign review
- settings: resolved reporting options

- `direction_note`: one-line interpretive note describing which direction the iteration moved (carried from the bias estimator; empty string when the underlying estimator does not emit one)
- `recommended_action`: one-line recommended action label (e.g. `"converged"`, `"increase max_iter"`); empty string when the underlying estimator does not emit one

See Also

`estimate_bias()`, `bias_interaction_report()`, `build_fixed_reports()`

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
out <- bias_iteration_report(fit, diagnostics = diag, facet_a = "Rater", facet_b = "Criterion")
summary(out)
```

---

bias_pairwise_report *Build a bias pairwise-contrast report (FACETS Table 14: pairwise contrasts)*

Description

Build a pairwise contrast table that, for a chosen target facet (e.g. raters), compares each pair of target-facet levels while holding a context facet (e.g. items / criteria) constant. This is the FACETS Table 14 view: it answers "is rater A consistently more severe than rater B on the same items?" rather than "which (rater, item) cell has the largest local bias?" – the latter is covered by `bias_interaction_report()`.

Usage

```
bias_pairwise_report(
  x,
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001,
  target_facet = NULL,
  context_facet = NULL,
  top_n = 50,
  p_max = 0.05,
  sort_by = c("abs_t", "abs_contrast", "prob")
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
max_abs	Bound for absolute bias size when estimating from fit.
omit_extreme	Omit extreme-only elements when estimating from fit.
max_iter	Iteration cap for bias estimation when x is fit.
tol	Convergence tolerance for bias estimation when x is fit.
target_facet	Facet whose local contrasts should be compared across the paired context facet. Defaults to the first interaction facet.
context_facet	Optional facet to condition on. Defaults to the other facet in a 2-way interaction.
top_n	Maximum number of ranked rows to keep.
p_max	Flagging cutoff for pairwise p-values.
sort_by	Ranking key: "abs_t", "abs_bias", or "prob".

Details

This helper exposes the pairwise contrast table that was previously only reachable through fixed-width output generation. It is available only for 2-way interactions. The pairwise contrast statistic uses a Welch/Satterthwaite approximation and is labeled as a Rasch-Welch comparison in the output metadata.

Value

A named list with:

- table: pairwise contrast rows
- summary: one-row contrast summary
- orientation_review: interaction-facet sign review
- settings: resolved reporting options
- direction_note: one-line interpretive note describing the dominant pairwise-contrast direction (carried from the underlying bias estimator; empty string when not applicable)
- recommended_action: one-line recommended-action label (e.g. routing the user to follow-up review of the largest flagged pairs); empty string when the underlying estimator does not emit one

Interpreting output

- `table`: one row per ordered (target_level_1, target_level_2) pair, with `Bias_diff`, `SE_diff`, `t_diff`, `df_diff`, `p_diff`, and the underlying per-level bias rows. Rows are sorted so that the largest-magnitude `|t_diff|` rises to the top.
- `summary`: one-row screening summary with `MaxAbsBiasDiff`, `MaxAbsT`, `Significant` (count of flagged pairs at `p_max`), `BonferroniSignificant`, and `HolmSignificant`.
- `orientation_review` carries the same facet-orientation sign review as the parent `estimate_bias()` run.
- The SE caveat below applies: read `Significant` / `BonferroniSignificant` as a screening triage, not as formal inferential tests.

Typical workflow

1. Fit and diagnose the model.
2. Run `estimate_bias()` to get the underlying interaction effects.
3. Pass that result to `bias_pairwise_report()` for the rater-pair contrast table.
4. Use `summary(out)$MaxAbsT` and the top rows of `out$table` to flag rater-pair systematic differences for follow-up review.
5. For the ranked flagged-cells view (which (rater, item) pairs have the largest local bias), use `bias_interaction_report()` on the same `estimate_bias()` output.

Standard-error caveat

The contrast standard error is computed as `SE(b_i - b_j) = sqrt(SE_i^2 + SE_j^2)` – the independence approximation. For same-facet bias values that share a sum-to-zero identification, `Cov(b_i, b_j) < 0`, so the true contrast variance is `SE_i^2 + SE_j^2 - 2 * Cov(b_i, b_j)`, which is **smaller** than the reported value. The reported t-statistics and p-values are therefore conservative for same-facet contrasts (the true significance is higher than reported). For across-facet contrasts the covariance term is approximately zero and the approximation is appropriate. Use the report as a screening / triage table; for inferential claims that hinge on a marginally-significant same-facet contrast, follow up with a contrast that uses the full parameter covariance.

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press.
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly, 2*(3), 197-221.
- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement, 4*(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement, 5*(2), 189-227.

See Also

`estimate_bias()`, `bias_interaction_report()`, `build_fixed_reports()`

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
out <- bias_pairwise_report(fit, diagnostics = diag, facet_a = "Rater", facet_b = "Criterion")
s <- summary(out)
s$summary
# Look for: `MaxAbsBiasDiff` < ~0.5 logits and `Significant = 0` mean
#   no rater pair contrasts above the screen. The `BonferroniSignificant`
#   / `HolmSignificant` columns count pairs that survive multiple-
#   testing correction; both being 0 is a stronger "no rater-pair
#   inconsistency" signal than the raw screen-positive count alone.
head(out$table)
# Look for: top rows with `|t_diff|` > 2 and |Bias_diff| > 0.5 logits
#   warrant content-review of the two raters' scoring conventions on
#   the conditioning context facet (e.g. compare their item-level
#   marks for systematic strictness/leniency patterns).
```

`build_apa_outputs` *Build APA text outputs from model results*

Description

Build APA text outputs from model results

Usage

```
build_apa_outputs(
  fit,
  diagnostics,
  bias_results = NULL,
  context = list(),
  whexact = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`.
`bias_results`	Optional output from `estimate_bias()`.
`context`	Optional named list for report context.
`whexact`	Use exact ZSTD transformation.

Details

`context` is an optional named list for narrative customization. Frequently used fields include:

- `assessment`, `setting`, `scale_desc`
- `rater_training`, `raters_per_response`
- `rater_facet` (used for targeted reliability note text)
- `line_width` (optional text wrapping width for `report_text`; default = 92)

Output text includes residual-PCA screening commentary if PCA diagnostics are available in `diagnostics`.

For bounded `GPCM`, this helper returns a caveated partial reporting bundle over supported diagnostics, direct tables, and plots. It also includes a `gpcm_boundary` table. Treat the output as slope-aware sensitivity-reporting text, not FACETS score-side equivalence, automatic operational scoring, or design-forecasting evidence.

By default, `report_text` includes:

- model/data design summary (N, facet counts, scale range)
- optimization/convergence metrics (`Converged`, `Iterations`, `LogLik`, `AIC`, `BIC`)
- anchor/constraint summary (`noncenter_facet`, anchored levels, group anchors, dummy facets)
- latent-regression population-model wording when `fit` has an active `population_formula`
- category/threshold diagnostics (including disordered-step details when present)
- overall fit, misfit count, and top misfit levels
- facet reliability/separation, residual PCA summary, and bias-screen counts

Value

An object of class `mfrm_apa_outputs` with:

- `report_text`: APA-style Method/Results draft prose
- `table_figure_notes`: consolidated draft notes for tables/visuals
- `table_figure_captions`: draft caption candidates without figure numbering
- `section_map`: package-native section table for manuscript assembly
- `contract`: structured APA reporting contract used for downstream checks

Interpreting output

- `report_text`: manuscript-draft narrative covering Method (model specification, estimation, convergence) and Results (global fit, facet separation/reliability, misfit triage, category diagnostics, residual-PCA screening, bias screening). Written in third-person past tense following APA 7th edition conventions, but still intended for human review.
- `table_figure_notes`: reusable draft note blocks for table/figure appendices.
- `table_figure_captions`: draft caption candidates aligned to generated outputs.
- active latent-regression fits add a population-model section and Table 5 notes/captions that distinguish conditional-normal coefficient reporting from post hoc regression on EAP/MLE scores.

When bias results or PCA diagnostics are not supplied, those sections are omitted from the narrative rather than producing placeholder text.

Typical workflow

1. Build diagnostics (and optional bias results). For RSM / PCM reporting runs, prefer an MML fit and `diagnose_mfrm(..., diagnostic_mode = "both")`.
2. Run `build_apa_outputs(...)`.
3. Check `summary(apa)` for completeness.
4. Insert `apa$report_text` and note/caption fields into manuscript drafts after checking the listed cautions.

Context template

A minimal `context` list can include fields such as:

- `assessment`: name of the assessment task
- `setting`: administration context
- `scale_desc`: short description of the score scale
- `rater_facet`: rater facet label used in narrative reliability text

Input validation

`fit` must be an `mfrm_fit` object from `fit_mfrm()`. `diagnostics` must be an `mfrm_diagnostics` object from `diagnose_mfrm()`. `context` must be a list (use `NULL` or `list()` for no extra context). If supplied, `bias_results` must come from `estimate_bias()` or another package-native bias helper that provides a table component.

See Also

`build_visual_summaries()`, `estimate_bias()`, `reporting_checklist()`, mfrmr_reporting_and_apa

Examples

```
# Fast smoke run: a JML fit and a legacy diagnostic let us build the
# APA bundle and confirm `report_text` is non-empty in well under
# a second.
toy <- load_mfrmr_data("example_core")
fit_quick <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                      method = "JML", maxit = 30)
diag_quick <- diagnose_mfrm(fit_quick, residual_pca = "none",
                            diagnostic_mode = "legacy")
apa_quick <- build_apa_outputs(fit_quick, diag_quick)
nchar(apa_quick$report_text) > 0


fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
apa <- build_apa_outputs(
  fit,
  diag,
  context = list(
```

```
    assessment = "Toy writing task",
    setting = "Demonstration dataset",
    scale_desc = "0-2 rating scale",
    rater_facet = "Rater"
  )
)
s_apa <- summary(apa)
s_apa$overview
# Look for: `SentenceCount` non-zero in every section that the run
#   should support (Method / Results / fit / reliability / bias).
#   Zero counts mean that section's prose is empty and the
#   manuscript will need to fill it manually.
chk <- reporting_checklist(fit, diagnostics = diag)
head(chk$checklist[, c("Section", "Item", "DraftReady", "NextAction")])
# Look for: rows with `DraftReady = "yes"` are ready to paste into
#   the manuscript. `"no"` rows tell you which helper / setting
#   needs to run before that paragraph can be drafted, via
#   `NextAction`. Aim for every Visual Displays / Reliability /
#   Diagnostics row to be `"yes"` before submitting.
cat(apa$report_text)
apa$section_map[, c("SectionId", "Available")]
```

`build_conquest_overlap_bundle` *Build a scoped ConQuest-overlap bundle*

Description

Build a scoped ConQuest-overlap bundle

Usage

```
build_conquest_overlap_bundle(
  fit = NULL,
  case = c("synthetic_latent_regression"),
  output_dir = NULL,
  prefix = "conquest_overlap",
  overwrite = FALSE,
  quad_points = 7L,
  maxit = 40L,
  reltol = 1e-06
)
```

Arguments

`fit`	Optional output from `fit_mfrm()` or `run_mfrm_facets()`. When omitted, the helper builds the package's `"synthetic_latent_regression"` overlap case.

case	Overlap case used when fit = NULL. Currently only "synthetic_latent_regression" is supported.
output_dir	Optional directory where the bundle files should be written. When NULL, the helper returns the in-memory bundle only.
prefix	File-name prefix used when writing the bundle to disk.
overwrite	If FALSE, refuse to overwrite existing files.
quad_points	Quadrature points used when fit = NULL and the overlap case is fit on the fly.
maxit	Maximum optimizer iterations used when fit = NULL.
reltol	Relative convergence tolerance used when fit = NULL.

Details

This helper prepares a narrow ConQuest comparison bundle for an RSM / PCM latent-regression MML fit and records the mfrmr-side tables to compare after an external ConQuest run. The supported overlap is intentionally narrow:

- ordered-response RSM / PCM only;
- binary responses only;
- exactly one non-person facet, treated as the item facet;
- active latent-regression MML;
- exactly one numeric person covariate beyond the intercept;
- complete person-by-item rectangular data.

The returned bundle standardizes the responses to {0, 1}, pivots them to a one-row-per-person wide CSV, stores the corresponding person covariates, and records the mfrmr estimates that should be compared externally.

The conquest_command component is a conservative starting template, not a guaranteed version-invariant automation. The conquest_output_contract component records which requested external output should feed each normalized review table. Use normalize_conquest_overlap_files() or normalize_conquest_overlap_tables() and then review_conquest_overlap() only after the matching ConQuest run has been executed externally and the relevant output tables have been extracted. The bundle and command template alone are not external validation evidence.

Value

A named list with class mfrm_conquest_overlap_bundle.

Comparison targets

- regression slope: compare directly;
- residual variance sigma2: compare directly;
- item estimates: compare after centering because the Rasch location origin remains constraint-dependent;
- case EAP estimates: compare as posterior summaries under the fitted population model.

Output

The returned object has class `mfrm_conquest_overlap_bundle` and includes:

- `summary`: one-row scope summary with posterior-basis and population-model review fields
- `comparison_targets`: comparison rules for the exported tables
- `conquest_output_contract`: requested ConQuest outputs and review handoff
- `response_long`: long-format binary response data used by the bundle
- `response_wide`: wide CSV-ready response matrix for the ConQuest template
- `person_data`: one-row-per-person covariate table
- `item_map`: mapping from exported response columns to original item levels
- `mfrmr_population`: fitted population-model coefficients plus `sigma2`
- `mfrmr_item_estimates`: fitted item estimates with centered values
- `mfrmr_case_eap`: posterior EAP summaries for the fitted persons
- `conquest_command`: conservative ConQuest command template
- `written_files`: file inventory when `output_dir` is supplied
- `settings`: bundle settings
- `notes`: interpretation notes

See Also

`normalize_conquest_overlap_files()`, `normalize_conquest_overlap_tables()`, `review_conquest_overlap()`, `reference_case_benchmark()`, `build_mfrm_replay_script()`, `export_mfrm_bundle()`

Examples

```
bundle <- build_conquest_overlap_bundle(quad_points = 3, maxit = 30)
bundle$summary[, c("Case", "Facet", "Covariate", "Persons", "Items")]
summary(bundle)$conquest_command_scope
summary(bundle)$conquest_output_contract
cat(substr(bundle$conquest_command, 1, 120))
```

`build_equating_chain` *Build a screened linking chain across ordered calibrations*

Description

Links a series of calibration waves by computing mean offsets between adjacent pairs of fits. Common linking elements (e.g., raters or items that appear in consecutive administrations) are used to estimate the scale shift. Cumulative offsets place all waves on a common metric anchored to the first wave. The procedure is intended as a practical screened linking aid, not as a full general-purpose equating framework.

Usage

```
build_equating_chain(
  fits,
  anchor_facets = NULL,
  include_person = FALSE,
  drift_threshold = 0.5
)

## S3 method for class 'mfrm_equating_chain'
print(x, ...)

## S3 method for class 'mfrm_equating_chain'
plot(
  x,
  y = NULL,
  type = c("common_anchors", "graph", "chain"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ...
)

## S3 method for class 'mfrm_equating_chain'
summary(object, ...)

## S3 method for class 'summary.mfrm_equating_chain'
print(x, ...)
```

Arguments

fits	Named list of mfrm_fit objects in chain order.
anchor_facets	Character vector of facets to use as linking elements.
include_person	Include person estimates in linking.
drift_threshold	Threshold for flagging large residuals in links.
x	An mfrm_equating_chain object.
...	Ignored.
y	Unused (S3 plot signature requirement).
type	One of "graph" (bipartite Wave x anchor-element graph; requires the igraph package), "common_anchors" (default; bar chart of common-anchor counts per wave pair), or "chain".
preset	Visual preset.
draw	If TRUE, draw the plot with base graphics.
object	An mfrm_equating_chain object (for summary).

Details

The screened linking chain uses a screened link-offset method. For each pair of adjacent waves (A, B), the function:

1. Identifies common linking elements (facet levels present in both fits).
2. Computes per-element differences:

$$d_e = \hat{\delta}_{e,B} - \hat{\delta}_{e,A}$$

3. Computes a preliminary link offset using the inverse-variance weighted mean of these differences when standard errors are available (otherwise an unweighted mean).
4. Screens out elements whose residual from that preliminary offset exceeds `drift_threshold`, then recomputes the final offset on the retained set.
5. Records `Offset_SD` (standard deviation of retained residuals) and `Max_Residual` (maximum absolute deviation from the mean) as indicators of link quality.
6. Flags links with fewer than 5 retained common elements in any linking facet as having thin support.

Cumulative offsets are computed by chaining link offsets from Wave 1 forward, placing all waves onto the metric of the first wave.

Elements whose per-link residual exceeds `drift_threshold` are flagged in `$element_detail$Flag`. A high `Offset_SD`, many flagged elements, or a thin retained anchor set signals an unstable link that may compromise the resulting scale placement.

Value

Object of class `mfrm_equating_chain` with components:

links Tibble of link-level statistics (offset, SD, etc.).

cumulative Tibble of cumulative offsets per wave.

element_detail Tibble of element-level linking details.

common_by_facet Tibble of retained common-element counts by facet.

config List of analysis configuration.

Which function should I use?

- Use `anchor_to_baseline()` for a single new wave anchored to a known baseline.
- Use `detect_anchor_drift()` when you want direct comparison against one reference wave.
- Use `build_equating_chain()` when no single wave should dominate and you want ordered, adjacent links across the series.

Interpreting output

- `$links`: one row per adjacent pair with `From`, `To`, `N_Common`, `N_Retained`, `Offset_Prelim`, `Offset`, `Offset_SD`, and `Max_Residual`. Small `Offset_SD` relative to the offset indicates a consistent shift across elements. `LinkSupportAdequate = FALSE` means at least one linking facet retained fewer than 5 common elements after screening.
- `$cumulative`: one row per wave with its cumulative offset from Wave 1. Wave 1 always has offset 0.
- `$element_detail`: per-element linking statistics (estimate in each wave, difference, residual from mean offset, and flag status). Flagged elements may indicate DIF or rater re-training effects.
- `$common_by_facet`: retained common-element counts by linking facet for each adjacent link.
- `$config`: records wave names and analysis parameters.
- Read `links` before `cumulative`: weak adjacent links can make later cumulative offsets less trustworthy.

Typical workflow

1. Fit each administration wave separately: `fit_a <- fit_mfrm(...)`.
2. Combine into an ordered named list: `fits <- list(Spring23 = fit_s, Fall23 = fit_f, Spring24 = fit_s2)`.
3. Call `chain <- build_equating_chain(fits)`.
4. Review `summary(chain)` for link quality.
5. Visualize with `plot_anchor_drift(chain, type = "chain")`.
6. For problematic links, investigate flagged elements in `chain$element_detail` and consider removing them from the anchor set.

See Also

`detect_anchor_drift()`, `anchor_to_baseline()`, `make_anchor_table()`, `plot_anchor_drift()`

Examples

```
toy <- load_mfrmr_data("example_core")
people <- unique(toy$Person)
d1 <- toy[toy$Person %in% people[1:12], , drop = FALSE]
d2 <- toy[toy$Person %in% people[13:24], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
chain <- build_equating_chain(list(Form1 = fit1, Form2 = fit2))
summary(chain)
chain$cumulative
```

`build_fixed_reports` *Build legacy-compatible fixed-width text reports*

Description

Build legacy-compatible fixed-width text reports

Usage

```
build_fixed_reports(
  bias_results,
  target_facet = NULL,
  branch = c("facets", "original")
)
```

Arguments

`bias_results`	Output from `estimate_bias()`.
`target_facet`	Optional target facet for pairwise contrast table.
`branch`	Output branch: `"facets"` keeps the legacy-compatible fixed-width layout; `"original"` returns compact sectioned fixed-width text for report drafts.

Details

This function generates plain-text, fixed-width output intended to be read in console/log environments or exported into text reports.

The pairwise section (Table 14 style) is only generated for 2-way bias runs. For higher-order interactions (`interaction_facets` length >= 3), the function returns the bias table text and a note explaining why pairwise contrasts were skipped.

Value

A named list with class `mfrm_fixed_reports` (and a branch-specific subclass `mfrm_fixed_reports_<branch>`):

- `bias_fixed`: fixed-width interaction table text
- `pairwise_fixed`: fixed-width pairwise contrast text
- `pairwise_table`: underlying pairwise data.frame
- `branch`: character scalar `"original"` or `"facets"` echoing which fixed-width style was rendered
- `style`: character scalar carrying the resolved style preset used when building the text artifact
- `interaction_label`: human-readable label for the interaction that drove the bias run (`"Rater x Criterion"`-style); `NA` when no bias rows are available
- `target_facet`: character scalar identifying which facet was used as the target facet for pairwise contrasts; `NA` when no pairwise contrasts were requested or available

Interpreting output

- `bias_fixed`: fixed-width table of interaction effects.
- `pairwise_fixed`: pairwise contrast text (2-way only).
- `pairwise_table`: structured contrast table.
- `interaction_label`: facets used for the bias run.

Typical workflow

1. Run `estimate_bias()`.
2. Build text bundle with `build_fixed_reports(...)`.
3. Use `summary()`/`plot()` for quick checks, then export text blocks.

Preferred route for new analyses

For new reporting workflows, prefer `bias_interaction_report()` and `build_apa_outputs()`. Use `build_fixed_reports()` when a fixed-width text artifact is specifically required for a compatibility handoff.

See Also

`estimate_bias()`, `build_apa_outputs()`, `bias_interaction_report()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
fixed <- build_fixed_reports(bias)
fixed_original <- build_fixed_reports(bias, branch = "original")
summary(fixed)
p <- plot(fixed, draw = FALSE)
p2 <- plot(fixed, type = "pvalue", draw = FALSE)
if (interactive()) {
  plot(
    fixed,
    type = "contrast",
    draw = TRUE,
    main = "Pairwise Contrasts (Customized)",
    palette = c(pos = "#1b9e77", neg = "#d95f02"),
    label_angle = 45
  )
}
```

`build_linking_review` *Build a linking-review synthesis object*

Description

Build a linking-review synthesis object

Usage

```
build_linking_review(
  anchor_review = NULL,
  drift = NULL,
  chain = NULL,
  top_n = 10
)
```

Arguments

`anchor_review`	Optional output from `review_mfrm_anchors()`.
`drift`	Optional output from `detect_anchor_drift()`.
`chain`	Optional output from `build_equating_chain()`.
`top_n`	Maximum number of linking-risk rows to highlight in summary outputs. The full object keeps the full risk tables.

Details

`build_linking_review()` does not recompute anchor, drift, or chain statistics. It is a synthesis layer that organizes package-native evidence into one operational review surface with:

- a front-door status block,
- ranked linking risks,
- explicit next actions,
- plot routing metadata,
- a reporting/export handoff map.

The helper keeps the current conservative interpretation policy: anchor drift and screened links are operational review tools, not automatic proofs of scale equivalence or score comparability.

Value

An object of class `mfrm_linking_review`.

Recommended input route

Use existing package-native outputs in this order:

1. `review_mfrm_anchors()` for pre-fit anchor adequacy.
2. `detect_anchor_drift()` for direct wave-to-reference drift screening.
3. `build_equating_chain()` for adjacent screened-link review across waves.

Interpreting output

- `overview`: which evidence sources were supplied and the current review status.
- `top_linking_risks`: primary operational triage table.
- `group_view_index`: stable wave/link/facet/source-family grouping routes.
- `plot_map`: which existing plotting helper should be used next.
- `reporting_map`: what is covered here versus which manuscript-oriented helper should be used separately.

GPCM boundary

This helper is currently intended for the validated `RSM` / `PCM` linking workflow. If the supplied drift/chain sources resolve to bounded `GPCM`, the helper stops with a package-level message rather than silently implying support.

See Also

`review_mfrm_anchors()`, `detect_anchor_drift()`, `build_equating_chain()`, `plot_anchor_drift()`, mfrmr_linking_and_dff

Examples

```
d1 <- load_mfrmr_data("study1")
d2 <- load_mfrmr_data("study2")
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
anchor_review_obj <- review_mfrm_anchors(d1, "Person", c("Rater", "Criterion"), "Score")
drift <- detect_anchor_drift(list(Wave1 = fit1, Wave2 = fit2))
chain <- build_equating_chain(list(Wave1 = fit1, Wave2 = fit2))
review <- build_linking_review(anchor_review = anchor_review_obj, drift = drift, chain = chain)
summary(review)
review$top_linking_risks
review$group_view_index
```

`build_mfrm_manifest` *Build a reproducibility manifest for an MFRM analysis*

Description

Build a reproducibility manifest for an MFRM analysis

Usage

```
build_mfrm_manifest(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  include_person_anchors = FALSE,
  data = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()` or `run_mfrm_facets()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. When `NULL`, diagnostics are computed with `residual_pca = "none"`.
`bias_results`	Optional output from `estimate_bias()` or a named list of bias bundles.
`population_prediction`	Optional output from `predict_mfrm_population()`.
`unit_prediction`	Optional output from `predict_mfrm_units()`.
`plausible_values`	Optional output from `sample_mfrm_plausible_values()`.
`include_person_anchors`	If `TRUE`, include person measures in the exported anchor table.
`data`	Optional original analysis data frame. When supplied, the manifest's `input_hash` row for `data` is computed against the user's untouched input rather than the package's internal `prep$data` (which carries synthesised `Weight` / `score_k` columns) so the recorded fingerprint matches what `read.csv()` will produce in a replay session.

Details

This helper captures the package-native equivalent of the Streamlit app's configuration export. It summarizes analysis settings, source columns, anchoring information, and which downstream outputs are currently available.

Value

A named list with class `mfrm_manifest`.

When to use this

Use `build_mfrm_manifest()` when you want a compact, machine-readable record of how an analysis was run. Compared with related helpers:

- `export_mfrm()` writes analysis tables only.
- `build_mfrm_manifest()` records settings and available outputs.
- `build_mfrm_replay_script()` creates an executable R script.
- `export_mfrm_bundle()` writes a shareable folder of files.

Output

The returned bundle has class `mfrm_manifest` and includes:

- `summary`: one-row analysis overview
- `environment`: package/R/platform metadata
- `model_settings`: key-value model settings table
- `source_columns`: key-value data-column table
- `estimation_control`: key-value optimizer settings table
- `anchor_summary`: facet-level anchor summary
- `anchors`: machine-readable anchor table
- `hierarchical_review`: retained traceability table for hierarchical / small-sample design flags
- `missing_recoding`: retained traceability table for missing-code recoding
- `shrinkage_review`: retained traceability table for shrinkage settings
- `available_outputs`: availability table for diagnostics/bias/PCA/prediction outputs
- `dependencies`, `input_hash`, and `session_info`: reproducibility metadata tables
- `settings`: manifest build settings

Interpreting output

The `summary` table is the direct place to confirm that you are looking at the intended analysis. The `model_settings`, `source_columns`, and `estimation_control` tables are designed for reproducibility records and method write-up. Active latent-regression fits also record their population-model provenance there, including the fitted scoring basis, stored `population_formula`, and person-level contract used by the fitted population model. When categorical background variables are expanded through `stats::model.matrix()`, `population_xlevel_variables` and `population_contrast_variables` identify the variables whose fitted coding must be preserved for replay/scoring. The `available_outputs` table is especially useful before building bundles, because it tells you whether residual PCA, anchors, bias results, or prediction-side artifacts are already available. A practical reading order is `summary` first, `available_outputs` second, and `anchors` last when reproducibility depends on fixed constraints.

Typical workflow

1. Fit a model with `fit_mfrm()` or `run_mfrm_facets()`.
2. Compute diagnostics once with `diagnose_mfrm()` if you want explicit control over residual PCA.
3. Build a manifest and inspect `summary` plus `available_outputs`.
4. If you need files on disk, pass the same objects to `export_mfrm_bundle()`.

For bounded `GPCM` fits, the manifest is available with an explicit `gpcm_boundary` table. It records supported direct diagnostics/reporting surfaces while keeping full FACETS score-side contract review blocked and routing design forecasting through its separate caveated capability row.

See Also

`export_mfrm_bundle()`, `build_mfrm_replay_script()`, `make_anchor_table()`, `reporting_checklist()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
manifest <- build_mfrm_manifest(fit, diagnostics = diag)
manifest$summary[, c("Model", "Method", "Observations", "Facets")]
manifest$available_outputs[, c("Component", "Available")]
```

---

build_mfrm_network_review

Build an MFRM network review

Description

Build an MFRM network review

Usage

```
build_mfrm_network_review(
  fit,
  diagnostics = NULL,
  sparse_design = NULL,
  peer_review_design = NULL,
  top_n_subsets = NULL,
  min_observations = 0,
  top_n = 10,
  include_graph = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
sparse_design	Optional sparse-design metadata. Supply either the generated data frame that carries the mfrm_sparse_design attribute, the attribute itself, or a data frame with sparse design columns such as SparseDesignActive, DesignDensity, MinCommonPersonsPerRaterPair, and ZeroCommonRaterPairs.
peer_review_design	Optional peer-review design metadata. Supply either the generated data frame that carries the mfrm_peer_review_design attribute, the attribute itself, or its overview data frame.
top_n_subsets	Optional maximum number of connected-subset rows to retain before constructing the graph; passed to mfrm_network_analysis().
min_observations	Minimum observations required to keep a subset row; passed to mfrm_network_analysis().
top_n	Number of central/cut/bridge rows to retain in the review.
include_graph	Logical; if TRUE, keep the underlying igraph object in the nested source_network bundle.

Details

build_mfrm_network_review() is a synthesis layer over mfrm_network_analysis(). It keeps the measurement model and graph view in separate lanes: MFRM estimates remain the measurement results, while the network review summarizes co-observation connectedness and linking vulnerability in the observed design. This is especially useful for sparse or incomplete rater-mediated designs, where common-person links, connected subsets, articulation points, and bridge edges can explain why an otherwise estimable model depends on fragile design links.

The review status is deliberately conservative and descriptive. It is not a literature-derived adequacy cut point for fit, separation, recovery, or rater quality. Use it to decide which design links, anchors, or additional observations need inspection before making common-scale claims.

Value

A bundle of class mfrm_network_review containing:

- overview: connectedness and front-door review status
- network_summary: graph-level metrics from mfrm_network_analysis()
- facet_summary: facet-level vulnerability summaries
- top_central_nodes, top_cut_nodes, top_bridge_edges: follow-up rows
- sparse_review: optional sparse-design linking review
- peer_review: optional peer-review assignment and linkage diagnostics
- reporting_map: boundary between MFRM, design network, sparse design, peer-review design, and rater-effect network routes

References

- Wind, S. A., & Jones, E. (2018). The stabilizing influences of linking set size and model-data fit in sparse rater-mediated assessment networks. *Educational and Psychological Measurement*. doi:10.1177/0013164417703733.
- Wind, S. A., Jones, E., & Grajeda, S. (2023). Does sparseness matter? Examining the use of generalizability theory and many-facet Rasch measurement in sparse rating designs. *Applied Psychological Measurement*, 47(5-6), 351-364. doi:10.1177/01466216231182148.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

See Also

`mfrm_network_analysis()`, `subset_connectivity_report()`, `build_summary_table_bundle()`, `rater_network_analysis()`, `rater_halo_network_analysis()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("igraph", quietly = TRUE)) {
  review <- build_mfrm_network_review(fit)
  summary(review)
  build_summary_table_bundle(review)
}
```

---

build_mfrm_replay_script

Build a package-native replay script for an MFRM analysis

Description

Build a package-native replay script for an MFRM analysis

Usage

```
build_mfrm_replay_script(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  data_file = "your_data.csv",
  fit_person_data_file = NULL,
  script_mode = c("auto", "fit", "facets"),
```

```
  include_bundle = FALSE,
  bundle_dir = "analysis_bundle",
  bundle_prefix = "mfrmr_replay"
)
```

Arguments

fit	Output from fit_mfrm() or run_mfrm_facets().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are reused from run_mfrm_facets() when available, otherwise recomputed.
bias_results	Optional output from estimate_bias() or a named list of bias bundles. When supplied, the generated script includes package-native bias estimation calls.
population_prediction	Optional output from predict_mfrm_population() to recreate in the generated script.
unit_prediction	Optional output from predict_mfrm_units() to recreate in the generated script.
plausible_values	Optional output from sample_mfrm_plausible_values() to recreate in the generated script.
data_file	Path to the analysis data file used in the generated script.
fit_person_data_file	Optional CSV filename to read for the fit-level latent-regression replay person table. When NULL, the replay script embeds that table inline. export_mfrm_bundle() uses this to keep replay scripts portable while avoiding large inline literals.
script_mode	One of "auto", "fit", or "facets". "auto" uses run_mfrm_facets() when the input object came from that workflow.
include_bundle	If TRUE, append an export_mfrm_bundle() call to the generated script.
bundle_dir	Output directory used when include_bundle = TRUE.
bundle_prefix	Prefix used by the generated bundle exporter call.

Details

This helper mirrors the Streamlit app's reproducible-download idea, but uses mfrmr's installed API rather than embedding a separate estimation engine. The generated script assumes the user has the package installed and provides a data file at data_file.

Anchor and group-anchor constraints are embedded directly from the fitted object's stored configuration, so the script can replay anchored analyses without manual table reconstruction.

When the supplied fit uses the latent-regression MML branch, the generated fit-mode script also carries the stored replay-ready person table together with the corresponding population_formula / person_id / population_policy arguments needed to recreate the population model. By default that replay-ready table is embedded inline; when fit_person_data_file is supplied, the generated script reads it from that sidecar CSV relative to the replay script location.

For bounded GPCM, replay scripts are available with an explicit gpcm_boundary table. The generated script records step_facet and slope_facet settings, but full FACETS score-side contract review remains outside this replay contract. Role-based design forecasting is available through predict_mfrm_population() as a separate caveated helper route.

Value

A named list with class `mfrm_replay_script`.

When to use this

Use `build_mfrm_replay_script()` when you want a package-native recipe that another analyst can rerun later. Compared with related helpers:

- `build_mfrm_manifest()` records settings but does not run anything.
- `build_mfrm_replay_script()` produces executable R code.
- `export_mfrm_bundle()` can optionally write the replay script to disk.

Interpreting output

The returned object contains:

- `summary`: a one-row overview of the chosen replay mode and whether bundle export was included
- `script`: the generated R code as a single string
- `anchors` and `group_anchors`: the exact stored constraints that were embedded into the script

If `ScriptMode` is `"facets"`, the script replays the higher-level `run_mfrm_facets()` workflow. If it is `"fit"`, the script uses `fit_mfrm()` directly.

Mode guide

- `"auto"` is the safest default and follows the structure of the supplied object.
- `"fit"` is useful when you want a minimal script centered on `fit_mfrm()`.
- `"facets"` is useful when you want to preserve the higher-level `run_mfrm_facets()` workflow, including stored column mapping.

Typical workflow

1. Finalize a fit and diagnostics object.
2. Generate the replay script with the path you want users to read from.
3. Write `replay$script` to disk, or let `export_mfrm_bundle()` do it for you.
4. Rerun the script in a fresh R session to confirm reproducibility.

See Also

`build_mfrm_manifest()`, `export_mfrm_bundle()`, `run_mfrm_facets()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
replay <- build_mfrm_replay_script(fit, data_file = "your_data.csv")
replay$summary[, c("ScriptMode", "ResidualPCA", "BiasPairs")]
cat(substr(replay$script, 1, 120))
```

`build_mfrm_resampling_spec` *Build an observed-data resampling specification*

Description

Build an observed-data resampling specification

Usage

```
build_mfrm_resampling_spec(
  data,
  person,
  facets,
  score,
  strata = NULL,
  preserve_facets = NULL,
  design = c("stratified_subsample", "stratified_bootstrap"),
  reps = 20,
  sample_fraction = 0.5,
  sample_n = NULL,
  replace = NULL,
  seed = NULL,
  min_per_stratum = 1,
  topup_preserve_facets = TRUE
)
```

Arguments

`data`	A long-format observed MFRM data set.
`person`	Person/respondent identifier column.
`facets`	Non-person facet columns used by the target MFRM fit.
`score`	Ordered score column.
`strata`	Optional person-level stratification columns, for example a `Region` or L1 group column. Each person must have at most one unique stratum combination.
`preserve_facets`	Optional facet columns whose level coverage should be reviewed and, when possible, topped up after the stratified person draw. A common choice is the rater facet.

design — Resampling design. "stratified_subsample" samples persons without replacement inside each stratum. "stratified_bootstrap" samples persons with replacement inside each stratum and re-keys duplicate person instances in the returned data.

reps — Number of resampling replicates to draw.

sample_fraction — Fraction of persons to draw within each stratum when sample_n = NULL.

sample_n — Optional target number of persons to draw per stratum. Supply either one scalar used for every stratum, or a named numeric vector whose names match the computed stratum labels.

replace — Optional logical override for replacement. By default, replacement is FALSE for "stratified_subsample" and TRUE for "stratified_bootstrap".

seed — Optional seed used by draw_mfrm_resamples().

min_per_stratum — Minimum target persons per represented stratum.

topup_preserve_facets — Logical; if TRUE, add extra person clusters when possible to recover missing levels of preserve_facets.

Details

This helper defines a resampling design for observed-data stability checks. It is intentionally separate from build_mfrm_sim_spec() and evaluate_mfrm_recovery(). The full-data estimates used with these draws are reference estimates, not known truth, so downstream summaries should be described as estimation stability, reproducibility, or agreement with a full-data reference rather than strict parameter recovery.

The design is person-clustered: all rows for a selected person are kept together. For bootstrap draws, duplicated person clusters are re-keyed in the returned data while the original identifier is retained in .mfrm_original_person.

Value

An object of class mfrm_resampling_spec.

See Also

draw_mfrm_resamples(), build_mfrm_sim_spec()

Examples

```
toy <- simulate_mfrm_data(n_person = 12, n_rater = 3, n_criterion = 2,
                          raters_per_person = 2, seed = 11)
region_map <- setNames(rep(c("A", "B", "C"),
                           length.out = length(unique(toy$Person))),
                       unique(toy$Person))
toy$Region <- unname(region_map[toy$Person])
spec <- build_mfrm_resampling_spec(
  toy, person = "Person", facets = c("Rater", "Criterion"),
```

```
  score = "Score", strata = "Region", preserve_facets = "Rater",
  reps = 2, sample_fraction = 0.5, seed = 99
)
draws <- draw_mfrm_resamples(spec)
summary(draws)$overview
```

`build_mfrm_sim_spec` *Build an explicit simulation specification for MFRM design studies*

Description

Build an explicit simulation specification for MFRM design studies

Usage

```
build_mfrm_sim_spec(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  thresholds = NULL,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  facet_names = NULL,
 assignment = c("crossed", "rotating", "sparse_linked", "resampled", "skeleton"),
  latent_distribution = c("normal", "empirical"),
  empirical_person = NULL,
  empirical_rater = NULL,
  empirical_criterion = NULL,
  assignment_profiles = NULL,
  design_skeleton = NULL,
  sparse_controls = NULL,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL,
  population_formula = NULL,
  population_coefficients = NULL,
  population_sigma2 = NULL,
  population_covariates = NULL
)
```

Arguments

n_person	Number of persons/respondents to generate.
n_rater	Number of rater facet levels to generate.
n_criterion	Number of criterion/item facet levels to generate.
raters_per_person	Number of raters assigned to each person.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases implied by facet_names (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). The schema-only future branch input design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the currently exposed facet keys. Do not specify the same variable through both design and the scalar count arguments.
score_levels	Number of ordered score categories.
theta_sd	Standard deviation of simulated person measures.
rater_sd	Standard deviation of simulated rater severities.
criterion_sd	Standard deviation of simulated criterion difficulties.
noise_sd	Optional observation-level noise added to the linear predictor.
step_span	Spread used to generate equally spaced thresholds when thresholds = NULL.
thresholds	Optional threshold specification. Use a numeric vector of common thresholds; a named list such as list(C01 = c(-1, 0, 1)); a numeric matrix with one row per StepFacet and one column per step; or a long data frame with columns StepFacet, Step/StepIndex, and Estimate.
model	Measurement model recorded in the simulation specification.
step_facet	Step facet used when model = "PCM" and threshold values vary across levels.
slope_facet	Slope facet used when model = "GPCM". The current bounded GPCM branch requires slope_facet == step_facet.
slopes	Optional slope specification for model = "GPCM". Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns SlopeFacet and Estimate. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean- one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact PCM reduction. The GPCM model form follows Muraki's generalized partial credit model; the package's slope-regime labels are validation stress labels, not published psychometric cut points.
facet_names	Optional public names for the two simulated non-person facet columns. Supply either an unnamed character vector of length 2 in rater-like / criterion-like order, or a named vector with names c("rater", "criterion").
assignment	Assignment design. "crossed" means every person sees every rater; "rotating" uses a balanced rotating subset; "sparse_linked" uses an incomplete rating design with optional linking persons; "resampled" reuses empirical person-level rater-assignment profiles; "skeleton" reuses an observed person-by-facet design skeleton.

latent_distribution
: Latent-value generator. "normal" samples from centered normal distributions using the supplied standard deviations. "empirical" resamples centered support values from empirical_person/empirical_rater/empirical_criterion.

empirical_person
: Optional numeric support values used when latent_distribution = "empirical".

empirical_rater
: Optional numeric support values used when latent_distribution = "empirical".

empirical_criterion
: Optional numeric support values used when latent_distribution = "empirical".

assignment_profiles
: Optional data frame with columns TemplatePerson and the public rater-like facet column (optionally Group) describing empirical person-level rater-assignment profiles used when assignment = "resampled". The canonical name Rater is also accepted.

design_skeleton
: Optional data frame with columns TemplatePerson, the public rater-like facet column, and the public criterion-like facet column (optionally Group, Weight, and TemplatePersonReuse) describing an observed response skeleton used when assignment = "skeleton". The canonical names Rater and Criterion are also accepted. TemplatePersonReuse = TRUE asks simulate_mfrm_data() to keep the template-person order instead of resampling templates, which is used by build_peer_review_sim_spec().

sparse_controls
: Optional named list used when assignment = "sparse_linked". Supported entries are link_fraction (default 0.1), link_persons (overrides link_fraction), link_raters_per_person (default n_rater), assignment_mode ("balanced" or "random"), and min_common_persons_per_rater_pair (diagnostic target, default 1).

group_levels
: Optional character vector of group labels.

dif_effects
: Optional data frame of true group-linked DIF effects.

interaction_effects
: Optional data frame of true interaction effects.

population_formula
: Optional one-sided formula describing a person-level latent-regression population model used when generating person measures, for example ~ X + G. When supplied, person measures are generated from X %*% beta + e rather than from N(0, theta_sd^2).

population_coefficients
: Optional numeric vector of latent-regression coefficients corresponding to the design matrix implied by population_formula.

population_sigma2
: Optional residual variance for the latent-regression person distribution.

population_covariates
: Optional template data frame containing one row per template person and the background variables referenced by population_formula. Numeric/logical

and categorical factor/character variables are expanded through the same `stats::model.matrix()` contract used by latent-regression fitting. During simulation, template rows are resampled to the requested `n_person`.

Details

`build_mfrm_sim_spec()` creates an explicit, portable simulation specification that can be passed to `simulate_mfrm_data()`. The goal is to make the data-generating mechanism inspectable and reusable rather than relying only on ad hoc scalar arguments.

The resulting object records:

- design counts (`n_person`, `n_rater`, `n_criterion`, `raters_per_person`)
- latent spread assumptions (`theta_sd`, `rater_sd`, `criterion_sd`)
- optional empirical latent support values for semi-parametric simulation
- threshold structure (`threshold_table`)
- optional discrimination structure for bounded `GPCM` (`slope_table`) and its identified log-slope spread label (`slope_regime`)
- assignment design (`assignment`)
- optional sparse linked-design controls (`sparse_controls`) when `assignment = "sparse_linked"`
- optional empirical assignment profiles (`assignment_profiles`) with optional person-level `Group` labels
- optional observed response skeleton (`design_skeleton`) with optional person-level `Group` labels and observation-level `Weight` values
- optional person-level latent-regression population metadata including `population_formula`, `population_coefficients`, `population_sigma2`, and a reusable template of person-level covariates, including model-matrix xlevel/contrast provenance for categorical covariates
- `planning_scope`, an explicit record that the current planning/forecasting helpers target the role-based person x rater-like x criterion-like design contract rather than a fully arbitrary-facet planner
- `planning_constraints`, an explicit record of which design variables can currently be changed from that specification without rebuilding it
- `planning_schema`, a combined schema contract bundling the role descriptor, scope boundary, current mutability map, a `facet_manifest`, a schema-only `future_facet_table`, and a matching `future_design_template`, plus a nested `future_branch_schema` scaffold for a future arbitrary-facet planning branch
- the current `design$facets(...)` parser now normalizes nested facet-count input through that bundled `future_branch_schema`, whose nested `design_schema` is now the authoritative schema-only branch object
- optional signal tables for DIF and interaction bias

The current generator targets the package's standard person x rater x criterion workflow, but the public output names for those two facet roles can now be customized with `facet_names`. This naming layer improves public ergonomics; it does not yet turn the generator into a fully arbitrary-facet simulator. Internally, helper objects keep canonical role mappings so that planning functions can treat the first non-person facet as rater-like and the second as criterion-like. When threshold values are

provided by StepFacet, the supported step facets are the generated levels of the chosen public rater-like or criterion-like column. For convenience, step-facet-specific thresholds can be supplied as a named list or as a numeric matrix whose row names are StepFacet labels. When model = "GPCM", the same public facet naming rules apply to the slope table; the current bounded branch keeps slope_facet equal to step_facet. The slope_regime field summarizes the centered log-slope spread so recovery simulations can be read against the intended generator stress level. Its labels (unit_slopes, near_flat, moderate, and high_dispersion) are package validation labels; they are not model-fit decisions and should not be interpreted as literature-derived adequacy thresholds. The GPCM data-generating form follows Muraki (1992, doi:10.1177/014662169201600206), while information-function interpretation follows Muraki (1993, doi:10.1177/014662169301700403). The explicit simulation-specification metadata is intended to support ADEMP-style simulation reporting as described by Morris, White, and Crowther (2019, doi:10.1002/sim.8086).

The assignment = "sparse_linked" branch follows sparse rater-mediated assessment work in treating incomplete rater assignment as planned missingness rather than incidental nonresponse. Its link_persons and link_raters_per_person controls emulate a common linking set so users can inspect rater-pair common-person counts before using the generated data in recovery or design studies. This is a design generator and diagnostic metadata layer; it does not impose a universal minimum-linking cutoff. Sparse-design motivation follows Wind, Jones, and Grajeda (2023, doi:10.1177/01466216231182148), Wind and Jones (2018, doi:10.1177/0013164417703733), and DeMars, Shapovalov, and Hathcoat (2023).

If population_formula is supplied, the simulation specification carries a first-version person-level latent-regression generator. This affects only the person distribution. The current implementation keeps the non-person facets in the existing many-facet Rasch generator and resamples rows from population_covariates to the requested design size before computing $\theta_n = x_n^\top \beta + \varepsilon_n$ with $\varepsilon_n \sim N(0, \sigma^2)$.

Value

An object of class mfrm_sim_spec.

Interpreting output

This object does not contain simulated data. It is a data-generating specification that tells simulate_mfrm_data() how to generate them.

See Also

extract_mfrm_sim_spec(), simulate_mfrm_data()

Examples

```
spec <- build_mfrm_sim_spec(
  design = list(person = 8, rater = 2, criterion = 2, assignment = 1),
  assignment = "rotating"
)
spec$model
spec$assignment
nrow(spec$threshold_table)
```

`build_misfit_casebook` *Build a case-level misfit review bundle*

Description

Build a case-level misfit review bundle

Usage

```
build_misfit_casebook(
  fit,
  diagnostics = NULL,
  unexpected = NULL,
  displacement = NULL,
  administration_id = NULL,
  wave_id = NULL,
  top_n = 25
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`unexpected`	Optional output from `unexpected_response_table()`.
`displacement`	Optional output from `displacement_table()`.
`administration_id`	Optional scalar identifier describing the current administration or form. It is stored in row-level provenance and summary outputs when supplied.
`wave_id`	Optional scalar identifier for the current wave or occasion. It is stored in row-level provenance and summary outputs when supplied.
`top_n`	Maximum number of rows to keep in compact summary outputs.

Details

`build_misfit_casebook()` is a synthesis layer over package-native screening outputs. It does not invent a new misfit statistic. Instead, it organizes existing evidence families into one case-level review surface:

- element-level Infit / Outfit MnSq misfit from `diagnostics$fit` (rows whose Infit or Outfit MnSq falls outside the 0.5-1.5 Linacre acceptance band)
- strict marginal cell screens from `diagnostics$marginal_fit$top_cells`
- strict pairwise screens from `diagnostics$marginal_fit$pairwise$top_pairs`
- unexpected responses from `unexpected_response_table()`
- displacement flags from `displacement_table()`

The result is an operational review bundle. It is not a formal adjudication system, and repeated signals across evidence families should be prioritized over any single isolated case row. In addition to raw case rows, the object includes stable grouping views such as `by_person`, `by_facet_level`, `by_source_family`, and `by_wave` to support operational triage. The `source_support` component records which evidence families are currently supported, caveated, or deferred under the active model.

Value

An object of class `mfrm_misfit_casebook`.

Recommended input route

1. Fit with `fit_mfrm()`.
2. Build diagnostics with `diagnose_mfrm()`.
3. Optionally build `unexpected_response_table()` and `displacement_table()` yourself when you want custom thresholds before synthesizing the casebook.

GPCM boundary

For bounded `GPCM`, the helper is available with caveat. The casebook inherits exploratory screening semantics from the underlying residual and strict marginal sources; it should not be read as a formal inferential case test.

See Also

`diagnose_mfrm()`, `unexpected_response_table()`, `displacement_table()`, `plot_unexpected()`, `plot_displacement()`, `plot_marginal_fit()`, `plot_marginal_pairwise()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", model = "RSM", quad_points = 5)
diag <- diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")
casebook <- build_misfit_casebook(fit, diagnostics = diag, top_n = 10)
summary(casebook)
casebook$top_cases
```

---

build_model_choice_review

Build a model-choice review across RSM, PCM, and bounded GPCM fits

---

Description

Build a model-choice review across RSM, PCM, and bounded GPCM fits

Usage

```
build_model_choice_review(
  ...,
  labels = NULL,
  run_weighting_review = NULL,
  theta_range = c(-6, 6),
  theta_points = 61L,
  top_n = 10L,
  warn_constraints = TRUE
)
```

Arguments

`...`	Two or more fitted `mfrm_fit` objects from `fit_mfrm()`.
`labels`	Optional labels for the supplied fits. If omitted, names from `...` are used when available; otherwise labels are generated from model/method combinations.
`run_weighting_review`	Logical. If `TRUE` and the supplied fits include at least one `RSM`/`PCM` reference plus one bounded `GPCM` fit, also run `build_weighting_review()` for the first such pair.
`theta_range, theta_points, top_n`	Passed to `build_weighting_review()` when `run_weighting_review = TRUE`.
`warn_constraints`	Passed to `compare_mfrm()`.

Details

`build_model_choice_review()` is a user-facing synthesis helper. It does not estimate new models. It bundles:

- `compare_mfrm()` for AIC/BIC/log-likelihood comparison;
- model-role guidance for `RSM`, `PCM`, and bounded `GPCM`;
- downstream-route availability for APA output, score-side export, linking, recovery, fair averages, bias screening, and summary-appendix handoff;
- report wording templates that avoid treating better bounded-`GPCM` fit as an automatic operational-scoring decision;
- `gpcm_capability_matrix()` when bounded `GPCM` is present;
- optionally, `build_weighting_review()` for the first Rasch-family reference versus bounded-`GPCM` pair.

The word "bounded" is intentional: the package implements a bounded GPCM route, not every possible generalized partial-credit many-facet extension. The current route uses positive slopes, requires `slope_facet == step_facet`, identifies slopes on the log scale with geometric mean 1, and keeps several downstream score-side/reporting helpers outside the validated boundary.

Value

An object of class `mfrm_model_choice_review`.

See Also

compare_mfrm(), build_weighting_review(), gpcm_capability_matrix(), compute_information()

Examples

```
toy <- load_mfrmr_data("example_core")
fit_rsm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "RSM", quad_points = 7)
fit_pcm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "PCM", step_facet = "Criterion",
                    quad_points = 7)
review <- build_model_choice_review(RSM = fit_rsm, PCM = fit_pcm)
summary(review)
```

---

build_peer_review_design_review

Build a peer-review design review

Description

Build a peer-review design review

Usage

```
build_peer_review_design_review(peer_review_design, top_n = 10)
```

Arguments

peer_review_design	A generated data frame carrying the mfrm_peer_review_design attribute, the attribute itself, or its overview data frame.
top_n	Number of reviewer-load, submission-load, common-link, and reciprocal-pair rows to keep in compact summary tables.

Details

build_peer_review_design_review() converts peer-review simulation metadata into a reportable design-review object. The review summarizes self-review checks, reviewer and submission load, common submissions per reviewer pair, and reciprocal review pairs. These rows are assignment-design diagnostics: they do not replace MFRM estimates, fit, separation, reliability, or substantive review-quality evidence.

Peer-review use of MFRM follows studies that model peer/self/teacher rater severity and leniency. Common-link anchor interpretation follows sparse rater-mediated design work; the review status is therefore descriptive and conservative rather than a literature-derived universal adequacy cutoff.

Value

A bundle of class `mfrm_peer_review_design_review`.

References

- Farrokhi, F., Esfandiari, R., & Schaefer, E. (2012). A many-facet Rasch measurement of differential rater severity/leniency in three types of assessment. *JALT Journal*, 34(1), 79-102. doi:10.37546/JALTJJ34.1-3.
- Uto, M., & Ueno, M. (2020). A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo. *Behaviormetrika*, 47, 469-496. doi:10.1007/s41237-020-00115-7.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

See Also

`build_peer_review_sim_spec()`, `simulate_mfrm_data()`, `build_mfrm_network_review()`, `build_summary_table_bu`

Examples

```
peer_spec <- build_peer_review_sim_spec(
  n_submission = 12,
  n_criterion = 3,
  reviewers_per_submission = 2,
  anchor_submissions = 2
)
peer_sim <- simulate_mfrm_data(sim_spec = peer_spec, seed = 123)
review <- build_peer_review_design_review(peer_sim)
summary(review)$overview
```

---

build_peer_review_sim_spec

Build a peer-review simulation specification

Description

Build a peer-review simulation specification

Usage

```
build_peer_review_sim_spec(
  n_submission = 50,
  n_criterion = 4,
  reviewers_per_submission = 3,
  anchor_fraction = 0.1,
  anchor_submissions = NULL,
  anchor_reviewers_per_submission = NULL,
```

```
  avoid_self_review = TRUE,
  assignment_mode = c("balanced", "random"),
  seed = NULL,
  score_levels = 4,
  theta_sd = 1,
  reviewer_sd = 0.45,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = "Criterion",
  thresholds = NULL,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL
)
```

Arguments

n_submission	Number of submissions/authors to generate.
n_criterion	Number of rubric criteria.
reviewers_per_submission	Number of peer reviewers assigned to each ordinary submission.
anchor_fraction	Fraction of submissions treated as common-link anchor submissions when anchor_submissions is not supplied.
anchor_submissions	Optional number of common-link anchor submissions. Anchor submissions receive anchor_reviewers_per_submission reviewers.
anchor_reviewers_per_submission	Number of reviewers assigned to each anchor submission. Defaults to all eligible peers when anchors are used and self-review is disallowed; recorded as 0 when no anchor submissions are requested.
avoid_self_review	Logical; if TRUE, a reviewer is never assigned to review their own submission.
assignment_mode	Assignment algorithm. "balanced" assigns reviewers with the lowest current load using deterministic rotating tie-breaks. "random" samples eligible reviewers without replacement.
seed	Optional seed used only for random peer-review assignment when assignment_mode = "random".
score_levels, theta_sd, reviewer_sd, criterion_sd, noise_sd, step_span	Generator settings passed to build_mfrm_sim_spec(). reviewer_sd maps to the standard MFRM rater-severity spread.
model, step_facet, thresholds	Measurement-model settings passed to build_mfrm_sim_spec(). The first public facet is Reviewer; the second is Criterion.

`group_levels`, `dif_effects`, `interaction_effects`
: Optional signal settings passed to `build_mfrm_sim_spec()`.

Details

`build_peer_review_sim_spec()` creates a fixed person-by-reviewer-by-rubric skeleton for peer-assessment or peer-review studies. Submissions and peer reviewers share the same ID universe (`P001`, `P002`, ...), so self-review can be structurally excluded and checked in the generated data. The specification uses the existing `assignment = "skeleton"` generator and records peer-review metadata; it does not introduce a new measurement model. MFRM still estimates person/submission measures, reviewer severity, and criterion difficulty, while design-network review can inspect whether the peer-review graph is sufficiently linked.

The common-link anchor controls follow the same logic used in sparse rater-mediated designs: when most submissions receive only a few peer reviews, assigning all or many reviewers to a small anchor set can strengthen links among reviewers. The helper labels these rows as design diagnostics, not universal adequacy thresholds for fit, separation, or recovery.

Value

An object of class `mfrm_sim_spec` with `peer_review` metadata and a fixed peer-review design skeleton.

References

- Farrokhi, F., Esfandiari, R., & Schaefer, E. (2012). A many-facet Rasch measurement of differential rater severity/leniency in three types of assessment. *JALT Journal*, 34(1), 79-102. doi:10.37546/JALTJJ34.1-3.
- Uto, M., & Ueno, M. (2020). A generalized many-facet Rasch model and its Bayesian estimation using Hamiltonian Monte Carlo. *Behaviormetrika*, 47, 469-496. doi:10.1007/s41237-020-00115-7.
- DeMars, C. E., Shapovalov, Y. A., & Hathcoat, J. D. (2023). *Many-Facet Rasch Designs: How Should Raters be Assigned to Examinees?* NCME presentation.

See Also

`simulate_mfrm_data()`, `build_mfrm_network_review()`, `build_mfrm_sim_spec()`

Examples

```
peer_spec <- build_peer_review_sim_spec(
  n_submission = 12,
  n_criterion = 3,
  reviewers_per_submission = 2,
  anchor_submissions = 2
)
peer_spec$peer_review$overview
```

build_summary_table_bundle *Build a manuscript-oriented table bundle from* summary() *outputs*

Description

Build a manuscript-oriented table bundle from summary() outputs

Usage

```
build_summary_table_bundle(
  x,
  which = NULL,
  appendix_preset = NULL,
  include_empty = FALSE,
  digits = 3,
  top_n = 10,
  preview_chars = 160
)
```

Arguments

x	An mfrm_fit, mfrm_diagnostics, mfrm_precision_review, mfrm_fit_measures, mfrm_facets_fit_review, mfrm_person_fit_indices, mfrm_data_description, mfrm_reporting_checklist, mfrm_apa_outputs, mfrm_design_evaluation, mfrm_signal_detection, mfrm_recovery_simulation, mfrm_recovery_assessment, mfrm_population_prediction, mfrm_future_branch_active_branch, mfrm_facets_run, mfrm_bias, mfrm_anchor_review, mfrm_linking_review, mfrm_misfit_casebook, mfrm_weighting_review, mfrm_unit_prediction, or mfrm_plausible_values object, one of their summary() outputs, or a summary.mfrmr_recovery_validation object from the packaged validation protocol.
which	Optional character vector selecting a subset of named tables.
appendix_preset	Optional appendix-oriented table preset: "all", "recommended", "compact", "methods", "results", "diagnostics", or "reporting". Cannot be combined with which. Section-aware presets keep returned tables whose bundle catalog maps to the requested appendix section.
include_empty	If TRUE, retain empty tables in the returned bundle.
digits	Digits forwarded when summary() must be computed from a raw object.
top_n	Row cap forwarded to compact summary() methods when x is a raw object.
preview_chars	Character cap forwarded to summary.mfrm_apa_outputs() when x is a raw APA-output object.

Details

This helper turns the package's compact summary objects into a reproducible table bundle for manuscript drafting, appendix handoff, or downstream formatting. It does not replace apa_table(); instead, it provides a consistent bridge from summary() to named data.frame components that can later be rendered with apa_table() or exported directly.

The public entry point validates x and the summary-object contract up front, so malformed summaries fail with a package-level message instead of falling through to opaque downstream errors.

The function first normalizes x through the corresponding summary() method when needed, then records a table_index describing every available table and returns the selected tables in tables. Optional appendix presets can be applied at bundle-construction time when you want a conservative manuscript-facing subset before plotting or export.

Value

An object of class mfrm_summary_table_bundle with:

- overview
- table_index
- plot_index
- tables
- appendix_preset
- notes
- source_class
- summary_class

Supported inputs

- fit_mfrm() or summary(fit)
- diagnose_mfrm() or summary(diag)
- precision_review_report() or summary(precision_review)
- fit_measures_table() or summary(fit_measures)
- facets_fit_review() or summary(facets_fit_review)
- compute_person_fit_indices() or summary(person_fit)
- describe_mfrm_data() or summary(ds)
- reporting_checklist() or summary(chk)
- build_apa_outputs() or summary(apa)
- evaluate_mfrm_design() or summary(sim_eval)
- evaluate_mfrm_signal_detection() or summary(sig_eval)
- evaluate_mfrm_recovery() or summary(rec)
- assess_mfrm_recovery() or summary(rec_assessment)
- summary(validation) from recovery-validation.R
- predict_mfrm_population() or summary(pred)

- planning_schema$future_branch_active_branch or summary(...)
- run_mfrm_facets() or summary(out)
- estimate_bias() or summary(bias)
- review_mfrm_anchors() or summary(review)
- build_linking_review() or summary(review)
- build_misfit_casebook() or summary(casebook)
- build_weighting_review() or summary(review)
- predict_mfrm_units() or summary(pred_units)
- sample_mfrm_plausible_values() or summary(pv)

Interpreting output

- overview: one-row metadata about the source summary and table counts.
- table_index: table names, dimensions, roles, and manuscript-oriented descriptions.
- plot_index: which returned tables contain numeric content and which bundle-level plot types can use them directly.
- tables: named data.frame objects ready for formatting or export.
- appendix_preset: active appendix subset mode ("none" when not used).
- notes: short guidance about omitted empty tables or source-level caveats.
- fit-level caveats use the analysis_caveats role; pre-fit data score-support caveats use the score_category_caveats role. Both roles are classified as diagnostics and stay in recommended appendix subsets.
- recovery-assessment and recovery-validation summaries expose diagnostic_reporting_notes before diagnostic_review or diagnostic_oc_summary so fit/separation caveats can be reported without treating them as recovery or release gates.
- recovery-validation summaries expose condition_reporting_notes before condition_summary so GPCM generator stress and sparse score support are not mistaken for recovery-metric failures.
- precision-review summaries expose fit_separation_basis so fit, ZSTD, separation/reliability/strata, and QC thresholds remain separate reporting surfaces rather than implicit validation gates.
- fit-measure and FACETS fit-review summaries expose df/ZSTD sensitivity tables under precision-review roles, keeping MnSq status, ZSTD standardization, and external FACETS matching distinct in appendix handoffs.
- latent-regression fit summaries expose population_coding in the methods appendix role so categorical levels, contrasts, and encoded columns can be documented with the coefficient table.

Typical workflow

1. Build a compact object with summary(...).
2. Convert it with build_summary_table_bundle(...).
3. Use bundle$tables[[...]] directly, or hand a selected table to apa_table() for formatted manuscript output.

4. If you want a manuscript appendix subset up front, use a preset such as `appendix_preset = "recommended"`, `"compact"`, or `"diagnostics"`.
5. For recovery-assessment or recovery-validation summaries, inspect `bundle$tables$reading_order` first when it is available.
6. For recovery-assessment or recovery-validation summaries with retained diagnostics, read `diagnostic_reporting_notes` before the raw `diagnostic_review` or `diagnostic_oc_summary`. Read `condition_reporting_notes` before `condition_review` or `condition_summary` when bounded GPCM generator stress is part of the plan.

See Also

`summary()`, `apa_table()`, `reporting_checklist()`, `build_apa_outputs()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
bundle <- build_summary_table_bundle(fit)
bundle$table_index
summary(bundle)$role_summary


# Recovery-validation output can be converted to appendix-ready tables.
## Not run:
source(system.file("validation", "recovery-validation.R", package = "mfrmr"))
validation <- mfrmr_run_recovery_validation(
  case_ids = c("gpcm_slope_profile", "gpcm_high_dispersion_sparse"),
  quick = TRUE,
  seed = 20260525
)
validation_bundle <- build_summary_table_bundle(summary(validation))
validation_bundle$tables$reading_order
validation_bundle$tables$topline_release_decision
validation_bundle$tables$condition_reporting_notes
validation_bundle$tables$condition_summary
validation_bundle$tables$diagnostic_reporting_notes

## End(Not run)
```

`build_visual_summaries` *Build warning and narrative summaries for visual outputs*

Description

Build warning and narrative summaries for visual outputs

Usage

```
build_visual_summaries(
  fit,
  diagnostics,
  threshold_profile = "standard",
  thresholds = NULL,
  summary_options = NULL,
  whexact = FALSE,
  branch = c("original", "facets")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Output from diagnose_mfrm().
threshold_profile	Threshold profile name (strict, standard, lenient).
thresholds	Optional named overrides for profile thresholds.
summary_options	Summary options for build_visual_summary_map().
whexact	Use exact ZSTD transformation.
branch	Output branch: "facets" adds FACETS crosswalk metadata for manual-aligned reporting; "original" keeps package-native summary output.

Details

This function returns visual-keyed text maps to support dashboard/report rendering without hard-coding narrative strings in UI code.

thresholds can override any profile field by name. Common overrides:

- n_obs_min, n_person_min
- misfit_ratio_warn, zstd2_ratio_warn, zstd3_ratio_warn
- pca_first_eigen_warn, pca_first_prop_warn

summary_options supports:

- detail: "standard" or "detailed"
- max_facet_ranges: max facet-range snippets shown in visual summaries
- top_misfit_n: number of top misfit entries included

For bounded GPCM, this helper returns caveated warning/summary maps over supported diagnostics, direct tables, and plots. The returned object includes gpcm_boundary so score-side, design-forecasting, DFF, and linking routes remain visibly separate capability rows.

Value

An object of class `mfrm_visual_summaries` with:

- `warning_map`: visual-level warning text vectors
- `summary_map`: visual-level descriptive text vectors
- `warning_counts`, `summary_counts`: message counts by visual key
- `plot_payloads`: reusable draw-free `mfrm_plot_data` objects for `comparison`, `warning_counts`, `summary_counts`, and optionally `category_probability_surface`
- `public_plot_routes`: public helper / draw-free route map for follow-up
- `crosswalk`: FACETS-reference mapping for main visual keys
- `branch`, `style`, `threshold_profile`: branch metadata

Interpreting output

- `warning_map`: rule-triggered warning text by visual key.
- `summary_map`: descriptive narrative text by visual key.
- strict marginal keys appear when `diagnose_mfrm(..., diagnostic_mode = "both")` supplies latent-integrated first-order and pairwise screening summaries.
- `warning_counts` / `summary_counts`: message-count tables for QA checks.
- `plot_payloads`: ready-to-reuse `mfrm_plot_data` objects for the bundle's own comparison/count plots and, when step estimates are available, the exploratory `category_probability_surface` data from `plot(fit, type = "ccc_surface", draw = FALSE)`. The surface data carry `category_support`, `interpretation_guide`, and `reporting_policy` tables for zero-frequency category and reporting-boundary checks.
- `public_plot_routes`: draw-free helper routes for the dedicated public plot functions behind each visual family.

Typical workflow

1. inspect defaults with `mfrm_threshold_profiles()`
2. choose `threshold_profile` (`strict` / `standard` / `lenient`)
3. optionally override selected fields via `thresholds`
4. pass result maps to report/dashboard rendering logic

See Also

`mfrm_threshold_profiles()`, `build_apa_outputs()`, `plot_marginal_fit()`, `plot_marginal_pairwise()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", model = "RSM", quad_points = 7, maxit = 30
)
```

```
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
vis <- build_visual_summaries(fit, diag, threshold_profile = "strict")
vis2 <- build_visual_summaries(
  fit,
  diag,
  threshold_profile = "standard",
  thresholds = c(misfit_ratio_warn = 0.20, pca_first_eigen_warn = 2.0),
  summary_options = list(detail = "detailed", top_misfit_n = 5)
)
vis_facets <- build_visual_summaries(fit, diag, branch = "facets")
vis_facets$branch
summary(vis)
p <- plot(vis, type = "comparison", draw = FALSE)
p2 <- plot(vis, type = "warning_counts", draw = FALSE)
vis$plot_payloads$comparison$data$plot
vis$public_plot_routes[, c("Visual", "PlotHelper", "DrawFreeRoute")]
if (interactive()) {
  plot(
    vis,
    type = "comparison",
    draw = TRUE,
    main = "Warning vs Summary Counts (Customized)",
    palette = c(warning = "#cb181d", summary = "#3182bd"),
    label_angle = 45
  )
}
```

`build_weighting_review` *Build a weighting-policy review between Rasch-family and bounded GPCM fits*

Description

Build a weighting-policy review between Rasch-family and bounded GPCM fits

Usage

```
build_weighting_review(
  rasch_fit,
  gpcm_fit,
  theta_range = c(-6, 6),
  theta_points = 101L,
  top_n = 10L
)
```

Arguments

rasch_fit	Output from fit_mfrm() using model = "RSM" or "PCM".
gpcm_fit	Output from fit_mfrm() using bounded model = "GPCM".
theta_range	Numeric vector of length 2 passed to compute_information() for the information-redistribution comparison.
theta_points	Integer number of theta grid points passed to compute_information().
top_n	Maximum number of rows to keep in compact summary outputs.

Details

build_weighting_review() is an operational model-choice review helper. It is designed for the common question:

- what changes when a Rasch-family equal-weighting model is replaced with a bounded GPCM that allows discrimination-based reweighting?

The helper does not estimate a new model. Instead, it synthesizes four package-native evidence sources:

- compare_mfrm() for same-data model comparison
- the non-person facet measures from each fit
- the bounded GPCM slope table
- compute_information() for design-weighted information redistribution

The result is intended for substantive review, not for automatic model selection. In particular, a better-fitting GPCM should not by itself be interpreted as a reason to discard an equal-weighting Rasch-family route.

Value

An object of class mfrm_weighting_review.

Recommended input route

1. Fit an equal-weighting reference model with model = "RSM" or "PCM".
2. Fit a bounded GPCM on the same prepared response data.
3. Run build_weighting_review(rasch_fit, gpcm_fit).
4. Read summary(review) before deciding whether the discrimination-based reweighting is substantively acceptable.

What the returned tables mean

- model_comparison: same-data model-comparison bundle from compare_mfrm().
- facet_shift: how non-person facet estimates move under bounded GPCM.
- slope_profile: which slope_facet levels are upweighted or downweighted.
- information_redistribution: within-facet information-share changes between the Rasch-family fit and bounded GPCM.
- top_reweighted_levels: compact triage table for the strongest slope-facet-level redistribution signals.

GPCM boundary

This helper is available only for the current bounded GPCM branch. It requires the package's existing slope_facet == step_facet contract and should be read as an operational weighting-policy review, not as a formal validity adjudication.

See Also

compare_mfrm(), compute_information(), gpcm_capability_matrix()

Examples

```
toy <- load_mfrmr_data("example_core")
rasch_fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "MML",
  model = "RSM",
  quad_points = 9
)
gpcm_fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "MML",
  model = "GPCM",
  step_facet = "Criterion",
  slope_facet = "Criterion",
  quad_points = 9
)
review <- build_weighting_review(rasch_fit, gpcm_fit, theta_points = 41)
summary(review)
review$top_reweighted_levels
```

---

category_curves_report

Build a category curve export bundle (preferred alias)

Description

Build a category curve export bundle (preferred alias)

Usage

```
category_curves_report(
  fit,
  theta_range = c(-6, 6),
  theta_points = 241,
  digits = 4,
  include_fixed = FALSE,
  fixed_max_rows = 400
)
```

Arguments

fit	Output from fit_mfrm().
theta_range	Theta/logit range for curve coordinates.
theta_points	Number of points on the theta grid.
digits	Rounding digits for numeric graph output.
include_fixed	If TRUE, include a legacy-compatible fixed-width text block.
fixed_max_rows	Maximum rows shown in fixed-width graph tables.

Details

Preferred high-level API for category-probability curve exports. Returns tidy curve coordinates and summary metadata for quick plotting/report integration without calling low-level helpers directly. The expected-score table also carries the per-curve score variance and information function. For GPCM, the information column follows the Muraki/Samejima identity $a^2\mathrm{Var}(X \mid \theta)$; for RSM / PCM, this reduces to the usual score variance because discrimination is fixed at one. The category_information table decomposes that total into category-level contributions, $a^2 P_k(\theta)(k - E[X \mid \theta])^2$, whose sum equals the reported information at the same theta value. The cumulative_probabilities table follows the FACETS / Winsteps graph convention of accumulating modeled probabilities across ordered categories (P(X <= k) by default, with P(X >= k) also returned for flipped curves). cumulative_boundaries reports approximate theta values where P(X <= k) = .5, with BoundaryStatus and CrossingCount to avoid over-interpreting boundaries outside the requested theta range or with multiple crossings.

Value

A named list with category-curve components. Class: mfrm_category_curves.

Interpreting output

Use this report to inspect:

- where each category has highest probability across theta
- where cumulative category probabilities cross .5
- whether adjacent categories cross in expected order
- whether probability bands look compressed (often sparse categories)

Recommended read order:

1. `summary(out)` for compact diagnostics.
2. `out$probabilities`, `out$expected_ogive`, and `out$category_information` for custom graphics.
3. `plot(out)` for a default visual check, or `plot(out, type = "cumulative")` to inspect cumulative probabilities. `plot(out, type = "information")` to inspect curve-level information. Use `plot(out, type = "category_information")` when category-level contributions are needed.

References

Category response curves follow Andrich's rating-scale formulation, Masters' partial-credit model, and Muraki's generalized partial-credit model. The `Information` column for bounded `GPCM` uses Muraki's item-information result obtained from Samejima's general polytomous information formula.

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176. doi:10.1177/014662169201600206
- Muraki, E. (1993). *Information functions of the generalized partial credit model*. Applied Psychological Measurement, 17(4), 351-363. doi:10.1177/014662169301700403

Typical workflow

1. Fit model with `fit_mfrm()`.
2. Run `category_curves_report()` with suitable `theta_points`.
3. Use `summary()` and `plot()`; export tables for manuscripts/dashboard use. `plot(out)` gives a four-panel overview. Use `preset = "monochrome"` for grayscale/line-type output and `boundary_status = "none"` when cumulative `.5` boundary lines should be suppressed. `plot(out, type = "category_probability")` and `plot(out, type = "conditional_probability")` are explicit aliases for the same category-probability curves as `type = "ccc"`. Use `plot_data(out, component = "plot_long")` when rebuilding the curves with ggplot2, plotly, or another R graphics system.

See Also

`category_structure_report()`, `rating_scale_table()`, `plot.mfrm_fit()`, mfrmr_reports_and_tables, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- category_curves_report(fit, theta_points = 101)
summary(out)
```

```
head(out$probabilities[, c("CurveGroup", "Theta", "Category", "Probability")])
p_overview <- plot(out, draw = FALSE)
p_overview$data$plot
p_cum <- plot(out, type = "cumulative", draw = FALSE)
head(p_cum$data$cumulative_boundaries)
p_info <- plot(out, type = "category_information", draw = FALSE)
head(p_info$data$category_information)
curve_long <- plot_data(out, component = "plot_long")
head(curve_long[, c("PlotType", "Theta", "Series", "Value")])
```

`category_structure_report` *Build a category structure report (preferred alias)*

Description

Build a category structure report (preferred alias)

Usage

```
category_structure_report(
  fit,
  diagnostics = NULL,
  theta_range = c(-6, 6),
  theta_points = 241,
  drop_unused = FALSE,
  include_fixed = FALSE,
  fixed_max_rows = 200
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`theta_range`	Theta/logit range used to derive transition points.
`theta_points`	Number of grid points used for transition-point search.
`drop_unused`	If `TRUE`, remove zero-count categories from outputs.
`include_fixed`	If `TRUE`, include a legacy-compatible fixed-width text block.
`fixed_max_rows`	Maximum rows per fixed-width section.

Details

Preferred high-level API for category-structure diagnostics. This wraps the legacy-compatible bar/transition export and returns a stable bundle interface for reporting and plotting.

Value

A named list with category-structure components. Class: `mfrm_category_structure`.

Interpreting output

Key components include:

- category usage/fit table (count, expected, infit/outfit, ZSTD)
- threshold ordering and adjacent threshold gaps
- category transition-point table on the requested theta grid

Practical read order:

1. `summary(out)` for compact warnings and threshold ordering.
2. `out$category_table` for sparse/misfitting categories.
3. `out$median_thresholds` for adjacent-threshold caveats when zero-count categories are retained.
4. `plot(out)` for quick visual check.

Typical workflow

1. `fit_mfrm()` -> model.
2. `diagnose_mfrm()` -> residual/fit diagnostics (optional argument here).
3. `category_structure_report()` -> category health snapshot.
4. `summary()` and `plot()` for draft-oriented review of category structure.

See Also

`rating_scale_table()`, `category_curves_report()`, `plot.mfrm_fit()`, mfrmr_reports_and_tables, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- category_structure_report(fit)
summary(out)
head(out$category_table[, c("Category", "Count", "Infit", "Outfit")])
p_cs <- plot(out, draw = FALSE)
p_cs$data$plot
```

compare_mfrm *Compare two or more fitted MFRM models*

Description

Produce a side-by-side comparison of multiple fit_mfrm() results using information criteria, log-likelihood, and parameter counts. When exactly two models are supplied and the current conservative nesting review passes, a likelihood-ratio test is included.

Usage

```
compare_mfrm(..., labels = NULL, warn_constraints = TRUE, nested = FALSE)
```

Arguments

...	Two or more mfrm_fit objects to compare.
labels	Optional character vector of labels for each model. If NULL, labels are generated from model/method combinations.
warn_constraints	Logical. If TRUE (the default), emit a warning when models use different centering constraints (noncenter_facet or dummy_facets), which can make information-criterion comparisons misleading.
nested	Logical. Set to TRUE only when the supplied models are known to be nested and fitted with the same likelihood basis on the same observations. The default is FALSE, in which case no likelihood-ratio test is reported. When TRUE, the function still runs a conservative structural nesting review and computes the LRT only for supported nesting patterns.

Details

Models should be fit to the **same data** (same rows, same person/facet columns) for the comparison to be meaningful. The function checks that observation counts match and warns otherwise.

Information-criterion ranking is reported only when all candidate models use the package's MML estimation path, analyze the same observations, and converge successfully. Raw AIC and BIC values are still shown for each model, but Delta_*, weights, and preferred-model summaries are suppressed when the likelihood basis is not comparable enough for primary reporting. The comparison table records both row count (nobs) and the sample-size basis used for the BIC penalty (ICSampleSize, ICSampleSizeBasis); for weighted fits this is the sum of weights rather than the number of rows.

Nesting: Two models are *nested* when one is a special case of the other obtained by imposing equality constraints. The most common nesting in MFRM is RSM (shared thresholds) inside PCM (item-specific thresholds). Models that differ only in estimation method (MML vs JML) on the same specification are not nested in the usual sense—use information criteria rather than LRT for that comparison.

In the **current** mfrmr **model space**, the automatic nesting review is intentionally conservative. It currently supports two fixed-effect restrictions under shared data and shared constraints:

- RSM nested inside PCM when the PCM fit has an explicit step_facet;
- same-family additive-vs-interaction comparisons when the smaller fit's facet_interactions set is a subset of the larger fit's set.

Cross-method comparisons, comparisons that change anchors/dummying/centering, and same-family comparisons that do not add fixed interaction terms are not automatically promoted to LRT claims.

The **likelihood-ratio test (LRT)** is reported only when exactly two models are supplied, nested = TRUE, the structural nesting review passes, and the difference in the number of parameters is positive:

$$\Lambda = -2(\ell_{\text{restricted}} - \ell_{\text{full}}) \sim \chi^2_{\Delta p}$$

The LRT is asymptotically valid when models are nested and the data are independent. With small samples or boundary conditions (e.g., variance components near zero), treat p-values as approximate.

Value

An object of class mfrm_comparison (named list) with:

- table: data.frame of model-level statistics (LogLik, AIC, BIC, Delta_AIC, AkaikeWeight, Delta_BIC, BICWeight, npar, nobs, WeightedN, ICSampleSize, ICSampleSizeBasis, Model, Method, Converged, ICComparable).
- lrt: data.frame with likelihood-ratio test result (only when two models are supplied and nested = TRUE). Contains ChiSq, df, p_value.
- evidence_ratios: data.frame of pairwise Akaike-weight ratios (Model1, Model2, EvidenceRatio). NULL when weights cannot be computed.
- preferred: named list with the preferred model label by each criterion.
- comparison_basis: list describing whether IC and LRT comparisons were considered comparable. Includes a conservative nesting_review plus lrt_status / lrt_reason so withheld LRTs are explicit rather than silently absent.

Information-criterion diagnostics

In addition to raw AIC and BIC values, the function computes:

- **Delta_AIC / Delta_BIC**: difference from the best (minimum) value. A Delta < 2 is typically considered negligible; 4–7 suggests moderate evidence; > 10 indicates strong evidence against the higher-scoring model (Burnham & Anderson, 2002).
- **AkaikeWeight / BICWeight**: model probabilities derived from exp(-0.5 * Delta), normalised across the candidate set. An Akaike weight of 0.90 means the model has a 90\ being the best in the candidate set.
- **Evidence ratios**: pairwise ratios of Akaike weights, quantifying the relative evidence for one model over another (e.g., an evidence ratio of 5 means the preferred model is 5 times more likely).

AIC penalises complexity less than BIC; when they disagree, AIC favours the more complex model and BIC the simpler one.

What this comparison means

`compare_mfrm()` is a same-basis model-comparison helper. Its strongest claims apply only when the models were fit to the same response data, under a compatible likelihood basis, and with compatible constraint structure.

What this comparison does not justify

- Do not treat AIC/BIC differences as primary evidence when `table$ICComparable` is `FALSE`.
- Do not interpret the LRT unless `nested = TRUE` and the structural nesting review in `comparison_basis$nesting_revie` passes.
- Same-family additive-vs-interaction fits are considered nested only when all other structural settings match and the smaller model's `facet_interactions` set is a subset of the larger model's set.
- Do not assume that `nested = TRUE` overrides the package's conservative nesting boundary; unsupported relations remain unsupported.
- Do not compare models fit to different datasets, different score codings, or materially different constraint systems as if they were commensurate.

Interpreting output

- Lower AIC/BIC values indicate better parsimony-accuracy trade-off only when `table$ICComparable` is `TRUE`.
- A significant LRT p-value suggests the more complex model provides a meaningfully better fit only when the nesting assumption truly holds.
- `preferred` indicates the model preferred by each criterion.
- `evidence_ratios` gives pairwise Akaike-weight ratios (returned only when Akaike weights can be computed for at least two models).
- When comparing more than two models, interpret evidence ratios cautiously—they do not adjust for multiple comparisons.

How to read the main outputs

- `table`: first-pass comparison table; start with `ICComparable`, `Model`, `Method`, `AIC`, and `BIC`.
- `comparison_basis`: records whether IC and LRT claims are defensible for the supplied models. Inspect `comparison_basis$nesting_review$relation` and `reason` before reading any LRT output.
- `lrt`: nested-model test summary, present only when the requested and reviewed conditions are met.
- `preferred`: candidate preferred by each criterion when those summaries are available.

Recommended next step

Inspect `comparison_basis` before writing conclusions. If comparability is weak, treat the result as descriptive and revise the model setup (for example, explicit `step_facet`, common data, or common constraints) before using IC or LRT results in reporting.

Typical workflow

1. Fit two models with `fit_mfrm()` (e.g., RSM and PCM).
2. Compare with `compare_mfrm(fit_rsm, fit_pcm)`.
3. Inspect `summary(comparison)` for AIC/BIC diagnostics and, when appropriate, an LRT.

References

- Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference: A practical information-theoretic approach* (2nd ed.). Springer.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716-723.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics, 6*(2), 461-464.

See Also

`fit_mfrm()`, `diagnose_mfrm()`

Examples

```
toy <- load_mfrmr_data("example_core")

fit_rsm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "RSM", quad_points = 7, maxit = 30)
fit_pcm <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                    method = "MML", model = "PCM",
                    step_facet = "Criterion", quad_points = 7, maxit = 30)
comp <- compare_mfrm(fit_rsm, fit_pcm, labels = c("RSM", "PCM"))
comp$table
comp$evidence_ratios
```

---

compatibility_alias_table *List retained compatibility aliases and preferred names*

Description

List retained compatibility aliases and preferred names

Usage

```
compatibility_alias_table(
  scope = c("all", "functions", "arguments", "fields", "columns", "plot_metrics")
)
```

Arguments

scope	Which alias surface to return: "all", "functions", "arguments", "fields", "columns", or "plot_metrics".

Details

This helper is a compact public registry of the compatibility aliases that mfrmr intentionally keeps visible for older scripts and downstream handoffs. It is meant to answer two questions quickly:

1. Which old names are still accepted?
2. Which package-native names should new code use instead?

Internal soft-deprecated helpers are deliberately excluded here. This table is only for retained user-facing aliases that remain part of the public surface.

Value

A data.frame with one row per retained alias and columns:

- Alias
- PreferredName
- Surface
- Lifecycle
- RetainedFor
- RemovalPlan
- Notes

Typical workflow

1. Call compatibility_alias_table() when reading older scripts or reports.
2. Use PreferredName when updating older analysis code.
3. Prefer the package-native name in all new outputs and scripts.

See Also

mfrmr_compatibility_layer, run_mfrm_facets(), analyze_dff(), reporting_checklist(), fair_average_table(), plot_fair_average()

Examples

```
compatibility_alias_table()
compatibility_alias_table("functions")
compatibility_alias_table("fields")
compatibility_alias_table("columns")
```

compute_facet_design_effect *Compute Kish design effects for each facet*

Description

Combines per-facet average cluster size with ICC estimates to return the Kish (1965) design effect `Deff = 1 + (m - 1) * rho`, where `m` is the average number of observations per facet element and `rho` is the ICC.

Usage

```
compute_facet_design_effect(
  data,
  facets,
  icc_table = NULL,
  score = NULL,
  person = NULL
)
```

Arguments

`data`	Data frame in long format.
`facets`	Character vector of facet column names.
`icc_table`	Output from `compute_facet_icc()` (optional; will be computed on the fly when `NULL`).
`score`	Score column name; required when `icc_table` is `NULL`.
`person`	Person column; passed through to compute_facet_icc().

Value

A data.frame of class `mfrm_facet_design_effect` with columns `Facet`, `AvgClusterSize`, `ICC`, `DesignEffect`, and `EffectiveN`.

Interpreting output

- `Deff = 1`: facet behaves like simple random sampling; no clustering-induced variance inflation.
- `Deff > 1`: variance of the mean estimate is inflated by a factor of `Deff` relative to SRS. `EffectiveN = N / Deff` is the sample size one would need under SRS to achieve the same precision. For rater-mediated designs, `Deff` well above 1 on the Rater facet means rater-level clustering is noticeable; consider whether rater generalisation is warranted.
- Reported `ICC` is pulled from `icc_table$ICC` (the variance share); interpretation is the same as in `compute_facet_icc()`.

Typical workflow

1. Run `compute_facet_icc()` to get the variance-component shares.
2. Feed the result and the data into `compute_facet_design_effect(data, facets, icc_table = icc)`.
3. Use `Deff` as part of the Methods discussion when generalising over raters or sites. Large `Deff` values argue for reporting robust SEs or moving to a hierarchical model.

References

Kish, L. (1965). *Survey Sampling*. New York: Wiley.

Park, I., & Lee, H. (2001). The design effect: Do we know all about it? In *Proceedings of the American Statistical Association, Survey Research Methods Section* (pp. 143-148).

See Also

`compute_facet_icc()`, `analyze_hierarchical_structure()`.

Examples

```
toy <- load_mfrmr_data("example_core")
if (requireNamespace("lme4", quietly = TRUE)) {
  icc <- compute_facet_icc(toy, facets = c("Rater", "Criterion"),
                           score = "Score", person = "Person")
  deff <- compute_facet_design_effect(toy,
                                      facets = c("Rater", "Criterion"),
                                      icc_table = icc)
  print(deff)
  # Large DesignEffect -> modest EffectiveN relative to raw N.
}
```

---

compute_facet_icc *Compute intra-class correlations for each facet*

Description

Fits a random-effects variance-components model `Score ~ 1 + (1 | Person) + (1 | Facet1) + (1 | Facet2) + ...` using `lme4::lmer` (in `Suggests`) and returns the proportion of observed score variance attributable to each facet. This is a descriptive summary complementary to the Rasch-metric rater separation/reliability reported elsewhere.

Usage

```
compute_facet_icc(
  data,
  facets,
  score,
  person = NULL,
  reml = TRUE,
  ci_method = c("none", "profile", "boot"),
  ci_level = 0.95,
  ci_boot_reps = 1000L,
  ci_boot_seed = NULL,
  ci_boot_parallel = c("no", "multicore", "snow"),
  ci_boot_ncpus = 1L
)
```

Arguments

data	Data frame in long format.
facets	Character vector of facet column names.
score	Name of the score column.
person	Optional person column. If supplied it is added as a separate random intercept so Person-level variance is partitioned out.
reml	Logical; whether to fit with REML. Default TRUE.
ci_method	Confidence-interval method for the ICC column. One of "none" (default, point estimate only), "profile" (a **first-order approximation**: marginal likelihood-profile bounds for each variance-component SD via lme4::confint.merMod() with method = "profile", squared to variances, then plugged into the ICC ratio while holding the other components at their point estimate; fast and deterministic, the default recommendation for reporting), or "boot" (parametric bootstrap via lme4::bootMer(); slower but robust to non-normal ICC sampling distributions because each bootstrap replicate resamples the full variance decomposition jointly).
ci_level	Confidence level when ci_method != "none"; default 0.95. Koo & Li (2016) recommend banding the CI rather than the point estimate when classifying reliability as Poor / Moderate / Good / Excellent.
ci_boot_reps	Number of bootstrap replicates used when ci_method = "boot". Default 1000.
ci_boot_seed	Optional integer seed for the bootstrap path (NULL leaves the RNG state untouched).
ci_boot_parallel	Parallelisation strategy for the parametric-bootstrap CI path, passed through to lme4::bootMer(): "no" (default), "multicore" (POSIX mclapply), or "snow" (PSOCK cluster). "multicore" does nothing on Windows and falls back to serial; in that case use "snow" with parallel::makeCluster() in scope.
ci_boot_ncpus	Number of CPUs to use for the parallel bootstrap path (ignored when ci_boot_parallel = "no"). The per-replicate progress bar is suppressed under parallel execution because worker processes cannot push updates to the parent's cli console.

Value

A data.frame of class `mfrm_facet_icc` with one row per variance component (including a `"Residual"` row) and columns:

- `Facet`: the grouping factor name (or `"Residual"`).
- `Variance`: REML variance estimate.
- `ICC`: variance share (`Variance / sum(Variance)`), in `[0, 1]`.
- `Interpretation`: band label according to the facet's scale.
- `InterpretationScale`: `"Koo-Li reliability"` for the person facet, `"Variance share"` for others.
- `ICC_CI_Lower` / `ICC_CI_Upper` / `ICC_CI_Level` / `ICC_CI_Method`: CI bounds, level, and method (populated when `ci_method != "none"`; `NA_real_` otherwise).
- `ICC_CI_NReps`: bootstrap replicate count when `ci_method = "boot"` (absent otherwise).

Interpreting output

The `Interpretation` column uses **two scales** so the same numeric ICC reads correctly for each facet role:

- For the `person` facet, higher ICC = better. Koo & Li (2016, p. 161) bands are applied: `< 0.5` Poor, `[0.5, 0.75]` Moderate, `(0.75, 0.9]` Good, `> 0.9` Excellent. The strict > boundary at 0.9 follows Koo & Li's wording "values greater than 0.90 indicate excellent reliability" (so an ICC of exactly 0.9 reads as Good).
- For non-person facets (Rater, Criterion, Task, Region, ...) the same numeric value is a **variance share**: how much of the total observed score variance sits at that facet. The bands used here are different (`Trivial share` < 0.05, `Small share` < 0.15, `Moderate share` < 0.30, `Large share` >= 0.30), and a large rater share is generally *bad* news (raters disagree about averages), not good news.

The `InterpretationScale` column explicitly records which scale applies to each row, so downstream reporting does not confuse the two. FACETS (Linacre, 2026) reports rater separation/reliability on the Rasch metric instead of an ICC; mfrmr surfaces both, with the Rasch-metric version in `diagnostics$reliability` and this variance-share view here.

Note: Koo & Li (2016) recommend applying the reliability bands to the **95% confidence interval** of the ICC rather than to the point estimate alone. Set `ci_method = "profile"` (default `"none"`) to obtain likelihood-profile CI bounds alongside the point estimate, or `ci_method = "boot"` for a parametric bootstrap with `ci_boot_reps` replicates. The returned data frame gains `ICC_CI_Lower` / `ICC_CI_Upper` columns so downstream reporting can apply the band to the CI rather than the point estimate. The `Interpretation` column still uses the point estimate so callers who want CI-aware banding can implement it externally from the supplied bounds.

Typical workflow

1. Fit the MFRM model with `fit_mfrm()` for the Rasch-metric separation/reliability.
2. Call `compute_facet_icc(data, facets, score, person)` to get the complementary variance-share summary.
3. Feed into `compute_facet_design_effect()` to convert ICCs and average cluster sizes into Kish (1965) design effects.

References

Koo, T. K., & Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine, 15*(2), 155-163.

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1-48.

See Also

`compute_facet_design_effect()`, `analyze_hierarchical_structure()`, `detect_facet_nesting()`, `facet_small_sample_review()`.

Examples

```
toy <- load_mfrmr_data("example_core")
if (requireNamespace("lme4", quietly = TRUE)) {
  icc <- compute_facet_icc(toy, facets = c("Rater", "Criterion"),
                           score = "Score", person = "Person")
  print(icc)
  # Look for:
  # - Person ICC reads as Koo & Li (2016) reliability: < 0.5 poor,
  #   0.5-0.75 moderate, 0.75-0.9 good, > 0.9 excellent.
  # - Rater / Criterion ICC reads as variance share, NOT reliability;
  #   here SMALL values are desirable (raters / items agree), and
  #   shares > 0.10 hint at meaningful systematic facet differences.
  # - `Interpretation` summarises the variance-share band the helper
  #   has assigned to each row.
}
```

---

compute_information *Compute design-weighted precision curves for ordered many-facet fits*

Description

Calculates design-weighted score-variance curves across the latent trait (theta) for a fitted ordered-category `RSM`, `PCM`, or bounded `GPCM` model. Returns both an overall precision curve (`$tif`) and per-facet-level contribution curves (`$iif`) based on the realized observation pattern.

Usage

```
compute_information(fit, theta_range = c(-6, 6), theta_points = 201L)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`theta_range`	Numeric vector of length 2 giving the range of theta values. Default `c(-6, 6)`.
`theta_points`	Integer number of points at which to evaluate information. Default `201`.

Details

For RSM / PCM, the score variance at theta for one observed design cell is:

$$I(\theta) = \sum_{k=0}^{K} P_k(\theta)\,(k - E(\theta))^2$$

where P_k is the category probability and $E(\theta)$ is the expected score at theta. In mfrmr, these cell-level variances are then aggregated with weights taken from the realized observation counts in fit$prep$data.

The resulting total curve is therefore a design-weighted precision screen rather than a pure textbook test-information function for an abstract fixed item set. The associated standard error summary is still $SE(\theta) = 1/\sqrt{I(\theta)}$ for positive information values.

In an ordered Rasch-family model, category discrimination is fixed at 1, so this score-variance representation is the natural conditional information identity rather than a separate approximation. For binary data it reduces to the familiar $p(\theta)\{1 - p(\theta)\}$ form. For PCM, the package evaluates each observed design cell using the threshold vector associated with that cell's realized step_facet level. For bounded GPCM, the same design-weighted score variance is scaled by the squared discrimination attached to the realized slope_facet level, which is the $a_j^2 \cdot \mathrm{Var}(T \mid \theta)$ item-information identity that Muraki (1993, Equation 10) derives by applying Samejima's (1974) polytomous information formula to the GPCM kernel of Muraki (1992).

Value

An object of class mfrm_information (named list) with:

- tif: tibble with columns Theta, Information, SE. The Information column stores the design-weighted precision value.
- iif: tibble with columns Theta, Facet, Level, Information, and Exposure. Here too, Information stores a design-weighted contribution value retained under that column name for compatibility.
- theta_range: the evaluated theta range.

What tif and iif mean here

In mfrmr, this helper supports ordered-category RSM, PCM, and the current bounded GPCM fit. The total curve ($tif) is the sum of design-weighted cell contributions across all non-person facet levels in the fitted model. The facet-level contribution curves ($iif) keep those weighted contributions separated, so you can see which observed rater levels, criteria, or other facet levels are driving precision at different parts of the scale. For PCM, step-facet-specific thresholds are respected when each observed design cell is evaluated. For bounded GPCM, those same cell-level variances are additionally scaled by the squared discrimination associated with the realized slope_facet level.

What this quantity does not justify

- It is not a textbook many-facet test-information function for an abstract fixed item set.
- It should not be used as if it were design-free evidence about a form's precision independent of the realized observation pattern.
- It does not currently extend beyond the ordered-category RSM / PCM / bounded GPCM family implemented by fit_mfrm().

When to use this

Use `compute_information()` when you want a design-weighted precision screen for an `RSM`, `PCM`, or bounded `GPCM` fit along the latent continuum. In practice:

- start with the total precision curve for overall targeting across the realized observation pattern
- inspect facet-level contribution curves when you want to see which raters, criteria, or other facet levels account for more of that design-weighted precision
- widen `theta_range` if you expect extreme measures and want to inspect the tails explicitly

Choosing the theta grid

The defaults (`theta_range = c(-6, 6)`, `theta_points = 201`) work well for routine inspection. Expand the range if person or facet measures extend into the tails, and increase `theta_points` only when you need a smoother grid for reporting or custom graphics.

References

The ordered-category probability structures come from Andrich's `RSM` formulation and Masters' `PCM`. The bounded `GPCM` information identity $a_j^2 \cdot \mathrm{Var}(T \mid \theta)$ is derived in Muraki (1993, Equation 10) by applying Samejima's (1974) general polytomous information formula $I_j(\theta) = \sum_k P_{jk}(\theta)[-\partial^2 \ln P_{jk}/\partial\theta^2]$ to the GPCM probability kernel of Muraki (1992). For the integer scoring function $T_k = k$ used by `mfrmr`, this reduces to $a_j^2 \cdot \mathrm{Var}(K \mid \theta)$. In `mfrmr`, those formulas are applied to the realized many-facet observation design, so the output should be read as a design-weighted precision summary rather than as a design-free abstract test function.

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176. doi:10.1177/014662169201600206 (See Equations 6, 10, and 13 for the probability kernel and the $\partial P_k/\partial\theta = a_j P_k(k - E[K])$ derivative used by all GPCM helpers in `mfrmr`.)
- Muraki, E. (1993). *Information functions of the generalized partial credit model*. Applied Psychological Measurement, 17(4), 351-363. doi:10.1177/014662169301700403 (Equation 10 derives the item information function for the GPCM, $I_j(\theta) = D^2 a_j^2 \mathrm{Var}(T \mid \theta)$, by applying Samejima's (1974) polytomous information formula to the GPCM kernel; this is the canonical reference for `compute_information()` under bounded `GPCM`.)
- Samejima, F. (1974). *Normal ogive model on the continuous response level in the multidimensional latent space*. Psychometrika, 39, 111-121. (Source for the general polytomous information formula that Muraki 1993 specializes to the GPCM.)

Interpreting output

- `$tif`: design-weighted precision curve data with theta, Information, and SE.
- `$iif`: design-weighted facet-level contribution curves for the fitted non-person facets.
- Higher information implies more precise measurement at that theta.

- SE is inversely related to information.
- Peaks in the total curve show the trait region where the realized calibration is most informative.
- Facet-level curves help explain *which observed facet levels* contribute to those peaks; they are not standalone item-information curves and should be read as design contributions.

How to read the main columns

- `Theta`: point on the latent continuum where the curve is evaluated.
- `Information`: design-weighted precision value at that theta.
- `SE`: approximate `1 / sqrt(Information)` summary for positive values.
- `Exposure`: total realized observation weight contributing to a facet-level curve in `$iif`.

Recommended next step

Compare the precision peak with person/facet locations from a Wright map or related diagnostics. If you need to decide how strongly SE/CI language can be used in reporting, follow with `precision_review_report()`.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Run `compute_information(fit)`.
3. Plot with `plot_information(info, type = "tif")`.
4. If needed, inspect facet contributions with `plot_information(info, type = "iif", facet = "Rater")`.

See Also

`fit_mfrm()`, `plot_information()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
info <- compute_information(fit)
head(info$tif)
info$tif$Theta[which.max(info$tif$Information)]
```

compute_person_fit_indices *Person fit indices: lz and Snijders-corrected lz**

Description

Computes person-level fit statistics for an MFRM bundle, extending the Infit / Outfit / ZSTD columns that `diagnose_mfrm()$measures` already exposes with the standardized log-likelihood `lz` and, when justified by the person-estimation method, Snijders' `lz*`.

Usage

```
compute_person_fit_indices(diagnostics, fit = NULL)
```

Arguments

`diagnostics`	Output from `diagnose_mfrm()`.
`fit`	Optional `mfrm_fit` from `fit_mfrm()`. Required to decide whether the person estimates are JML/fixed-effect estimates for which the Snijders (2001) correction is computed. MML/EAP person scores return `NA` for `lz_star` with an explanatory status.

Value

A data frame of class `mfrm_person_fit_indices` with one row per Person and columns:

`Person` Person ID.

`N` Number of contributing response opportunities.

`LogLik` Sum of log P(X = x | theta) under the fitted model. Computed from the per-observation category probability `PrObserved` (the model probability of the observed category), not from a Gaussian residual approximation.

`lz` Drasgow et al. (1985) standardized log-likelihood, in its proper polytomous form.

`lz_star` Snijders-corrected `lz*` when the source fit used JML/fixed-effect person estimates, conditioning on the fitted non-person calibration, and the diagnostics include the required derivative terms; otherwise `NA`.

`lz_star_status` Status string for `lz_star`, such as `"computed_jml_conditional_calibration"`, `"fit_required"`, `"not_applicable_eap"`, or `"insufficient_information"`.

`lz_star_c` Estimated Snijders projection coefficient `c_n` for each person, when available.

`lz_star_variance` Corrected variance denominator used for `lz_star`, when available.

`lz_flag_5pct`, `lz_flag_1pct` Logical flags for practical two-sided `lz` thresholds of `|z| > 1.96` and `|z| > 2.58`.

`lz_star_flag_5pct`, `lz_star_flag_1pct` The same flags for `lz_star`, returned as `FALSE` when `lz_star` is unavailable.

`ReportIndex`, `ReportValue`, `ReportFlagLevel`, `ReportFlag`, `ReviewStatus`, `ReviewReason`, `ReportCaveat`
: Compact reporting columns. `ReportIndex` prefers `lz_star` when the Snijders correction was computed; otherwise it falls back to `lz` with an explicit caveat.

Under the conditional-independence assumption of the MFRM, `lz` is asymptotically standard normal. Practical reporting thresholds: |lz| > 1.96 flags a person at the 5% level; |lz| > 2.58 at the 1% level. When `lz_star_status == "computed_jml_conditional_calibration"`, `lz_star` applies Snijders' estimated-ability correction for JML person estimates, conditional on the fitted non-person parameters. This does not propagate non-person calibration uncertainty. For MML/EAP person scores, use `lz` with its documented caveat rather than treating EAP scores as if they satisfied the Snijders estimating equation.

Note: this implementation reads the model category probabilities directly from the diagnostics bundle. Earlier mfrmr releases used a Gaussian-residual approximation $\log P(X = x) \approx -\frac{1}{2}(R^2/V) - \frac{1}{2}\log(2\pi V)$ as a stand-in for $\log P$, which overstated the per-item variance of $\log P$ for polytomous items, shrinking the reported `lz` toward zero. Numerical `lz` values are therefore not directly comparable across mfrmr releases; treat the values returned here as the polytomous statistic and re-evaluate any historical `|lz| > 1.96` flagging that was based on the earlier approximation.

References

- Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*(1), 67-86.
- Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika, 66*(3), 331-342.
- Magis, D., Raiche, G., & Beland, S. (2012). A didactic presentation of Snijders's lz* index of person fit with emphasis on response model selection and ability estimation. *Journal of Educational and Behavioral Statistics, 37*(1), 57-81.
- Sinharay, S. (2016). Asymptotically correct standardization of person-fit statistics beyond dichotomous items. *Psychometrika, 81*(4), 992-1013.

See Also

`diagnose_mfrm()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none",
                      diagnostic_mode = "legacy")
pf <- compute_person_fit_indices(diag, fit = fit)
head(pf)
summary(pf)
# Look for: |lz| > 1.96 (5% level) flags a person whose response
#   pattern is statistically inconsistent with the model; > 2.58 is
#   a 1% flag. lz_star is populated for JML/fixed-effect person
#   estimates and left NA for MML/EAP estimates. Use ReportIndex /
#   ReviewStatus for a compact report-ready reading.
```

`data_quality_report` *Build a data quality summary report (preferred alias)*

Description

Build a data quality summary report (preferred alias)

Usage

```
data_quality_report(
  fit,
  data = NULL,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  min_category_count = 10,
  dominant_category_cutoff = 0.95,
  include_fixed = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`data`	Optional raw data frame used for row-level review.
`person`	Optional person column name in `data`.
`facets`	Optional facet column names in `data`.
`score`	Optional score column name in `data`.
`weight`	Optional weight column name in `data`.
`min_category_count`	Minimum raw or weighted count used to label a non-zero facet-level score category as sparse. Default `10`.
`dominant_category_cutoff`	Proportion in `(0, 1]` used to flag a facet level whose responses are dominated by one score category. Default `0.95`.
`include_fixed`	If `TRUE`, include a legacy-compatible fixed-width text block.

Details

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_data_quality` (`type = "dashboard"`, `"quality_flags"`, `"row_review"`, `"category_counts"`, `"score_support"`, `"facet_category_usage"`, `"facet_response_patterns"`, `"score_map"`, `"missing_rows"`).

Value

A named list with data-quality report components. Class: `mfrm_data_quality`.

Interpreting output

- `summary`: retained/dropped row overview.
- `quality_overview`: area-level QC status for rows, score support, facet-category use, and design matching.
- `quality_flags`: prioritized QC flags with counts and recommended next actions. This is not an item/person/rater table.
- `row_review`: reason-level breakdown for data issues.
- `category_counts`: post-filter category usage, including retained zero-count score-support categories.
- `score_support_review`: quick view of zero-count boundary/intermediate categories and their threshold-functioning caveats.
- `category_usage_by_facet`: facet-level category counts over the retained score support.
- `category_usage_summary`: per-facet-level zero/sparse category summary.
- `facet_response_patterns`: facet-level response-pattern summaries, including single-category and dominant-category use.
- `caveats`: user-facing score-support warnings, including cases where non-consecutive original labels such as `1, 2, 4, 5` were recoded because `keep_original = FALSE`.
- `score_map`: original-to-internal score mapping used when labels are recoded.
- `unknown_elements`: facet levels in raw data but not in fitted design.

Typical workflow

1. Run `data_quality_report(...)` with raw data.
2. Check `summary(out)` and `plot(out, type = "dashboard")`, then inspect `quality_flags`, score-support, score-map, facet-response-pattern, and missing/unknown element sections as needed.
3. Resolve missing values, score-support gaps, and sparse categories before final estimation/reporting.

See Also

`fit_mfrm()`, `describe_mfrm_data()`, `specifications_report()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- data_quality_report(
  fit, data = toy, person = "Person",
  facets = c("Rater", "Criterion"), score = "Score"
)
summary(out)
p_dq <- plot(out, draw = FALSE)
p_dq$data$plot
```

describe_mfrm_data *Summarize MFRM input data (TAM-style descriptive snapshot)*

Description

Summarize MFRM input data (TAM-style descriptive snapshot)

Usage

```
describe_mfrm_data(
  data,
  person,
  facets,
  score,
  weight = NULL,
  rating_min = NULL,
  rating_max = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  include_person_facet = FALSE,
  include_agreement = TRUE,
  rater_facet = NULL,
  context_facets = NULL,
  agreement_top_n = NULL
)
```

Arguments

data	A data.frame in long format (one row per rating event).
person	Column name for person IDs.
facets	Character vector of facet column names.
score	Column name for observed score.
weight	Optional weight/frequency column name.
rating_min	Optional minimum category value. Supply with rating_max to retain unused boundary categories in the intended score support.
rating_max	Optional maximum category value. Supply with rating_min to retain unused boundary categories in the intended score support.
keep_original	Keep original category values. Use this with rating_min / rating_max when the intended scale has unused intermediate categories such as 1, 2, 4, 5 on a 1-5 scale.
missing_codes	Optional. NULL (default) is a no-op; TRUE or "default" activates the FACETS / SPSS / SAS convention (c("99", "999", "-1", "N", "NA", "n/a", ".", "")); supply a character vector for a custom code set. Replacement counts are returned in the missing_recoding component when supported by the calling helper. See recode_missing_codes() for the standalone version.

include_person_facet
: If TRUE, include person-level rows in facet_level_summary.

include_agreement
: If TRUE, include an observed-score inter-rater agreement bundle (summary/pairs/settings) in the output.

rater_facet
: Optional rater facet name used for agreement summaries. If NULL, inferred from facet names.

context_facets
: Optional facets used to define matched contexts for agreement. If NULL, all remaining facets (including Person) are used.

agreement_top_n
: Optional maximum number of agreement pair rows.

Details

This function provides a compact descriptive bundle similar to the pre-fit summaries commonly checked in TAM workflows: sample size, score distribution, per-facet coverage, and linkage counts. psych::describe() is used for numeric descriptives of score and weight.

Key data-quality checks to perform before fitting:

- *Sparse categories*: any score category with fewer than 10 weighted observations may produce unstable threshold estimates (Linacre, 2002). Consider collapsing adjacent categories.
- *Unlinked elements*: if a facet level has zero overlap with one or more levels of another facet, the design is disconnected and parameters cannot be placed on a common scale. Check linkage_summary for low connectivity.
- *Extreme scores*: persons or facet levels with all-minimum or all-maximum scores yield infinite logit estimates under JML; they are handled via Bayesian shrinkage under MML.

Value

A list of class mfrm_data_description with:

- overview: one-row run-level summary
- missing_by_column: missing counts in selected input columns
- missing_rate_summary: per-column missingness rate summary (one row per input column, with raw and proportion-of-N columns)
- score_descriptives: output from psych::describe() for score
- weight_descriptives: output from psych::describe() for weight
- score_distribution: weighted and raw score frequencies over the prepared score support. Unused boundary categories are retained when the rating range was supplied explicitly; unused intermediate categories require keep_original = TRUE.
- facet_level_summary: per-level usage and score summaries
- facet_crosstabs: pairwise observation-count crosstabs between non-person facets (named list keyed "facetA__facetB"); used by summary(ds)$design_links to flag sparse / disconnected facet-pair coverage
- linkage_summary: person-facet connectivity diagnostics

- `agreement`: observed-score inter-rater agreement bundle
- `row_retention`: row counts before and after preparation filters
- `preparation_notes`: structured notes for row drops, ID trimming, and design conditions detected during preparation
- `score_support`: minimal prepared score-support metadata used by `summary(ds)$caveats`

Interpreting output

Recommended order:

- `overview`: confirms sample size, facet count, and category span. The `MinWeightedN` column shows the smallest weighted observation count across all facet levels; values below 30 may lead to unstable parameter estimates.
- `missing_by_column`: identifies immediate data-quality risks. Any non-zero count warrants investigation before fitting.
- `score_distribution`: checks sparse/unused score categories. Balanced usage across categories is ideal; heavily skewed distributions may compress the measurement range.
- `facet_level_summary` and `linkage_summary`: checks per-level support and person-facet connectivity. Low linkage ratios indicate sparse or disconnected design blocks.
- `agreement`: optional observed inter-rater consistency summary (exact agreement, correlation, mean differences per rater pair).

Typical workflow

1. Run `describe_mfrm_data()` on long-format input.
2. Review `summary(ds)` and `plot(ds, ...)`.
3. Resolve missingness/sparsity issues before `fit_mfrm()`.

See Also

`fit_mfrm()`, `review_mfrm_anchors()`

Examples

```
toy <- load_mfrmr_data("example_core")
ds <- describe_mfrm_data(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score"
)
s_ds <- summary(ds)
s_ds$overview
p_ds <- plot(ds, draw = FALSE)
p_ds$data$plot
```

`detect_anchor_drift` *Detect anchor drift across multiple calibrations*

Description

Compares facet estimates across two or more calibration waves to identify elements whose difficulty/severity has shifted beyond acceptable thresholds. Useful for monitoring rater drift over time or checking the stability of item banks.

Usage

```
detect_anchor_drift(
  fits,
  facets = NULL,
  drift_threshold = 0.5,
  flag_se_ratio = 2,
  reference = 1L,
  include_person = FALSE
)

## S3 method for class 'mfrm_anchor_drift'
print(x, ...)

## S3 method for class 'mfrm_anchor_drift'
summary(object, ...)

## S3 method for class 'summary.mfrm_anchor_drift'
print(x, ...)
```

Arguments

`fits`	Named list of `mfrm_fit` objects (e.g., `list(Year1 = fit1, Year2 = fit2)`).
`facets`	Character vector of facets to compare (default: all non-Person facets).
`drift_threshold`	Absolute drift threshold for flagging (logits, default 0.5).
`flag_se_ratio`	Drift/SE ratio threshold for flagging (default 2.0).
`reference`	Index or name of the reference fit (default: first).
`include_person`	Include person estimates in comparison.
`x`	An `mfrm_anchor_drift` object.
`...`	Ignored.
`object`	An `mfrm_anchor_drift` object (for `summary`).

Details

For each non-reference wave, the function extracts facet-level estimates using `make_anchor_table()` and computes the element-by-element difference against the reference wave. Standard errors are obtained from `diagnose_mfrm()` applied to each fit. Only elements common to both the reference and a comparison wave are included. Before reporting drift, the function removes the weighted common-element link offset between the two waves so that `Drift` represents residual instability rather than the overall shift between calibrations. The function also records how many common elements survive the screening step within each linking facet and treats fewer than 5 retained common elements per facet as thin support.

An element is **flagged** when either condition is met:

$$|\Delta_e| > \texttt{drift\_threshold}$$

$$|\Delta_e / SE_{\Delta_e}| > \texttt{flag\_se\_ratio}$$

The dual-criterion approach guards against flagging elements with large but imprecise estimates, and against missing small but precisely estimated shifts.

When `facets` is `NULL`, all non-Person facets are compared. Providing a subset (e.g., `facets = "Criterion"`) restricts comparison to those facets only.

Value

Object of class `mfrm_anchor_drift` with components:

drift_table Tibble of element-level drift statistics.

summary Drift summary aggregated by facet and wave.

common_elements Tibble of pairwise common-element counts.

common_vs_reference Tibble of common-element counts between each wave and the reference wave (i.e., which elements remain comparable across the entire chain).

n_common_all_waves Integer count of elements that are common across every wave; used by `summary()` to gauge how robust the chain is to chained linking error.

common_by_facet Tibble of retained common-element counts by facet.

config List of analysis configuration.

Which function should I use?

- Use `anchor_to_baseline()` when your starting point is raw new data plus a single baseline fit.
- Use `detect_anchor_drift()` when you already have multiple fitted waves and want a reference-versus-wave comparison.
- Use `build_equating_chain()` when the waves form a sequence and you need cumulative linking offsets.

Interpreting output

- `$drift_table`: one row per element x wave combination, with columns `Facet`, `Level`, `Wave`, `Ref_Est`, `Wave_Est`, `LinkOffset`, `Drift`, `SE_Ref`, `SE_Wave`, `SE`, `Drift_SE_Ratio`, `LinkSupportAdequate`, and `Flag`. Large drift signals instability after alignment to the common-element link.
- `$summary`: aggregated statistics by facet and wave: number of elements, mean/max absolute drift, and count of flagged elements.
- `$common_elements`: pairwise common-element counts in tidy table form. Small overlap weakens the comparison and results should be interpreted cautiously.
- `$common_by_facet`: retained common-element counts by linking facet for each reference-vs-wave comparison. `LinkSupportAdequate = FALSE` means the link rests on fewer than 5 retained common elements in at least one facet.
- `$config`: records the analysis parameters for reproducibility.
- A practical reading order is `summary(drift)` first, then `drift$drift_table`, then `drift$common_by_facet` if overlap looks thin.

Typical workflow

1. Fit separate models for each administration wave.
2. Combine into a named list: `fits <- list(Spring = fit_s, Fall = fit_f)`.
3. Call `drift <- detect_anchor_drift(fits)`.
4. Review `summary(drift)` and `plot_anchor_drift(drift)`.
5. Flagged elements may need to be removed from anchor sets or investigated for substantive causes (e.g., rater re-training).

See Also

`anchor_to_baseline()`, `build_equating_chain()`, `make_anchor_table()`, `plot_anchor_drift()`, mfrmr_linking_and_dff

Examples

```
d1 <- load_mfrmr_data("study1")
d2 <- load_mfrmr_data("study2")
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
drift <- detect_anchor_drift(list(Wave1 = fit1, Wave2 = fit2))
summary(drift)
head(drift$drift_table[, c("Facet", "Level", "Wave", "Drift", "Flag")])
drift$common_elements
```

`detect_facet_nesting` *Detect nesting structure between facets*

Description

Classifies every ordered pair of facets (optionally including `Person`) as crossed, partially nested, near-perfectly nested, or fully nested, based on a conditional-entropy index:

$$\text{nesting\_index}(A \to B) = 1 - H(B \mid A)/H(B).$$

An index near 1 means that knowing the level of `A` essentially determines the level of `B` (A is nested in B).

Usage

```
detect_facet_nesting(data, facets, person = NULL, weight_col = NULL)
```

Arguments

`data`	Data frame in long format (one row per rating).
`facets`	Character vector of facet column names.
`person`	Optional name of the person column (adds Person to the nesting matrix if supplied).
`weight_col`	Optional name of a weight column; if supplied, rows are replicated proportionally when counting element co-occurrences.

Details

This is a pure descriptive review of the observed design. It does not affect estimation; fit_mfrm() continues to treat all facets as fixed effects.

Value

A list of class `mfrm_facet_nesting` with:

- `pairwise_table`: one row per ordered facet pair with `NestingIndex_AinB`, `NestingIndex_BinA`, classification strings, and `Direction`.
- `summary`: a one-line summary table with facet counts and whether any non-crossed structure was detected.
- `facets`: the facet vector that was reviewed.

Classification bands

- "Fully nested": nesting index >= 0.99.
- "Near-perfectly nested": 0.95 <= index < 0.99.
- "Partially nested": 0.50 <= index < 0.95.
- "Crossed": index < 0.50.

The direction column records which facet is nested in which, or "crossed" when neither direction is above 0.95.

Interpreting output

A Direction value of "Rater nested in Region" means that every rater appears in exactly one region (or very close to it). For additive fixed-effects MFRM, this is a concern: the severity of a rater is confounded with region-level variance that the model cannot partition. Consider reporting the nesting direction explicitly and, when relevant, refitting without the nested facet or moving to a hierarchical estimation tool (e.g. lme4::lmer, brms, TAM) to separate the variance components.

Direction = "crossed" is the most common reading when both nesting indices are below 0.5; the two facets largely co-occur at multiple combinations, which is the setting Linacre (1989) assumed.

Typical workflow

1. Call detect_facet_nesting(data, facets) before fitting.
2. If any pair is flagged as nested or partially nested, review the numeric index and the LevelsA/LevelsB counts.
3. For downstream reporting, use analyze_hierarchical_structure() to bundle this output with ICC and design-effect summaries, which build_mfrm_manifest() then records for reproducibility.

References

McEwen, M. R. (2018). *The effects of incomplete rating designs on results from many-facets-Rasch model analyses* (Doctoral thesis, Brigham Young University). https://scholarsarchive.byu.edu/etd/6689/

Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.

See Also

facet_small_sample_review(), analyze_hierarchical_structure(), compute_facet_icc(), compute_facet_design_effect(), fit_mfrm() (see "Fixed effects assumption" in its details).

Examples

```
toy <- load_mfrmr_data("example_core")
nesting <- detect_facet_nesting(toy, c("Rater", "Criterion"))
summary(nesting)

# Synthetic example: raters fully nested within regions.
d <- data.frame(
```

```
  Person = rep(paste0("P", formatC(1:20, width = 2, flag = "0")),
               each = 6),
  Rater  = rep(paste0("R", 1:6), 20),
  Region = rep(rep(c("A", "A", "B", "B", "C", "C"), 20)),
  Score  = sample(0:4, 120, replace = TRUE),
  stringsAsFactors = FALSE
)
nest <- detect_facet_nesting(d, c("Rater", "Region"))
nest$pairwise_table[, c("FacetA", "FacetB",
                        "NestingIndex_AinB", "Direction")]
```

`diagnose_mfrm` *Compute diagnostics for an* `mfrm_fit` *object*

Description

Compute diagnostics for an `mfrm_fit` object

Usage

```
diagnose_mfrm(
  fit,
  interaction_pairs = NULL,
  top_n_interactions = 20,
  whexact = FALSE,
  fit_df_method = c("engine", "facets", "both"),
  diagnostic_mode = c("both", "legacy", "marginal_fit"),
  residual_pca = c("none", "overall", "facet", "both"),
  pca_max_factors = 10L
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`interaction_pairs`	Optional list of facet pairs.
`top_n_interactions`	Number of top interactions.
`whexact`	Logical controlling the ZSTD standardisation of mean-square fit statistics. `FALSE` (default) applies the Wilson-Hilferty cube-root transformation $(\mathrm{MnSq}^{1/3} - (1 - 2/(9\,df)))/\sqrt{2/(9\,df)}$ (recommended; the Winsteps/FACETS convention for `WHEXACT=Y`). `TRUE` uses the simpler linear-normal standardisation $(\mathrm{MnSq} - 1)\sqrt{df/2}$, which is kept for backward compatibility with earlier mfrmr summaries and with FACETS' `WHEXACT=N` mode.

`fit_df_method` Degrees-of-freedom convention used for fit ZSTD. `"engine"` (default) keeps the package-native convention `DF_Infit = sum(Var * Weight)` and `DF_Outfit = sum(Weight)`. `"facets"` uses the FACETS/Wright-Masters fourth-moment approximation `df = 2 / q^2` as the primary `InfitZSTD` / `OutfitZSTD` basis and caps reported ZSTD values at +/-9. `"both"` keeps the engine convention as the primary columns and adds `*_FACETS` companion columns for comparison.

`diagnostic_mode` Diagnostic basis to compute: `"both"` (the current default) computes both the residual/EAP-based stack and the strict latent-integrated first-order marginal-fit companion; `"legacy"` keeps the residual/EAP-based stack only; `"marginal_fit"` returns only the marginal-fit companion. The `"both"` path adds a posterior-integrated pass that typically doubles to quintuples wall-clock time relative to `"legacy"`; pass `"legacy"` explicitly when iterating on large designs and only the residual stack is needed. Use `"both"` for RSM/PCM reporting fits because it enables `plot_marginal_fit()` and `plot_marginal_pairwise()` follow-up.

`residual_pca` Residual PCA mode: `"none"`, `"overall"`, `"facet"`, or `"both"`.

`pca_max_factors` Maximum number of PCA factors to retain per matrix.

Details

This function computes a diagnostic bundle used by downstream reporting. It calculates element-level fit statistics, approximate facet separation/reliability summaries, residual-based QC diagnostics, and optionally residual PCA for exploratory residual-structure screening.

`diagnostic_mode` keeps the legacy residual fit path explicit rather than silently replacing it. The legacy path is a compatibility-oriented residual/EAP stack, whereas the strict marginal path targets latent-integrated first-order category counts. When `diagnostic_mode = "both"`, the output includes a `diagnostic_basis` guide so downstream tables and summaries can distinguish these targets.

Choosing `diagnostic_mode`:

- `"legacy"`: use when continuity with historical residual-based workflows is the priority.
- `"marginal_fit"`: use when you want the strict latent-integrated screen without the extra legacy bundle.
- `"both"`: recommended when you want continuity with the legacy residual stack while making the strict marginal path explicit for `RSM`, `PCM`, and bounded `GPCM` fits.

For bounded `GPCM`, the same generalized partial credit kernel now drives both the residual/probability tables and the strict marginal category-fit companion. Residual-based MnSq summaries should still be read as exploratory screening tools rather than strict Rasch-style invariance tests because discrimination is free, and the strict marginal companion should likewise be treated as a slope-aware screen rather than a finalized inferential test family.

Key fit statistics computed for each element:

- **Infit MnSq**: information-weighted mean-square residual; sensitive to on-target misfitting patterns. Expected value = 1.0.
- **Outfit MnSq**: unweighted mean-square residual; sensitive to off-target outliers. Expected value = 1.0.

- **ZSTD**: Wilson-Hilferty cube-root transformation of MnSq to an approximate standard normal deviate.
- **PTMEA**: point-measure correlation (item-rest correlation in MFRM context); positive values confirm alignment with the latent trait.

The MnSq values and the ZSTD values should be read separately. `mfrmr` keeps the package-native engine df convention by default because it is the basis used by the R/Python/Julia validation engines. FACETS reports closely related MnSq values but standardizes them with a Wright-Masters fourth-moment df approximation (`df = 2 / q^2`) and caps reported ZSTD values. Use `fit_df_method = "both"` to review these two standardization conventions side by side without changing the primary `InfitZSTD` / `OutfitZSTD` columns.

Residual basis under MML. For `method = "MML"` fits, residuals, MnSq, and ZSTD are computed at the EAP person measures from the marginal model. EAP measures are shrunken toward the population mean, so expected scores – and therefore fit statistics – differ systematically from JMLE-based engines such as FACETS, especially for persons with extreme raw scores. The df conventions above do not remove this difference: it is a residual-basis difference, not a standardization difference. Refit with `method = "JML"` when an external FACETS fit comparison requires a JMLE-style residual basis (see `facets_fit_review()`).

Misfit flagging guidelines (Bond & Fox, 2015):

- MnSq < 0.5: overfit (too predictable; may inflate reliability)
- MnSq 0.5–1.5: productive for measurement
- MnSq > 1.5: underfit (noise degrades measurement)
- $|\text{ZSTD}| > 2$: statistically significant misfit (5\

When Infit and Outfit disagree, Infit is generally more informative because it downweights extreme observations. Large Outfit with acceptable Infit typically indicates a few outlying responses rather than systematic misfit.

`interaction_pairs` controls which facet interactions are summarized. Each element can be:

- a length-2 character vector such as `c("Rater", "Criterion")`, or
- omitted (`NULL`) to let the function select top interactions automatically.

Residual PCA behavior:

- `"none"`: skip PCA (fastest; recommended for initial exploration)
- `"overall"`: compute overall residual PCA across all facets
- `"facet"`: compute facet-specific residual PCA for each facet
- `"both"`: compute both overall and facet-specific PCA

Overall PCA examines the person $\times$ combined-facet residual matrix; facet-specific PCA examines person $\times$ facet-level matrices. These summaries are exploratory screens for residual structure, not standalone proofs for or against unidimensionality. Facet-specific PCA can help localise where a stronger residual signal is concentrated.

These residual-PCA summaries are not a DIMTEST/UNIDIM implementation. DIMTEST-style essential-unidimensionality tests work at an item-response layer and require an explicit decision about how many-facet rating data are collapsed, conditioned, or adjusted for rater/task/facet effects. For manuscripts, combine global/element fit, residual PCA, and local-dependence screens, and use limited wording such as "evidence consistent with essential unidimensionality under the specified facet structure" rather than "unidimensionality was established."

Value

An object of class `mfrm_diagnostics` including:

- `obs`: observed/expected/residual-level table
- `measures`: facet/person fit table (`Infit`, `Outfit`, `ZSTD`, `PTMEA`, `ModelSE`, `RealSE`, `CI_Lower`, `CI_Upper`, `CI_Level`, `CI_Method`)
- `overall_fit`: overall fit summary
- `fit`: element-level fit diagnostics
- `reliability`: facet-level model/real separation and reliability
- `precision_profile`: one-row summary of the active precision tier and its recommended use
- `precision_review`: package-native checks for SE, CI, and reliability
- `parameter_uncertainty`: MML observed-information uncertainty for structural parameters when available (`steps`, and bounded-`GPCM` `slopes` on both log and positive scales), plus covariance status metadata
- `facet_precision`: facet-level precision summary by distribution basis and SE mode
- `facets_chisq`: fixed/random facet variability summary
- `interactions`: top interaction diagnostics
- `interrater`: inter-rater agreement bundle (`summary`, `pairs`) including agreement and rater-severity spread indices
- `unexpected`: unexpected-response bundle
- `fair_average`: adjusted-score reference bundle (reported as unavailable for bounded `GPCM`)
- `displacement`: displacement diagnostics bundle
- `approximation_notes`: method notes for SE/CI/reliability summaries
- `diagnostic_basis`: guide to the statistical target of each diagnostic path
- `fit_standardization`: guide to the df convention behind fit ZSTD values
- `marginal_fit`: optional strict marginal-fit companion based on posterior-expected first-order category counts
- `residual_pca_overall`: optional overall PCA object
- `residual_pca_by_facet`: optional facet PCA objects

Reading key components

Practical interpretation often starts with:

- `overall_fit`: global infit/outfit and degrees of freedom.
- `reliability`: facet-level model/real separation and reliability. `MML` uses model-based `ModelSE` values where available; `JML` keeps these quantities as exploratory approximations.
- `fit`: element-level misfit scan (`Infit`, `Outfit`, `ZSTD`).
- `unexpected`, `fair_average`, `displacement`: targeted QC bundles. For bounded `GPCM`, `fair_average` is retained with an unavailable status because that compatibility calculation has not yet been validated for the generalized model.
- `approximation_notes`: method notes for SE/CI/reliability summaries.

Interpreting output

Start with `overall_fit` and `reliability`, then move to element-level diagnostics (`fit`) and targeted bundles (`unexpected`, `displacement`, `interrater`, `facets_chisq`). Treat `fair_average` as available only for the `RSM` / `PCM` branch.

Consistent signals across multiple components are typically more robust than a single isolated warning. For example, an element flagged for both high Outfit and high displacement is more concerning than one flagged on a single criterion.

`SE` is kept as a compatibility alias for `ModelSE`. `RealSE` is a fit-adjusted companion defined as `ModelSE * sqrt(max(Infit, 1))`. Reliability tables report model and fit-adjusted bounds from observed variance, error variance, and true variance; `JML` entries should still be treated as exploratory. Separation, strata, and reliability follow the Wright & Masters (1982) conventions: $G = \mathrm{TrueSD}/\mathrm{RMSE}$, $R = G^2/(1+G^2)$, and $H = (4G+1)/3$.

Typical workflow

1. Start with `diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")`.
2. Inspect `summary(diag)` and use `diagnostic_basis` to separate legacy residual evidence from strict marginal evidence.
3. If needed, rerun with residual PCA (`"overall"` or `"both"`).

References

- Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press. (G/R/H separation, reliability, and strata formulas summarized in `s_diag$reliability` follow this convention.)
- Wright, B. D., & Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions, 8*(3), 370. (Source for the 0.5-1.5 Infit / Outfit acceptance band that `s_diag$key_warnings` and `misfit_thresholds` apply.)
- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press. (FACETS Tables 6 + 7 correspond to the per-facet element measures, fit, and chi-square heterogeneity screen exposed via `s_diag$reliability` and `s_diag$facets_chisq`.)
- Bond, T. G., & Fox, C. M. (2015). *Applying the Rasch model: Fundamental measurement in the human sciences* (3rd ed.). Routledge. (Reference text for the Rasch-family fit conventions exposed by this helper.)
- Linacre, J. M. (2002). What do Infit and Outfit, Mean-square and Standardized mean? *Rasch Measurement Transactions, 16*(2), 878.
- Linacre, J. M. (2026). *A user's guide to Facets Rasch-model computer programs*. Winsteps.com. (WHEXACT / FACETS standardized fit df notes.)

See Also

`fit_mfrm()`, `analyze_residual_pca()`, `build_visual_summaries()`, mfrmr_visual_diagnostics, mfrmr_reporting_and_apa

Examples

```
# Fast smoke run: legacy-only diagnostic mode is enough to confirm
# the bundle has the expected slots. ~1 s on example_core.
toy <- load_mfrmr_data("example_core")
fit_quick <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                      method = "JML", maxit = 30)
diag_quick <- diagnose_mfrm(fit_quick, diagnostic_mode = "legacy",
                            residual_pca = "none")
summary(diag_quick)$overview[, c("Observations", "Facets", "Categories")]


fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, diagnostic_mode = "both", residual_pca = "none")
s_diag <- summary(diag)
s_diag$overview[, c("Observations", "Facets", "Categories")]
s_diag$diagnostic_basis[, c("DiagnosticPath", "Status", "Basis")]
s_diag$key_warnings
# Look for: "No immediate warnings ..." in `key_warnings` is the
#   "all clear" signal. Lines starting with "MnSq misfit:" name the
#   element + Infit / Outfit values that fell outside the
#   0.5-1.5 acceptance band; review those first.
s_diag$facets_chisq
# Look for: `FixedProb` < 0.05 means that facet's elements differ
#   reliably under the fixed-effect "all elements equal" null. A
#   facet with a non-significant chi-square contributes little
#   spread to the test scale.
s_diag$interrater
# Look for: ExactAgreement >= ExpectedExactAgreement and
#   AgreementMinusExpected >= 0 indicate raters agree at least as
#   often as the model expects. Negative values warrant a closer
#   look at `diag$interrater$pairs`.
p_qc <- plot_qc_dashboard(fit, diagnostics = diag, draw = FALSE)
p_qc$data$plot

# Optional: include residual PCA in the diagnostic bundle
diag_pca <- diagnose_mfrm(fit, residual_pca = "overall")
pca <- analyze_residual_pca(diag_pca, mode = "overall")
head(pca$overall_table)

# Reporting route:
prec <- precision_review_report(fit, diagnostics = diag)
summary(prec)
```

`dif_interaction_table` *Compute interaction table between a facet and a grouping variable*

Description

Produces a cell-level interaction table showing Obs-Exp differences, standardized residuals, and screening statistics for each facet-level x group-value cell.

Usage

```
dif_interaction_table(
  fit,
  diagnostics,
  facet,
  group,
  data = NULL,
  min_obs = 10,
  p_adjust = "holm",
  abs_t_warn = 2,
  abs_bias_warn = 0.5
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Output from diagnose_mfrm().
facet	Character scalar naming the facet.
group	Character scalar naming the grouping column.
data	Optional data frame with the group column. If NULL (default), the data stored in fit$prep$data is used, but it must contain the group column.
min_obs	Minimum observations per cell. Cells with fewer than this many observations are flagged as sparse and their test statistics set to NA. Default 10.
p_adjust	P-value adjustment method, passed to stats::p.adjust(). Default "holm".
abs_t_warn	Threshold for flagging cells by absolute t-value. Default 2.
abs_bias_warn	Threshold for flagging cells by absolute Obs-Exp average (in logits). Default 0.5.

Details

This function uses the fitted model's observation-level residuals (from the internal compute_obs_table() function) rather than re-estimating the model. For each facet-level x group-value cell, it computes:

- N: number of observations in the cell
- ObsScore: sum of observed scores
- ExpScore: sum of expected scores
- ObsExpAvg: mean observed-minus-expected difference
- Var_sum: sum of model variances
- StdResidual: (ObsScore - ExpScore) / sqrt(Var_sum)
- t: approximate t-statistic (equal to StdResidual)

- df: N - 1
- p_value: two-tailed p-value from the t-distribution

Value

Object of class `mfrm_dif_interaction` with:

- `table`: tibble with per-cell statistics and flags.
- `summary`: tibble summarizing flagged and sparse cell counts.
- `gpcm_boundary`: for bounded `GPCM` fits, a capability-boundary table.
- `config`: list of analysis parameters.

When to use this instead of analyze_dff()

Use `dif_interaction_table()` when you want cell-level screening for a single facet-by-group table. Use `analyze_dff()` when you want group-pair contrasts summarized into differential-functioning effect sizes and method-appropriate classifications.

Further guidance

For plot selection and follow-up diagnostics, see mfrmr_visual_diagnostics.

Interpreting output

- `$table`: the full interaction table with one row per cell.
- `$summary`: overview counts of flagged and sparse cells.
- `$config`: analysis configuration parameters.
- `$gpcm_boundary`: for bounded `GPCM` fits, a capability-boundary table marking the table as caveated DFF screening evidence.
- Cells with `|t| > abs_t_warn` or `|ObsExpAvg| > abs_bias_warn` are flagged in the `flag_t` and `flag_bias` columns.
- Sparse cells (N < min_obs) have `sparse = TRUE` and NA statistics.

GPCM boundary

For bounded `GPCM`, the interaction table uses the fitted slope-aware expected-score/residual scale and should be reported as screening evidence, not as a standalone fairness, invariance, or operational subgroup decision.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Run `dif_interaction_table(fit, diag, facet = "Rater", group = "Gender", data = df)`.
3. Inspect `$table` for flagged cells.
4. Visualize with `plot_dif_heatmap()`.

See Also

analyze_dff(), analyze_dif(), plot_dif_heatmap(), dif_report(), estimate_bias()

Examples

```
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
int <- dif_interaction_table(fit, diag, facet = "Rater",
                             group = "Group", data = toy, min_obs = 2)
int$summary
head(int$table[, c("Level", "GroupValue", "ObsExpAvg", "flag_bias")])
```

---

dif_report *Generate a differential-functioning interpretation report*

Description

Produces APA-style narrative text interpreting the results of a differential- functioning analysis or interaction table. For method = "refit", the report summarises the number of facet levels classified as negligible (A), moderate (B), and large (C). For method = "residual", it summarises screening-positive results, lists the specific levels and their direction, and includes a caveat about the distinction between construct-relevant variation and measurement bias.

Usage

```
dif_report(dif_result, ...)
```

Arguments

dif_result	Output from analyze_dff() / analyze_dif() (class mfrm_dff with compatibility class mfrm_dif) or dif_interaction_table() (class mfrm_dif_interaction).
...	Currently unused; reserved for future extensions.

Details

When dif_result is an mfrm_dff/mfrm_dif object, the report is based on the pairwise differential-functioning contrasts in $dif_table. When it is an mfrm_dif_interaction object, the report uses the cell-level statistics and flags from $table.

For method = "refit", ETS-style magnitude labels are used only when subgroup calibrations were successfully linked back to a common baseline scale; otherwise the report labels those contrasts as unclassified because the refit difference is descriptive rather than comparable on a linked logit scale. For method = "residual", the report describes screening-positive versus screening-negative contrasts instead of applying ETS labels.

Value

Object of class `mfrm_dif_report` with `narrative`, `counts`, `large_dif`, `gpcm_boundary`, and `config`.

Interpreting output

- `$narrative`: character scalar with the full narrative text.
- `$counts`: named integer vector of method-appropriate counts.
- `$large_dif`: tibble of large ETS results (`method = "refit"`) or screening-positive contrasts/cells (`method = "residual"`).
- `$gpcm_boundary`: for bounded `GPCM` inputs, a capability-boundary table marking the narrative as caveated DFF screening output.
- `$config`: analysis configuration inherited from the input.

GPCM boundary

If the input comes from a bounded `GPCM` fit, the narrative includes a bounded-`GPCM` note and the returned report carries `gpcm_boundary`. Treat the text as slope-aware screening/reporting support, not as a standalone fairness, invariance, or operational subgroup decision.

Typical workflow

1. Run `analyze_dff()` / `analyze_dif()` or `dif_interaction_table()`.
2. Pass the result to `dif_report()`.
3. Print the report or extract `$narrative` for inclusion in a manuscript.

References

The narrative caveat about distinguishing construct-relevant variation from unwanted measurement bias is grounded in:

- Eckes, T. (2011). *Introduction to Many-Facet Rasch Measurement: Analyzing and Evaluating Rater-Mediated Assessments*. Frankfurt am Main: Peter Lang. ISBN 978-3-631-61350-4.
- McNamara, T., & Knoch, U. (2012). The Rasch wars: The emergence of Rasch measurement in language testing. *Language Testing*, 29(4), 555–576. doi:10.1177/0265532211430367

See Also

`analyze_dff()`, `analyze_dif()`, `dif_interaction_table()`, `plot_dif_heatmap()`, `build_apa_outputs()`

Examples

```
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
dif <- analyze_dff(fit, diag, facet = "Rater", group = "Group", data = toy)
rpt <- dif_report(dif)
cat(rpt$narrative)
```

`displacement_table` *Compute displacement diagnostics for facet levels*

Description

Compute displacement diagnostics for facet levels

Usage

```
displacement_table(
  fit,
  diagnostics = NULL,
  facets = NULL,
  anchored_only = FALSE,
  abs_displacement_warn = 0.5,
  abs_t_warn = 2,
  top_n = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`facets`	Optional subset of facets.
`anchored_only`	If `TRUE`, keep only directly/group anchored levels.
`abs_displacement_warn`	Absolute displacement warning threshold.
`abs_t_warn`	Absolute displacement t-value warning threshold.
`top_n`	Optional maximum number of rows to keep after sorting.

Details

Displacement is computed as a one-step Newton update: `sum(residual) / sum(information)` for each facet level. This approximates how much a level would move if constraints were relaxed.

Value

A named list with:

- `table`: displacement diagnostics by level
- `summary`: one-row summary
- `thresholds`: applied thresholds

Interpreting output

- `table`: level-wise displacement and flag indicators.
- `summary`: count/share of flagged levels.
- `thresholds`: displacement and t-value cutoffs.

Large absolute displacement in anchored levels suggests potential instability in anchor assumptions.

Typical workflow

1. Run `displacement_table(fit, anchored_only = TRUE)` for anchor checks.
2. Inspect `summary(disp)` then detailed rows.
3. Visualize with `plot_displacement()`.

Output columns

The `table` data.frame contains:

Facet, Level Facet name and element label.

Displacement One-step Newton displacement estimate (logits).

DisplacementSE Standard error of the displacement.

DisplacementT Displacement / SE ratio.

Estimate, SE Current measure estimate and its standard error.

N Number of observations involving this level.

AnchorValue, AnchorStatus, AnchorType Anchor metadata.

Flag Logical; `TRUE` when displacement exceeds thresholds.

See Also

`diagnose_mfrm()`, `unexpected_response_table()`, `fair_average_table()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
disp <- displacement_table(fit, anchored_only = FALSE)
summary(disp)
p_disp <- plot(disp, draw = FALSE)
p_disp$data$plot
```

draw_mfrm_resamples *Draw observed-data MFRM resamples*

Description

Draw observed-data MFRM resamples

Usage

```
draw_mfrm_resamples(spec, keep_data = TRUE)
```

Arguments

spec	Output from build_mfrm_resampling_spec().
keep_data	Logical; if TRUE, return a list of replicate data frames. If FALSE, return manifest tables only.

Value

An object of class mfrm_resamples with samples, manifest, stratum_manifest, and preserve_manifest.

See Also

build_mfrm_resampling_spec()

ej2021_data *Simulated MFRM datasets based on Eckes and Jin (2021)*

Description

Synthetic many-facet rating datasets in long format. All datasets include one row per observed rating.

Format

A data.frame with 5 columns:

Study Study label ("Study1" or "Study2").

Person Person/respondent identifier.

Rater Rater identifier.

Criterion Criterion facet label.

Score Observed category score.

Details

Available data objects:

- `mfrmr_example_core`
- `mfrmr_example_bias`
- `ej2021_study1`
- `ej2021_study2`
- `ej2021_combined`
- `ej2021_study1_itercal`
- `ej2021_study2_itercal`
- `ej2021_combined_itercal`

Naming convention:

- `study1` / `study2`: separate simulation studies
- `combined`: row-bind of study1 and study2
- `_itercal`: iterative-calibration variant

Use `load_mfrmr_data()` for programmatic selection by key.

Data dimensions

Dataset	Rows	Persons	Raters	Criteria
study1	1842	307	18	3
study2	3287	206	12	9
combined	5129	307	18	12
study1_itercal	1842	307	18	3
study2_itercal	3341	206	12	9
combined_itercal	5183	307	18	12

Score range: 1–4 (four-category rating scale).

Simulation design

Person ability is drawn from N(0, 1). Rater severity effects span approximately -0.5 to +0.5 logits. Criterion difficulty effects span approximately -0.3 to +0.3 logits. Scores are generated from the resulting linear predictor plus Gaussian noise, then discretized into four categories. The `_itercal` variants use a second iteration of calibrated rater severity parameters.

Interpreting output

Each dataset is already in long format and can be passed directly to `fit_mfrm()` after confirming column-role mapping.

Typical workflow

1. Inspect available datasets with `list_mfrmr_data()`.
2. Load one dataset using `load_mfrmr_data()`.
3. Fit and diagnose with `fit_mfrm()` and `diagnose_mfrm()`.

Source

Simulated for this package with design settings informed by Eckes and Jin (2021). The Eckes & Jin (2021) Method section reports the following design parameters that motivated the synthetic versions shipped here: Study 1 had 307 examinees (149 males, 158 females), 18 raters (4 males, 14 females), and 3 criteria (global impression, task fulfillment, linguistic realization) on a 4-category rating scale (TDN levels rescored 1-4); Study 2 had 206 examinees (66 males, 140 females), 12 raters (1 male, 11 females), and 9 criteria on the same 4-category scale. The packaged datasets reproduce these (examinees, raters, criteria, categories) shapes but use simulated responses, so they are not the real TestDaF data.

References

Eckes, T., & Jin, K.-Y. (2021). Measuring rater centrality effects in writing assessment: A Bayesian facets modeling approach. *Psychological Test and Assessment Modeling, 63*(1), 65–94.

Examples

```
data("ej2021_study1", package = "mfrmr")
head(ej2021_study1)
table(ej2021_study1$Study)
```

---

estimate_all_bias *Estimate bias across multiple facet pairs*

Description

Estimate bias across multiple facet pairs

Usage

```
estimate_all_bias(
  fit,
  diagnostics = NULL,
  pairs = NULL,
  include_person = FALSE,
  drop_empty = TRUE,
  keep_errors = TRUE,
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are computed with residual_pca = "none".
pairs	Optional list of facet specifications. Each element should be a character vector of length 2 or more, for example list(c("Rater", "Criterion"), c("Task", "Criterion")). When NULL, all 2-way combinations of modeled facets are used.
include_person	If TRUE and pairs = NULL, include "Person" in the automatically generated pair set.
drop_empty	If TRUE, omit empty bias tables from by_pair while still recording them in the summary table.
keep_errors	If TRUE, retain per-pair error rows in the returned errors table instead of failing the whole batch.
max_abs	Passed to estimate_bias().
omit_extreme	Passed to estimate_bias().
max_iter	Passed to estimate_bias().
tol	Passed to estimate_bias().

Details

This function orchestrates repeated calls to estimate_bias() across multiple facet pairs and returns a consolidated bundle.

Bias/interaction in MFRM refers to a systematic departure from the additive model for a specific combination of facet elements (e.g., a particular rater is unexpectedly harsh on a particular criterion). See estimate_bias() for the mathematical formulation.

When pairs = NULL, the function builds all 2-way combinations of modelled facets automatically. For a model with facets Rater, Criterion, and Task, this yields Rater×Criterion, Rater×Task, and Criterion×Task.

The summary table aggregates results across pairs:

- Rows: number of interaction cells estimated
- Significant: count of cells with $|t| \geq 2$
- MeanAbsBias: average absolute bias magnitude (logits)

Per-pair failures (e.g., insufficient data for a sparse pair) are captured in errors rather than stopping the entire batch.

Value

A named list with class mfrm_bias_collection.

Output

The returned object is a bundle-like list with class `mfrm_bias_collection` and components such as:

- `summary`: one row per requested interaction
- `by_pair`: named list of successful `estimate_bias()` outputs
- `errors`: per-pair error log
- `settings`: resolved execution settings
- `primary`: first successful bias bundle, useful for downstream helpers

Typical workflow

1. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`. For `RSM` / `PCM` reporting runs, prefer `method = "MML"` plus `diagnostic_mode = "both"` in the diagnostics call.
2. Run `estimate_all_bias()` to compute app-style multi-pair interactions.
3. Pass the resulting `by_pair` list into `reporting_checklist()` or `facet_quality_dashboard()`.

See Also

`estimate_bias()`, `reporting_checklist()`, `facet_quality_dashboard()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
bias_all <- estimate_all_bias(fit, diagnostics = diag)
bias_all$summary[, c("Interaction", "Rows", "Significant")]
```

---

estimate_bias *Estimate bias and interaction screening terms*

Description

Estimate bias and interaction screening terms

Usage

```
estimate_bias(
  fit,
  diagnostics,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
```

```
  max_abs = 10,
  omit_extreme = TRUE,
  max_iter = 4,
  tol = 0.001
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Output from diagnose_mfrm().
facet_a	First facet name. Provide together with facet_b for the classic pairwise 2-way interaction. Ignored when interaction_facets is supplied.
facet_b	Second facet name. See facet_a.
interaction_facets	Character vector of two or more facets to model as one interaction effect. When supplied, this takes precedence over facet_a/facet_b. Use this form (rather than facet_a/facet_b) whenever you want 3+ way interactions, since facet_a/facet_b is restricted to the pairwise case.
max_abs	Bound for absolute bias size.
omit_extreme	Omit extreme-only elements.
max_iter	Iteration cap.
tol	Convergence tolerance.

Details

Bias (interaction) in MFRM refers to a systematic departure from the additive model: a specific rater-criterion (or higher-order) combination produces scores that are consistently higher or lower than predicted by the main effects alone. For example, Rater A might be unexpectedly harsh on Criterion 2 despite being lenient overall.

Mathematically, the bias term b_{jc} for rater j on criterion c modifies the linear predictor:

$$\eta_{njc} = \theta_n - \delta_j - \beta_c - b_{jc}$$

For RSM / PCM, the function estimates b_{jc} from the residuals of the fitted additive model using an iterative recalibration screen aligned with the many-facet bias literature (Myford & Wolfe, 2003, 2004):

$$b_{jc} = \frac{\sum_n (X_{njc} - E_{njc})}{\sum_n \mathrm{Var}_{njc}}$$

Each iteration updates expected scores using the current bias estimates, then re-computes the bias. Convergence is reached when the maximum absolute change in bias estimates falls below tol. For bounded GPCM, the same additive-bias idea is evaluated with the slope-aware GPCM kernel and conditional profile-likelihood follow-up columns; those quantities remain screening evidence because theta, facet, step, and slope estimates are held fixed.

- For two-way mode, use facet_a and facet_b (or interaction_facets with length 2).
- For higher-order mode, provide interaction_facets with length >= 3.

Value

An object of class `mfrm_bias` with:

- `table`: interaction rows with effect size, SE, screening t/p metadata, reporting-use flags, fit columns, and bounded-GPCM profile-likelihood columns when available
- `summary`: compact summary statistics
- `chi_sq`: fixed-effect chi-square style screening summary
- `facet_a`, `facet_b`: first two analyzed facet names (legacy compatibility)
- `interaction_facets`, `interaction_order`, `interaction_mode`: full interaction metadata
- `iteration`: iteration history/metadata
- `orientation_review`: facet-orientation sign-consistency review table
- `mixed_sign`: logical flag indicating whether bias-size signs flip across facets in a way that complicates direction interpretation
- `direction_note`: one-line interpretive note describing the dominant bias direction (empty when not applicable)
- `recommended_action`: one-line recommended-action label routing the user to the appropriate follow-up helper
- `inference_tier`: summary label indicating that the bias rows are intended for screening and follow-up review in this release
- `optimization_failures`: per-cell record of any inner-loop optimizer failures encountered while estimating the bias parameters; empty when every cell converged cleanly

What this screening means

`estimate_bias()` summarizes interaction departures from the additive MFRM. It is best read as a targeted screening tool for potentially noteworthy cells or facet combinations that may merit substantive review.

What this screening does not justify

- `t` and `Prob.` are screening metrics, not formal inferential quantities.
- A flagged interaction cell is not, by itself, proof of rater bias or construct-irrelevant variance.
- Non-flagged cells should not be over-read as evidence that interaction effects are absent.

Interpreting output

Use `summary` for global magnitude, then inspect `table` for cell-level interaction effects.

Prioritize rows with:

- larger `|Bias Size|` (effect on logit scale; > 0.5 logits is typically noteworthy, > 1.0 is large)
- larger `|t|` among the screening metrics ($|t| \geq 2$ suggests a screen-positive interaction cell)
- smaller `Prob.` among the screening metrics

A positive `Obs-Exp Average` means the cell produced *higher* scores than the additive model predicts (unexpected leniency); negative means unexpected harshness.

`iteration` helps verify whether iterative recalibration stabilized. If the maximum change on the final iteration is still above `tol`, consider increasing `max_iter`.

Typical workflow

1. Fit and diagnose model.
2. Run `estimate_bias(...)` for target interaction facets.
3. Review `summary(bias)` and `bias$table`.
4. Visualize/report via `plot_bias_interaction()` and `build_fixed_reports()`.

Interpreting key output columns

In `bias$table`, the most-used columns are:

- `Bias Size`: estimated interaction effect b_{jc} (logit scale)
- `t` and `Prob.`: screening metrics, not formal inferential quantities
- `Obs-Exp Average`: direction and practical size of observed-vs-expected gap on the raw-score metric
- for bounded `GPCM`, `LR ChiSq`, `LR Prob.`, and `Profile CI Lower` / `Profile CI Upper`: conditional profile-likelihood checks for a single additive bias shift, holding the fitted person, facet, step, and slope estimates fixed

The `chi_sq` element provides a fixed-effect heterogeneity screen across all interaction cells.

Recommended next step

Use `plot_bias_interaction()` to inspect the flagged cells visually, then integrate the result with DFF, linking, or substantive scoring review before making formal claims about fairness or invariance.

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press. (FACETS Table 13 corresponds to the bias / interaction estimation that this helper implements.)
- Eckes, T. (2005). Examining rater effects in TestDaF writing and speaking performance assessments: A many-facet Rasch analysis. *Language Assessment Quarterly, 2*(3), 197-221.
- Eckes, T. (2015). *Introduction to many-facet Rasch measurement: Analyzing and evaluating rater-mediated assessments* (2nd ed.). Peter Lang.
- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement, 4*(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement, 5*(2), 189-227.

See Also

`build_fixed_reports()`, `build_apa_outputs()`

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
s_bias <- summary(bias)
s_bias$overview
# Look for: `MaxAbsBias` < ~0.5 logits and `Significant = 0` mean
#   no cell exceeded the screen. The `BonferroniSignificant` /
#   `HolmSignificant` columns count cells that survive multiple-
#   testing correction; both being 0 is a stronger "no bias"
#   signal than the raw screen-positive count alone.
s_bias$top_rows
# Look for: rows with `|t|` > 2 and |Bias Size| > 0.5 logits warrant
#   review (large effect AND statistically reliable). Rows with only
#   one of those triggered are usually small-cell artefacts.
p_bias <- plot_bias_interaction(bias, draw = FALSE)
p_bias$data$plot
```

---

estimation_iteration_report

Build an estimation-iteration report (preferred alias)

Description

Build an estimation-iteration report (preferred alias)

Usage

```
estimation_iteration_report(
  fit,
  max_iter = 20,
  reltol = NULL,
  include_prox = TRUE,
  include_fixed = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
max_iter	Maximum replay iterations (excluding optional initial row).
reltol	Stopping tolerance for replayed max-logit change.
include_prox	If TRUE, include an initial pseudo-row labeled PROX.
include_fixed	If TRUE, include a legacy-compatible fixed-width text block.

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_iteration_report (type = "residual", "logit_change", "objective").

Value

A named list with iteration-report components. Class: mfrm_iteration_report.

Interpreting output

- iterations: trajectory of convergence indicators by iteration.
- summary: final status and stopping diagnostics.
- optional PROX row: pseudo-initial reference point when enabled.

Typical workflow

1. Run estimation_iteration_report(fit).
2. Inspect plateau/stability patterns in summary/plot.
3. Adjust optimization settings if convergence looks weak.

See Also

fit_mfrm(), specifications_report(), data_quality_report(), mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- estimation_iteration_report(fit, max_iter = 5)
summary(out)
p_iter <- plot(out, draw = FALSE)
p_iter$data$plot
```

---

evaluate_mfrm_design *Evaluate MFRM design conditions by repeated simulation*

Description

Evaluate MFRM design conditions by repeated simulation

Usage

```
evaluate_mfrm_design(
  n_person = c(30, 50, 100),
  n_rater = c(3, 5),
  n_criterion = c(3, 5),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  fit_method = c("JML", "MML"),
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  seed = NULL,
  progress = interactive(),
  parallel = c("no", "future")
)
```

Arguments

`n_person`	Vector of person counts to evaluate.
`n_rater`	Vector of rater counts to evaluate.
`n_criterion`	Vector of criterion counts to evaluate.
`raters_per_person`	Vector of rater assignments per person.
`design`	Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (`n_person`, `n_rater`, `n_criterion`, `raters_per_person`), current public aliases implied by `sim_spec` (for example `n_judge`, `n_task`, `judge_per_person`), or role keywords (`person`, `rater`, `criterion`, `assignment`). Values may be vectors. The schema-only future branch input `design$facets = c(person = ..., judge = ..., task = ...)` is also accepted for the currently exposed facet keys. Do not specify the same variable through both `design` and the scalar design-grid arguments.
`reps`	Number of replications per design condition.
`score_levels`	Number of ordered score categories.

theta_sd — Standard deviation of simulated person measures.

rater_sd — Standard deviation of simulated rater severities.

criterion_sd — Standard deviation of simulated criterion difficulties.

noise_sd — Optional observation-level noise added to the linear predictor.

step_span — Spread of step thresholds on the logit scale.

fit_method — Estimation method passed to fit_mfrm().

model — Measurement model passed to fit_mfrm(). RSM and PCM use the validated Rasch-family design-planning layer. Bounded GPCM is available as a caveated simulation/refit operating-characteristic route.

step_facet — Step facet passed to fit_mfrm() when model = "PCM" or model = "GPCM". When left NULL, the function inherits the generator step facet from sim_spec when available and otherwise defaults to "Criterion".

slope_facet — Slope facet passed to fit_mfrm() when model = "GPCM". The current bounded branch requires slope_facet == step_facet.

slopes — Optional bounded-GPCM generator slopes used when sim_spec = NULL. See build_mfrm_sim_spec() for accepted formats.

assignment — Optional assignment design used when sim_spec = NULL. "sparse_linked" activates planned-missing sparse rating designs; use sparse_controls to specify the linking set.

sparse_controls — Optional named list used when assignment = "sparse_linked" and sim_spec = NULL, or retained from a sparse linked sim_spec. See build_mfrm_sim_spec().

maxit — Maximum iterations passed to fit_mfrm().

quad_points — Quadrature points for fit_method = "MML".

residual_pca — Residual PCA mode passed to diagnose_mfrm().

sim_spec — Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec() used as the base data-generating mechanism. When supplied, the design grid still varies n_person, n_rater, n_criterion, and raters_per_person, but latent-spread assumptions, thresholds, and other generator settings come from sim_spec. If sim_spec contains step-facet-specific thresholds, the design grid may not vary the number of levels for that step facet away from the specification. If sim_spec stores an active latent-regression population generator, this helper currently requires fit_method = "MML" so each replication can refit the population model.

seed — Optional seed for reproducible replications.

progress — Logical. Whether to show a progress bar across design-by-replication cells. Defaults to interactive(), so interactive exploratory runs show progress while non-interactive tests, scripts, and report rendering stay quiet. Set TRUE or FALSE explicitly to override.

parallel — Parallelisation strategy for the rep loop within each design row. "no" (default) runs serially; "future" uses future.apply::future_lapply and respects whatever future::plan() is currently active. The Suggests package future.apply must be installed for the parallel path to activate; otherwise the call falls back to serial execution with a single message. Cross-design-row parallelism is planned for a future release.

Details

This helper runs a compact Monte Carlo design study for common rater-by-item many-facet settings.

For each design condition, the function:

1. generates synthetic data with `simulate_mfrm_data()`
2. fits the requested MFRM with `fit_mfrm()`
3. computes diagnostics with `diagnose_mfrm()`
4. stores recovery and precision summaries by facet

The result is intended for planning questions such as:

- how many raters are needed for stable rater separation?
- how does `raters_per_person` affect severity recovery?
- when do category counts become too sparse for comfortable interpretation?

This is a **parametric simulation study**. It does not take one observed design (for example, 4 raters x 30 persons x 3 criteria) and analytically extrapolate what would happen under a different design (for example, 2 raters x 40 persons x 5 criteria). Instead, you specify a design grid and data-generating assumptions (latent spread, facet spread, thresholds, noise, and scoring structure), and the function repeatedly generates synthetic data under those assumptions.

When you want the simulated conditions to resemble an existing study, use substantive knowledge or estimates from that study to choose `theta_sd`, `rater_sd`, `criterion_sd`, `score_levels`, and related settings before running the design evaluation.

When `sim_spec` is supplied, the function uses it as the explicit data-generating mechanism. This is the recommended route when you want a design study to stay close to a previously fitted run while still varying the candidate sample sizes or rater-assignment counts.

Sparse linked simulation specifications and direct `assignment = "sparse_linked"` calls are carried into the design-evaluation output. The resulting sparse-design columns report planned-missingness and rater-link diagnostics (for example design density and rater-pair common persons). They are design diagnostics, not fit statistics or universal adequacy thresholds.

If that specification also stores a latent-regression population generator, each replication carries forward the simulated one-row-per-person background data and refits the MML population-model branch. This remains a scenario study under explicit assumptions; it is not a closed-form predictive distribution for one future administration.

Bounded `GPCM` design evaluation is available with caveats. It repeatedly generates data from the supplied or fit-derived slope-aware specification, refits bounded `GPCM`, and summarizes facet-level operating characteristics. The route remains a role-based person x rater-like x criterion-like planner: it does not validate diagnostic-screening or signal-detection rules, does not provide a fully arbitrary-facet planner, and does not replace `evaluate_mfrm_recovery()` for slope-recovery adequacy review.

Recovery metrics are reported only when the generator and fitted model target the same facet-parameter contract. In practice this means the same `model`, and for `PCM`, the same `step_facet`. When these do not align, recovery fields are set to `NA` and the output records the reason. Even when these contract checks pass, the recovery summaries still assume compatible orientation and anchoring conventions across the generator and fitted model.

Value

An object of class `mfrm_design_evaluation` with components:

- `design_grid`: evaluated design conditions. When `sim_spec` carries custom public facet names, matching design-variable alias columns are included alongside the canonical internal columns.
- `results`: facet-level replicate results, with the same design-variable alias columns when applicable.
- `rep_overview`: run-level status and timing, with the same design-variable alias columns when applicable.
- `design_descriptor`: role-based design-variable metadata used by planning summaries and plots
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of which design variables remain mutable under the current simulation specification
- `planning_schema`: combined planner-schema contract bundling the role descriptor, scope boundary, and current mutability map
- `gpcm_boundary`: bounded-`GPCM` caveat row when a `GPCM` design route is used
- `notes`: short interpretation notes
- `settings`: simulation settings
- `ademp`: simulation-study metadata (aims, DGM, estimands, methods, performance measures)

Reported metrics

Facet-level simulation results include:

- `Separation` ($G = \mathrm{SD}_{\mathrm{adj}}/\mathrm{RMSE}$): how many statistically distinct strata the facet resolves.
- `Reliability` ($G^2/(1+G^2)$): analogous to Cronbach's α for the reproducibility of element ordering.
- `Strata` ($(4G+1)/3$): number of distinguishable groups.
- Mean `Infit` and `Outfit`: average fit mean-squares across elements.
- `MisfitRate`: share of elements with $|\mathrm{ZSTD}| > 2$.
- Sparse-design diagnostics when `assignment = "sparse_linked"`: `MeanDesignDensity`, `MeanPlannedMissingRate`, `MeanLinkPersons`, `MeanMinCommonPersonsPerRaterPair`, `MaxZeroCommonRaterPairs`, and `MaxRaterPairsBelowTarget`.
- `SeverityRMSE`: root-mean-square error of recovered parameters vs the known truth **after facet-wise mean alignment**, so that the usual Rasch/MFRM location indeterminacy does not inflate recovery error. This quantity is reported only when the generator and fitted model target the same facet-parameter contract.
- `SeverityBias`: mean signed recovery error after the same alignment; values near zero are expected. This is likewise omitted when the generator/fitted-model contract does not align.

Interpreting output

Start with `summary(x)$design_summary`, then plot one focal metric at a time (for example rater `Separation` or criterion `SeverityRMSE`).

Higher separation/reliability is generally better, whereas lower `SeverityRMSE`, `MeanMisfitRate`, and `MeanElapsedSec` are preferable.

When choosing among designs, look for the point where increasing `n_person` or `raters_per_person` yields diminishing returns in separation and RMSE—this identifies the cost-effective design frontier. `ConvergedRuns / reps` should be near 1.0; low convergence rates indicate the design is too small for the chosen estimation method.

This is a Monte Carlo design-evaluation helper. It can visualize how separation, reliability, strata, RMSE, and fit-screen rates change when you vary person, rater, criterion, or assignment counts. For analytic generalizability-theory planning, pair observed variance-component review from `mfrm_generalizability()` with D-study projections from `mfrm_d_study()`.

References

The simulation logic follows the general Monte Carlo / operating-characteristic framework described by Morris, White, and Crowther (2019) and the ADEMP-oriented planning/reporting guidance summarized for psychology by Siepe et al. (2024). In `mfrmr`, `evaluate_mfrm_design()` is a practical many-facet design-planning wrapper rather than a direct reproduction of one published simulation study.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

See Also

`simulate_mfrm_data()`, summary.mfrm_design_evaluation, plot.mfrm_design_evaluation

Examples

```
sim_eval <- suppressWarnings(evaluate_mfrm_design(
  design = list(person = c(8, 12), rater = 2, criterion = 2, assignment = 1),
  reps = 1,
  maxit = 30,
  seed = 123
))
s_eval <- summary(sim_eval)
s_eval$design_summary[, c("Facet", "n_person", "MeanSeparation", "MeanSeverityRMSE")]
p_eval <- plot(sim_eval, facet = "Rater", metric = "separation", x_var = "n_person", draw = FALSE)
names(p_eval)
```

`evaluate_mfrm_diagnostic_screening` *Evaluate legacy and strict marginal diagnostic screening under controlled misfit scenarios*

Description

Evaluate legacy and strict marginal diagnostic screening under controlled misfit scenarios

Usage

```
evaluate_mfrm_diagnostic_screening(
  n_person = c(30, 50, 100),
  n_rater = c(4),
  n_criterion = c(4),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  scenarios = c("well_specified", "local_dependence"),
  local_dependence_sd = 0.8,
  local_dependence_facet = NULL,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  slopes = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  include_report = FALSE,
  report_include = c("fit", "diagnostics", "tables", "precision", "reporting"),
  report_style = c("qc", "apa", "validation", "reviewer", "technical"),
  seed = NULL
)
```

Arguments

`n_person`	Vector of person counts to evaluate.
`n_rater`	Vector of rater counts to evaluate.
`n_criterion`	Vector of criterion counts to evaluate.

raters_per_person
: Vector of rater assignments per person.

design
: Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (`n_person`, `n_rater`, `n_criterion`, `raters_per_person`), current public aliases implied by `sim_spec`, or role keywords (`person`, `rater`, `criterion`, `assignment`). Values may be vectors.

reps
: Number of replications per design condition and scenario.

scenarios
: Screening scenarios to evaluate. The current first release supports `"well_specified"`, `"local_dependence"`, and `"latent_misspecification"`, plus `"step_structure_misspecification"`

local_dependence_sd
: Standard deviation of the shared context effect injected in the `"local_dependence"` scenario.

local_dependence_facet
: Facet that receives the shared `Person x facet` dependence effect. Use `"criterion"`, `"rater"`, or an active public facet name. Defaults to the criterion-like facet.

score_levels
: Number of ordered score categories.

theta_sd
: Standard deviation of simulated person measures.

rater_sd
: Standard deviation of simulated rater severities.

criterion_sd
: Standard deviation of simulated criterion difficulties.

noise_sd
: Optional observation-level noise added to the linear predictor.

step_span
: Spread of step thresholds on the logit scale.

model
: Measurement model passed to `fit_mfrm()`. Bounded `GPCM` is supported with caveats as slope-aware screening sensitivity evidence.

step_facet
: Step facet passed to `fit_mfrm()` when `model = "PCM"` or `model = "GPCM"`.

slope_facet
: Slope facet passed to `fit_mfrm()` when `model = "GPCM"`. Defaults to the fitted step facet.

slopes
: Optional bounded-`GPCM` slope specification used by direct simulation calls when `sim_spec = NULL`.

maxit
: Maximum iterations passed to `fit_mfrm()`.

quad_points
: Quadrature points for the internal `MML` fit.

residual_pca
: Residual PCA mode passed to `diagnose_mfrm()`.

sim_spec
: Optional output from `build_mfrm_sim_spec()` or `extract_mfrm_sim_spec()` used as the base data-generating mechanism.

include_report
: Logical; if `TRUE`, each successful replicate also builds `mfrm_results()` and `mfrm_report()` and records the `report_index` readiness/signaling surface. This is intentionally opt-in because it repeats the comprehensive result-building workflow.

report_include
: `include` vector passed to `mfrm_results()` when `include_report = TRUE`.

report_style
: Report style passed to `mfrm_report()` when `include_report = TRUE`.

seed
: Optional seed for reproducible replications.

Details

This helper performs a compact Monte Carlo validation study for the package's current diagnostic architecture.

For each design condition and scenario, the function:

1. generates synthetic data with `simulate_mfrm_data()`
2. fits the model with `method = "MML"`
3. computes diagnostics with `diagnostic_mode = "both"`
4. stores legacy residual-screen metrics and strict marginal-fit metrics
5. optionally stores `mfrm_report()` `report_index` readiness signals
6. aggregates the results into `scenario_summary`, `performance_summary`, `report_signal_summary`, and `scenario_contrast`

The `"well_specified"` scenario uses the ordinary generator with no injected extra structure. The `"local_dependence"` scenario adds a shared `Person x facet` random effect, centered within the selected facet levels, so responses in the same context become correlated without changing the facet-level mean effect contract. The `"latent_misspecification"` scenario keeps the same marginal spread targets but replaces the normal person distribution with a centered bimodal empirical support distribution, while leaving the non-person facets on the original scale contract. The `"step_structure_misspecification"` scenario uses a `PCM` or bounded-`GPCM` generator with facet-specific threshold tables that intentionally mismatch the fitted step contract: `RSM` fits receive criterion-specific thresholds, and `PCM` / `GPCM` fits receive threshold structures indexed by the opposite non-person facet. For bounded `GPCM`, the generator and fit each keep `slope_facet == step_facet`; the misspecification is the generator-versus-fit step/slope facet mismatch.

This function is intentionally screening-oriented. The strict marginal branch remains exploratory in the current release, so the returned summaries should be used to compare relative sensitivity across scenarios rather than to claim calibrated inferential power. Bounded-`GPCM` rows add explicit `gpcm_boundary` caveats and should be read as slope-aware operating characteristics under the evaluated role-based design.

Value

An object of class `mfrm_diagnostic_screening` with:

- `design_grid`: evaluated design conditions, including public alias columns when applicable
- `results`: replicate-level screening metrics for each design and scenario
- `scenario_summary`: aggregated scenario-by-design screening summaries
- `performance_summary`: scenario-by-design screening-performance summary including runtime, agreement, Type I proxy, and sensitivity proxy columns
- `report_signal_summary`: optional scenario-by-design summary of `mfrm_report()` `report_index` availability, readiness, and review-signal counts when `include_report = TRUE`
- `scenario_contrast`: each misspecification scenario minus the well-specified baseline when the baseline scenario was evaluated
- `design_descriptor`: role-based design-variable metadata
- `planning_scope`: explicit record of the current planning contract

- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: combined planner-schema contract
- gpcm_boundary: bounded-GPCM caveat row when present
- settings: simulation and fitting settings
- ademp: simulation-study metadata
- notes: short interpretation notes

See Also

simulate_mfrm_data(), evaluate_mfrm_design(), diagnose_mfrm()

Examples

```
diag_eval <- evaluate_mfrm_diagnostic_screening(
  design = list(person = 10, rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  seed = 123
)
diag_eval$scenario_summary
diag_eval$scenario_contrast
```

---

evaluate_mfrm_recovery

Evaluate parameter recovery by repeated simulation and refitting

---

Description

Runs a compact parameter-recovery simulation study: generate data from a known ordered many-facet data-generating setup, refit the requested model, align estimates to the known truth where location indeterminacy requires it, and summarize bias, RMSE, MAE, correlation, and standard-error coverage.

Usage

```
evaluate_mfrm_recovery(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
```

```
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
  slope_facet = NULL,
  thresholds = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  sim_spec = NULL,
  fit_method = c("JML", "MML"),
  maxit = 25,
  quad_points = 7,
  include_person = TRUE,
  include_diagnostics = FALSE,
  diagnostic_fit_df_method = c("both", "engine", "facets"),
  seed = NULL
)
```

Arguments

n_person	Number of persons/respondents.
n_rater	Number of rater facet levels.
n_criterion	Number of criterion/item facet levels.
raters_per_person	Number of raters assigned to each person.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. When sim_spec = NULL, names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person) or role keywords (person, rater, criterion, assignment). For the currently exposed facet keys, the schema-only future branch input design$facets = c(person = ..., rater = ..., criterion = ...) is also accepted. Do not specify the same variable through both design and the scalar count arguments.
reps	Number of Monte Carlo replications.
score_levels	Number of ordered score categories.
theta_sd	Standard deviation of simulated person measures.
rater_sd	Standard deviation of simulated rater severities.
criterion_sd	Standard deviation of simulated criterion difficulties.
noise_sd	Optional observation-level noise added to the linear predictor.
step_span	Spread of step thresholds on the logit scale.
model	Measurement model recorded in the simulation setup. The current public generator supports RSM, PCM, and bounded GPCM.
step_facet	Step facet used when model = "PCM" and threshold values vary across levels. Currently "Criterion" and "Rater" are supported.

slope_facet — Slope facet used when model = "GPCM". The current bounded GPCM branch requires slope_facet == step_facet.

thresholds — Optional threshold specification. Use a numeric vector of common thresholds; a named list such as list(C01 = c(-1, 0, 1)); a numeric matrix with one row per StepFacet and one column per step; or a long data frame with columns StepFacet, Step/StepIndex, and Estimate.

slopes — Optional slope specification used when model = "GPCM". Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns SlopeFacet and Estimate. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean-one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact PCM reduction.

assignment — Assignment design. "crossed" means every person sees every rater; "rotating" uses a balanced rotating subset; "resampled" reuses person-level rater-assignment profiles stored in sim_spec; "sparse_linked" uses an incomplete rating design with optional linking persons; "skeleton" reuses an observed response skeleton stored in sim_spec, including optional Group/Weight columns when available. When omitted, the function chooses "crossed" if raters_per_person == n_rater, otherwise "rotating".

sparse_controls — Optional named list used when assignment = "sparse_linked". Supported entries are link_fraction, link_persons, link_raters_per_person, assignment_mode, and min_common_persons_per_rater_pair. See build_mfrm_sim_spec() for the same contract.

sim_spec — Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec(). When supplied, it defines the generator setup; direct scalar arguments are treated as legacy inputs and should generally be left at their defaults except for seed. Any custom public two-facet names recorded in sim_spec$facet_names are also carried into the simulated output and downstream planning helpers. If sim_spec stores an active latent-regression population generator, the returned object also carries the generated one-row-per-person background-data table needed to refit that population model later.

fit_method — Estimation method passed to fit_mfrm().

maxit — Maximum optimizer iterations passed to fit_mfrm().

quad_points — Quadrature points used when fit_method = "MML".

include_person — Logical. When TRUE, include person-measure recovery rows when the fitted object exposes person estimates.

include_diagnostics — Logical. When TRUE, run diagnose_mfrm() after each successful refit and retain facet-level fit/separation operating characteristics. These diagnostics are reported separately from recovery metrics and are not release-success criteria by themselves.

diagnostic_fit_df_method — Fit-ZSTD degrees-of-freedom convention used for optional diagnostic operating-characteristic summaries. Use "both" when reviewing FACETS-style df sensitivity.

seed — Optional random seed.

Details

This helper is deliberately narrower than `evaluate_mfrm_design()`. Design evaluation asks which design condition is operationally adequate; recovery simulation asks whether the fitted model recovers the known parameters under one explicit data-generating setup.

Location-like parameters (`Person`, non-person facets, and steps) are summarized after mean alignment within each replication and parameter group. This follows the usual Rasch/MFRM identification convention: adding a common constant to one location block should not be counted as recovery failure. Raw, unaligned errors are retained in `recovery` and summarized as `RawBias` / `RawRMSE`.

For bounded `GPCM`, supplied generator slopes are treated as relative discriminations and normalized to the same geometric-mean-one log-slope identification used by the fitter. Slope recovery is therefore summarized on the identified log-slope scale without an additional mean-alignment step. Direct data generation and refitting are supported, but broader GPCM design- planning claims remain outside the current package boundary.

Sparse linked generators are supported through `sim_spec` or direct `assignment = "sparse_linked"` plus `sparse_controls`. Their design-density and rater-link diagnostics are retained in `rep_overview`; recovery metrics remain parameter-recovery summaries and should not be read as evidence that a sparse linking design is adequate by itself.

The returned `ademp` component follows the simulation-study framing of Morris, White, and Crowther (2019) and the ADEMP planning/reporting template used in later simulation-study guidance.

Value

An object of class `mfrm_recovery_simulation` with components:

- `recovery`: row-level truth/estimate comparisons by replication.
- `recovery_summary`: parameter-type summaries across replications.
- `rep_overview`: replication-level convergence, timing, error status, and sparse-design diagnostics when applicable.
- `diagnostic_oc`: optional replication-by-facet fit/separation operating characteristics when `include_diagnostics = TRUE`.
- `diagnostic_oc_summary`: optional facet-level diagnostic operating- characteristic summary.
- `settings`: fitting and simulation settings.
- `ademp`: simulation-study metadata.

Typical workflow

1. Build a simulation specification with `build_mfrm_sim_spec()` or pass scalar generator arguments directly.
2. Run `evaluate_mfrm_recovery(...)` with a modest `reps` value for a smoke check, then increase `reps` for stable Monte Carlo summaries.
3. Inspect `summary(x)$recovery_summary` and the row-level `x$recovery` table.

See Also

`simulate_mfrm_data()`, `evaluate_mfrm_design()`, `fit_mfrm()`

Examples

```
rec <- evaluate_mfrm_recovery(
  n_person = 12,
  n_rater = 2,
  n_criterion = 2,
  reps = 1,
  maxit = 30,
  seed = 123
)
summary(rec)$recovery_summary[, c("ParameterType", "Facet", "RMSE", "Bias")]
```

---

evaluate_mfrm_signal_detection

Evaluate DIF power and bias-screening behavior under known simulated signals

---

Description

Evaluate DIF power and bias-screening behavior under known simulated signals

Usage

```
evaluate_mfrm_signal_detection(
  n_person = c(30, 50, 100),
  n_rater = c(4),
  n_criterion = c(4),
  raters_per_person = n_rater,
  design = NULL,
  reps = 10,
  group_levels = c("A", "B"),
  reference_group = NULL,
  focal_group = NULL,
  dif_level = NULL,
  dif_effect = 0.6,
  bias_rater = NULL,
  bias_criterion = NULL,
  bias_effect = -0.8,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  fit_method = c("JML", "MML"),
  model = c("RSM", "PCM", "GPCM"),
  step_facet = NULL,
```

```
  slope_facet = NULL,
  slopes = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  sim_spec = NULL,
  dif_method = c("residual", "refit"),
  dif_min_obs = 10,
  dif_p_adjust = "holm",
  dif_p_cut = 0.05,
  dif_abs_cut = 0.43,
  bias_max_iter = 2,
  bias_p_cut = 0.05,
  bias_abs_t = 2,
  seed = NULL
)
```

Arguments

n_person	Vector of person counts to evaluate.
n_rater	Vector of rater counts to evaluate.
n_criterion	Vector of criterion counts to evaluate.
raters_per_person	Vector of rater assignments per person.
design	Optional named design-grid override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases implied by sim_spec (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). Values may be vectors. The schema-only future branch input design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the currently exposed facet keys. Do not specify the same variable through both design and the scalar design-grid arguments.
reps	Number of replications per design condition.
group_levels	Group labels used for DIF simulation. The first two levels define the default reference and focal groups.
reference_group	Optional reference group label used when extracting the target DIF contrast.
focal_group	Optional focal group label used when extracting the target DIF contrast.
dif_level	Target criterion level for the true DIF effect. Can be an integer index or a criterion label such as "C04". Defaults to the last criterion level in each design.
dif_effect	True DIF effect size added to the focal group on the target criterion.
bias_rater	Target rater level for the true interaction-bias effect. Can be an integer index or a label such as "R04". Defaults to the last rater level in each design.
bias_criterion	Target criterion level for the true interaction-bias effect. Can be an integer index or a criterion label. Defaults to the last criterion level in each design.

bias_effect	True interaction-bias effect added to the target Rater x Criterion cell.
score_levels	Number of ordered score categories.
theta_sd	Standard deviation of simulated person measures.
rater_sd	Standard deviation of simulated rater severities.
criterion_sd	Standard deviation of simulated criterion difficulties.
noise_sd	Optional observation-level noise added to the linear predictor.
step_span	Spread of step thresholds on the logit scale.
fit_method	Estimation method passed to fit_mfrm().
model	Measurement model passed to fit_mfrm(). Bounded GPCM is supported with caveats as slope-aware signal-detection sensitivity evidence.
step_facet	Step facet passed to fit_mfrm() when model = "PCM" or model = "GPCM". When left NULL, the function inherits the generator step facet from sim_spec when available and otherwise defaults to "Criterion".
slope_facet	Slope facet passed to fit_mfrm() when model = "GPCM". Defaults to the fitted step facet.
slopes	Optional bounded-GPCM slope specification used by direct simulation calls when sim_spec = NULL.
maxit	Maximum iterations passed to fit_mfrm().
quad_points	Quadrature points for fit_method = "MML".
residual_pca	Residual PCA mode passed to diagnose_mfrm().
sim_spec	Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec() used as the base data-generating mechanism. When supplied, the design grid still varies n_person, n_rater, n_criterion, and raters_per_person, but latent spread, thresholds, and other generator settings come from sim_spec. The target DIF and interaction-bias signals specified in this function override any signal tables stored in sim_spec. If sim_spec stores an active latent-regression population generator, this helper currently requires fit_method = "MML" so each replication can refit the population model.
dif_method	Differential-functioning method passed to analyze_dff().
dif_min_obs	Minimum observations per group cell for analyze_dff().
dif_p_adjust	P-value adjustment method passed to analyze_dff().
dif_p_cut	P-value cutoff for counting a target DIF detection.
dif_abs_cut	Optional absolute contrast cutoff used when counting a target DIF detection. When omitted, the effective default is 0.43 for dif_method = "refit" and 0 (no additional magnitude cutoff) for dif_method = "residual".
bias_max_iter	Maximum iterations passed to estimate_bias().
bias_p_cut	P-value cutoff for counting a target bias screen-positive result.
bias_abs_t	Absolute t cutoff for counting a target bias screen-positive result.
seed	Optional seed for reproducible replications.

Details

This function performs Monte Carlo design screening for two related tasks: DIF detection via `analyze_dff()` and interaction-bias screening via `estimate_bias()`.

For each design condition (combination of `n_person`, `n_rater`, `n_criterion`, `raters_per_person`), the function:

1. Generates synthetic data with `simulate_mfrm_data()`
2. Injects one known Group $\times$ Criterion DIF effect (`dif_effect` logits added to the focal group on the target criterion)
3. Injects one known Rater $\times$ Criterion interaction-bias effect (`bias_effect` logits)
4. Fits and diagnoses the MFRM
5. Runs `analyze_dff()` and `estimate_bias()`
6. Records whether the injected signals were detected or screen-positive

Bounded-`GPCM` runs preserve the current package constraint `slope_facet == step_facet` within the generator and fitted model. The resulting DIF and bias rates are slope-aware screening summaries, not formal inferential power, alpha calibration, operational scoring, or arbitrary-facet planning evidence.

Detection criteria: A DIF signal is counted as "detected" when the target contrast has $p <$ `dif_p_cut` **and**, when an absolute contrast cutoff is in force, $|\text{Contrast}| \geq$ `dif_abs_cut`. For `dif_method = "refit"`, `dif_abs_cut` is interpreted on the logit scale. For `dif_method = "residual"`, the residual-contrast screening result is used and the default is to rely on the significance test alone.

Bias results are different: `estimate_bias()` reports `t` and `Prob.` as screening metrics rather than formal inferential quantities. Here, a bias cell is counted as **screen-positive** only when those screening metrics are available and satisfy

$p <$ `bias_p_cut` **and** $|t| \geq$ `bias_abs_t`.

Power is the proportion of replications in which the target signal was correctly detected. For DIF this is a conventional power summary. For bias, the primary summary is `BiasScreenRate`, a screening hit rate rather than formal inferential power.

False-positive rate is the proportion of non-target cells that were incorrectly flagged. For DIF this is interpreted in the usual testing sense. For bias, `BiasScreenFalsePositiveRate` is a screening rate and should not be read as a calibrated inferential alpha level.

Default effect sizes: `dif_effect = 0.6` logits corresponds to a moderate criterion-linked differential-functioning effect; `bias_effect = -0.8` logits represents a substantial rater-criterion interaction. Adjust these to match the smallest effect size of practical concern for your application.

This is again a **parametric simulation study**. The function does not estimate a new design directly from one observed dataset. Instead, it evaluates detection or screening behavior under user-specified design conditions and known injected signals.

If you want to approximate a real study, choose the design grid and simulation settings so that they reflect the empirical context of interest. For example, you may set `n_person`, `n_rater`, `n_criterion`, `raters_per_person`, and the latent-spread arguments to values motivated by an existing assessment program, then study how operating characteristics change as those design settings vary.

When sim_spec is supplied, the function uses it as the explicit data-generating mechanism for the latent spreads, thresholds, and assignment archetype, while still injecting the requested target DIF and bias effects for each design condition.

If that specification also stores a latent-regression population generator, each replication carries simulated one-row-per-person background data into the MML fit. This remains a screening-oriented Monte Carlo study; it is not a person-level posterior prediction for one observed sample.

Value

An object of class mfrm_signal_detection with:

- design_grid: evaluated design conditions. When sim_spec carries custom public facet names, matching design-variable alias columns are included alongside the canonical internal columns.
- results: replicate-level detection results, with the same design-variable alias columns when applicable.
- rep_overview: run-level status and timing, with the same design-variable alias columns when applicable.
- design_descriptor: role-based design-variable metadata used by planning summaries and plots
- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of which design variables remain mutable under the current simulation specification
- planning_schema: combined planner-schema contract bundling the role descriptor, scope boundary, and current mutability map
- gpcm_boundary: bounded-GPCM caveat row when a GPCM screening route is used
- settings: signal-analysis settings
- ademp: simulation-study metadata (aims, DGM, estimands, methods, performance measures)
- notes: short interpretation notes

References

The simulation logic follows the general Monte Carlo / operating-characteristic framework described by Morris, White, and Crowther (2019) and the ADEMP-oriented planning/reporting guidance summarized for psychology by Siepe et al. (2024). In mfrmr, evaluate_mfrm_signal_detection() is a many-facet screening helper specialized to DIF and interaction-bias use cases; it is not a direct implementation of one published many-facet Rasch simulation design.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

See Also

simulate_mfrm_data(), evaluate_mfrm_design(), analyze_dff(), analyze_dif(), estimate_bias()

Examples

```
sig_eval <- suppressWarnings(evaluate_mfrm_signal_detection(
  design = list(person = 8, rater = 2, criterion = 2, assignment = 1),
  reps = 1,
  maxit = 30,
  bias_max_iter = 1,
  seed = 123
))
s_sig <- summary(sig_eval)
s_sig$overview
```

`export_mfrm` *Export MFRM results to CSV files*

Description

Writes tidy CSV files suitable for import into spreadsheet software or further analysis in other tools.

Usage

```
export_mfrm(
  fit,
  diagnostics = NULL,
  output_dir = ".",
  prefix = "mfrm",
  tables = c("person", "facets", "summary", "steps", "measures"),
  overwrite = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm`.
`diagnostics`	Optional output from `diagnose_mfrm`. When provided, enriches facet estimates with SE, fit statistics, and writes the full measures table.
`output_dir`	Directory for CSV files. Created if it does not exist.
`prefix`	Filename prefix (default `"mfrm"`).
`tables`	Character vector of tables to export. Any subset of `"person"`, `"facets"`, `"summary"`, `"steps"`, `"measures"`. Default exports all available tables.
`overwrite`	If `FALSE` (default), refuse to overwrite existing files.

Value

Invisibly, a data.frame listing written files with columns `Table` and `Path`.

Exported files

`{prefix}_person_estimates.csv` Person ID, Estimate, SD.

`{prefix}_facet_estimates.csv` Facet, Level, Estimate, and optionally SE, Infit, Outfit, PTMEA when diagnostics supplied.

`{prefix}_fit_summary.csv` One-row model summary.

`{prefix}_step_parameters.csv` Step/threshold parameters.

`{prefix}_measures.csv` Full measures table (requires diagnostics).

Interpreting output

The returned data.frame tells you exactly which files were written and where. This is convenient for scripted pipelines where the output directory is created on the fly.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Optionally compute diagnostics with `diagnose_mfrm()` when you want enriched facet or measures exports.
3. Call `export_mfrm(...)` and inspect the returned `Path` column.

See Also

`fit_mfrm`, `diagnose_mfrm`, `as.data.frame.mfrm_fit`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
out <- export_mfrm(
  fit,
  diagnostics = diag,
  output_dir = tempdir(),
  prefix = "mfrmr_example",
  overwrite = TRUE
)
out$Table
```

export_mfrm_bundle *Export an analysis bundle for sharing or archiving*

Description

Export an analysis bundle for sharing or archiving

Usage

```
export_mfrm_bundle(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  population_prediction = NULL,
  unit_prediction = NULL,
  plausible_values = NULL,
  summary_tables = NULL,
  output_dir = ".",
  prefix = "mfrmr_bundle",
  include = c("core_tables", "checklist", "dashboard", "apa", "anchors", "manifest",
    "visual_summaries", "predictions", "summary_tables", "script", "html"),
  facet = NULL,
  include_person_anchors = FALSE,
  overwrite = FALSE,
  zip_bundle = FALSE,
  zip_name = NULL,
  data = NULL
)
```

Arguments

fit	Output from fit_mfrm() or run_mfrm_facets().
diagnostics	Optional output from diagnose_mfrm(). When NULL, diagnostics are reused from run_mfrm_facets() when available, otherwise computed with residual_pca = "none" (or "both" when visual summaries are requested).
bias_results	Optional output from estimate_bias() or a named list of bias bundles.
population_prediction	Optional output from predict_mfrm_population().
unit_prediction	Optional output from predict_mfrm_units().
plausible_values	Optional output from sample_mfrm_plausible_values().
summary_tables	Optional manuscript-summary bundle input. Can be build_summary_table_bundle() output, any object supported by build_summary_table_bundle(), or a named list of such objects. When NULL and "summary_tables" is requested in include,

	a default set is built from `fit`, `diagnostics`, `reporting_checklist()`, and `build_apa_outputs()`. Recovery-validation summaries can be supplied here to co-locate release-review appendix tables with a fit-based export bundle.
`output_dir`	Directory where files will be written.
`prefix`	File-name prefix.
`include`	Components to export. Supported values are `"core_tables"`, `"checklist"`, `"dashboard"`, `"apa"`, `"anchors"`, `"manifest"`, `"visual_summaries"`, `"predictions"`, `"summary_tables"`, `"script"`, and `"html"`.
`facet`	Optional facet for `facet_quality_dashboard()`.
`include_person_anchors`	If `TRUE`, include person measures in the exported anchor table.
`overwrite`	If `FALSE`, refuse to overwrite existing files.
`zip_bundle`	If `TRUE`, attempt to zip the written files into a single archive using `utils::zip()`. This is best-effort and may depend on the local R installation.
`zip_name`	Optional zip-file name. Defaults to `"{prefix}_bundle.zip"`.
`data`	Optional original analysis data frame. When supplied, `export_mfrm_bundle()` co-locates a CSV copy of the data alongside the replay script and updates the script's `read.csv()` path to point at it. The manifest's `input_hash` row for `data` is also computed against the user's untouched input so the recorded fingerprint matches what the replay script will load. Default `NULL` falls back to the legacy `your_data.csv` placeholder path.

Details

This function is the package-native counterpart to the app's download bundle. It reuses existing `mfrmr` helpers instead of reimplementing estimation or diagnostics.

Value

A named list with class `mfrm_export_bundle`.

Choosing exports

The `include` argument lets you assemble a bundle for different audiences:

- `"core_tables"` for analysts who mainly want CSV output.
- `"manifest"` for a compact analysis record.
- `"script"` for reproducibility and reruns. For latent-regression fits, this also writes the fit-level replay person-data sidecar when available.
- `"html"` for a light, shareable summary page. When replay sidecars are present, the HTML shows an artifact index for them rather than embedding the raw person-level replay table.
- `"summary_tables"` for manuscript-facing CSV exports of validated `summary()` surfaces and their compact indexes.
- `"visual_summaries"` when you want warning maps or residual PCA summaries to travel with the bundle.

Recommended presets

Common starting points are:

- minimal tables: `include = c("core_tables", "manifest")`
- reporting bundle: `include = c("core_tables", "checklist", "dashboard", "summary_tables", "html")`
- archival bundle: `include = c("core_tables", "manifest", "script", "visual_summaries", "html")`

Written outputs

Depending on `include`, the exporter can write:

- core CSV tables via `export_mfrm()`
- checklist CSVs via `reporting_checklist()`
- facet-dashboard CSVs via `facet_quality_dashboard()`
- APA text files via `build_apa_outputs()`
- manuscript-summary CSVs via `build_summary_table_bundle()`
- anchor CSV via `make_anchor_table()`
- manifest CSV/TXT via `build_mfrm_manifest()`
- visual warning/summary artifacts via `build_visual_summaries()`
- prediction/forecast CSVs via `predict_mfrm_population()`, `predict_mfrm_units()`, and `sample_mfrm_plausible_values()`
- a package-native replay script via `build_mfrm_replay_script()`
- for latent-regression fits, a replay-side person-data CSV paired with the replay script
- a lightweight HTML report that bundles the exported tables/text and, for replay sidecars, an artifact summary instead of raw person-level rows

For latent-regression fits, prediction-side artifacts can carry the fitted population-model scoring basis when you explicitly supply the corresponding prediction objects. `predict_mfrm_population()` remains the scenario-level forecast helper, whereas `predict_mfrm_units()` and `sample_mfrm_plausible_values()` are the scoring layer. To keep exports and replay scripts practical, large future-planning schemas from scenario-level population predictions are not flattened into `*_population_prediction_settings.csv` or ADeMP CSVs; the compact simulation specification files carry the replay-relevant settings instead.

For bounded `GPCM`, this exporter is available as a caveated partial bundle over supported diagnostics, report text, visual summaries, manifests, and replay scripts. The returned object and manifest include `gpcm_boundary`. Package-native bounded-`GPCM` scorefile export is available with caveats, while full FACETS-style score-side contract review and design forecasting remain outside this bundle contract.

Interpreting output

The returned object reports both high-level bundle status and the exact files written. In practice, `bundle$summary` is the direct status check, while `bundle$written_files` is the file inventory to inspect or hand off to other tools.

Typical workflow

1. Fit a model and compute diagnostics once.
2. Decide whether the audience needs tables only, or also a manifest, replay script, and HTML summary.
3. Call `export_mfrm_bundle()` with a dedicated output directory.
4. Inspect `bundle$written_files` or open the generated HTML file.

See Also

`build_mfrm_manifest()`, `build_mfrm_replay_script()`, `export_mfrm()`, `reporting_checklist()`, `export_summary_appendix()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bundle <- export_mfrm_bundle(
  fit,
  diagnostics = diag,
  output_dir = tempdir(),
  prefix = "mfrmr_bundle_example",
  include = c("core_tables", "manifest", "script", "html"),
  overwrite = TRUE
)
bundle$summary[, c("FilesWritten", "HtmlWritten", "ScriptWritten")]
head(bundle$written_files)
```

---

export_mfrm_results *Export a lightweight mfrm_results archive*

Description

`export_mfrm_results()` writes the contents of an existing `mfrm_results()` object to a small shareable folder. It is a results-download helper for the comprehensive first-screen workflow, not a new estimation, diagnostics, or validation step.

Usage

```
export_mfrm_results(
  x,
  output_dir = ".",
  prefix = "mfrmr_results",
  include = "default",
  overwrite = FALSE,
  zip_bundle = FALSE,
```

```
  zip_name = NULL,
  plot_width = 1200,
  plot_height = 900,
  plot_res = 144
)
```

Arguments

x	An mfrm_results() object.
output_dir	Directory where files should be written.
prefix	File-name prefix. Non-alphanumeric characters are converted to underscores.
include	Export components. "default" expands to "summary", "tables", "html", "rds", "replay", and "manifest". Add "report" to write mfrm_report() tables plus Markdown and HTML; add "plots" to write available plot routes as PNG files, or use "all".
overwrite	Logical; if FALSE, existing files stop the export.
zip_bundle	Logical; if TRUE, create a best-effort zip archive of the written files.
zip_name	Optional zip file name. When omitted, {prefix}_mfrm_results.zip is used.
plot_width, plot_height, plot_res	PNG device settings used when include contains "plots".

Details

The helper writes:

- summary CSVs from summary(x) such as overview, status, triage, plot routes, next actions, mapping, and replay-code lines;
- collected x$tables as CSV files;
- optional report artifacts from mfrm_report(x), including report-index, evidence-summary, and reporting-template CSVs plus Markdown and HTML;
- a lightweight HTML report equivalent to mfrm_results(x, output = "html") for the already-created object;
- an .rds copy of the mfrm_results object;
- a replay .R scaffold from x$input$reproducible_code;
- a written-files manifest and compact export summary.

Plot export is intentionally optional because some plot routes can be comparatively slow or require richer graphics devices. Plot failures are recorded in the returned plot_errors table rather than stopping the export.

Value

An mfrm_results_export object with summary, written_files, plot_errors, and zip status fields.

See Also

mfrm_results(), launch_mfrmr_viewer(), export_mfrm_bundle(), export_summary_appendix()

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:6], , drop = FALSE]
fit <- fit_mfrm(toy_small, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
res <- mfrm_results(fit, include = c("fit", "diagnostics", "tables"))

exported <- export_mfrm_results(
  res,
  output_dir = tempdir(),
  prefix = "mfrmr_results_example",
  overwrite = TRUE
)
exported$summary[, c("FilesWritten", "CsvWritten", "HtmlWritten")]
```

---

export_summary_appendix

Export manuscript appendix tables from validated summary surfaces

---

Description

Export manuscript appendix tables from validated summary surfaces

Usage

```
export_summary_appendix(
  x,
  output_dir = ".",
  prefix = "mfrmr_appendix",
  include_html = TRUE,
 preset = c("all", "recommended", "compact", "methods", "results", "diagnostics",
    "reporting"),
  overwrite = FALSE,
  zip_bundle = FALSE,
  zip_name = NULL,
  digits = 3,
  top_n = 10,
  preview_chars = 160
)
```

Arguments

x	A supported summary() source, a prebuilt build_summary_table_bundle() result, or a named list of such objects.
output_dir	Directory where files will be written.
prefix	File-name prefix for written artifacts.
include_html	If TRUE, also write a lightweight HTML appendix page.
preset	Appendix table-selection preset: "all" keeps every returned summary table, "recommended" keeps manuscript-facing summary tables while dropping bridge-only or preview-only surfaces, and "compact" keeps a smaller reviewer-facing subset. Section-aware presets "methods", "results", "diagnostics", and "reporting" keep only the returned tables classified to those appendix sections in the summary-table catalog.
overwrite	If FALSE, refuse to overwrite existing files.
zip_bundle	If TRUE, attempt to zip the written appendix artifacts.
zip_name	Optional zip-file name. Defaults to "{prefix}_appendix.zip".
digits	Digits forwarded when raw objects must be normalized through build_summary_table_bundle().
top_n	Row cap forwarded when raw objects must be normalized through build_summary_table_bundle().
preview_chars	Character cap forwarded when APA-output summaries must be normalized through build_summary_table_bundle().

Details

This helper is the narrow public bridge from validated summary() surfaces to manuscript appendix artifacts. It accepts the same reporting objects that build_summary_table_bundle() supports, exports their table bundles as CSV, and optionally assembles a lightweight HTML appendix page.

Fit-level caveats are exported through the analysis_caveats role, and pre-fit score-support caveats are exported through the score_category_caveats role. Both roles are classified as diagnostics, so they remain available under "recommended" and "diagnostics" presets when the source summary contains caveat rows.

Precision-review summaries keep fit_separation_basis in the exported precision-review role so fit, ZSTD, separation/reliability/strata, and package QC thresholds can be reported without turning them into release or recovery success gates. Fit-measure and FACETS fit-review summaries keep df/ZSTD sensitivity and optional external FACETS matching tables in the same precision-review lane.

Parameter-recovery studies can be exported by passing evaluate_mfrm_recovery() or assess_mfrm_recovery() output directly. The exported bundle keeps the ADEMP-style simulation basis, recovery metrics, replication status, adequacy checklist, thresholds, and next actions in separate appendix-ready tables.

Recovery-validation summaries from the packaged validation protocol can be exported by passing summary(validation), including top-line release decisions, condition notes, diagnostic notes, and domain decisions.

Unlike export_mfrm_bundle(), this helper does not require a fitted model. It is intended for the stage where compact reporting summaries already exist and the task is to hand off appendix-ready tables, catalogs, and reporting maps.

Value

A named list of class `mfrm_summary_appendix_export` with:

- `summary`
- `written_files`
- `selection_summary`
- `selection_table_summary`
- `selection_section_table_summary`
- `selection_handoff_table_summary`
- `selection_handoff_preset_summary`
- `selection_handoff_summary`
- `selection_handoff_bundle_summary`
- `selection_handoff_role_summary`
- `selection_handoff_role_section_summary`
- `selection_role_summary`
- `selection_section_summary`
- `selection_catalog`
- `settings`
- `notes`

Typical workflow

1. Build `summary(...)` objects from fit, diagnostics, data description, reporting checklist, or APA outputs.
2. Call `export_summary_appendix(...)` on one object or a named list.
3. Hand off the written CSV/HTML appendix artifacts to manuscript or QA workflows.

See Also

`build_summary_table_bundle()`, `export_mfrm_bundle()`, `apa_table()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
appendix <- export_summary_appendix(
  list(fit = fit, diagnostics = diag),
  output_dir = tempdir(),
  prefix = "mfrmr_appendix_example",
  include_html = TRUE,
  overwrite = TRUE
)
appendix$summary
```

`extract_mfrm_sim_spec` *Derive a simulation specification from a fitted MFRM object*

Description

Derive a simulation specification from a fitted MFRM object

Usage

```
extract_mfrm_sim_spec(
  fit,
  assignment = c("auto", "crossed", "rotating", "resampled", "skeleton"),
  latent_distribution = c("normal", "empirical"),
  source_data = NULL,
  person = NULL,
  group = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`assignment`	Assignment design to record in the returned specification. Use `"resampled"` to reuse empirical person-level rater-assignment profiles from the fitted data, or `"skeleton"` to reuse the observed person-by-facet design skeleton from the fitted data.
`latent_distribution`	Latent-value generator to record in the returned specification. `"normal"` stores spread summaries for parametric draws; `"empirical"` additionally activates centered empirical resampling from the fitted person/rater/criterion estimates.
`source_data`	Optional original source data used to recover additional non-calibration columns, currently person-level `group` labels, when building a fit-derived observed response skeleton.
`person`	Optional person column name in `source_data`. Defaults to the person column recorded in `fit`.
`group`	Optional group column name in `source_data` to merge into the returned `design_skeleton` as person-level metadata.

Details

`extract_mfrm_sim_spec()` uses a fitted model as a practical starting point for later simulation studies. It extracts:

- design counts from the fitted data
- empirical spread of person and facet estimates
- optional empirical support values for semi-parametric draws

- fitted threshold values
- either a simplified assignment summary (`"crossed"` / `"rotating"`), empirical resampled assignment profiles (`"resampled"`), or an observed response skeleton (`"skeleton"`, optionally carrying `Group`/`Weight`)
- when the fit used the latent-regression branch, the fitted `population_formula`, coefficient vector, residual variance, and the stored person-level covariate table, including model-matrix xlevel and contrast provenance for categorical covariates

This is intended as a **fit-derived parametric starting point**, not as a claim that the fitted object perfectly recovers the true data-generating mechanism. Users should review and, if necessary, edit the returned specification before using it for design planning.

First-release `GPCM` fits are now supported here for direct data generation and parameter-recovery checks, provided that the returned simulation specification stores both a threshold table and a parallel slope table. The same fit-derived specification can feed caveated role-based design evaluation, population forecasting, and fit-based report/export bundles. Diagnostic/signal-detection design screening, full FACETS score-side contract review, posterior predictive checks, and heavy backend extensions remain outside the bounded-`GPCM` boundary until those downstream contracts are widened explicitly.

If you want to carry person-level group labels into a fit-derived observed response skeleton, provide the original `source_data` together with `person` and `group`. Group labels are treated as person-level metadata and are checked for one-label-per-person consistency before being merged.

Value

An object of class `mfrm_sim_spec`.

Interpreting output

The returned object is a simulation specification, not a prediction about one future sample. It captures one convenient approximation to the observed design and estimated spread in the fitted run.

See Also

`build_mfrm_sim_spec()`, `simulate_mfrm_data()`

Examples

```
toy <- simulate_mfrm_data(
  n_person = 8,
  n_rater = 3,
  n_criterion = 2,
  seed = 123
)
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
spec <- extract_mfrm_sim_spec(fit, latent_distribution = "empirical")
spec$assignment
spec$model
head(spec$threshold_table)
```

`facets_chisq_table` *Build facet variability diagnostics with fixed/random reference tests*

Description

Build facet variability diagnostics with fixed/random reference tests

Usage

```
facets_chisq_table(
  fit,
  diagnostics = NULL,
  fixed_p_max = 0.05,
  random_p_max = 0.05,
  top_n = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`fixed_p_max`	Warning cutoff for fixed-effect chi-square p-values.
`random_p_max`	Warning cutoff for random-effect chi-square p-values.
`top_n`	Optional maximum number of facet rows to keep.

Details

This helper summarizes facet-level variability with fixed and random chi-square indices for spread and heterogeneity checks.

Value

A named list with:

- `table`: facet-level chi-square diagnostics
- `summary`: one-row summary
- `thresholds`: applied p-value thresholds

Interpreting output

- `table`: facet-level fixed/random chi-square and p-value flags.
- `summary`: number of significant facets and overall magnitude indicators.
- `thresholds`: p-value criteria used for flagging.

Use this table together with inter-rater and displacement diagnostics to distinguish global facet effects from local anomalies.

Typical workflow

1. Run `facets_chisq_table(fit, ...)`.
2. Inspect `summary(chi)` then facet rows in `chi$table`.
3. Visualize with `plot_facets_chisq()`.

Output columns

The `table` data.frame contains:

Facet Facet name.

Levels Number of estimated levels in this facet.

MeanMeasure, SD Mean and standard deviation of level measures.

FixedChiSq, FixedDF, FixedProb Fixed-effect chi-square test (null hypothesis: all levels equal). Significant result means the facet elements differ more than measurement error alone.

RandomChiSq, RandomDF, RandomProb, RandomVar Random-effect test (null hypothesis: variation equals that of a random sample from a single population). Significant result suggests systematic heterogeneity beyond sampling variation.

FixedFlag, RandomFlag Logical flags for significance.

See Also

`diagnose_mfrm()`, `interrater_agreement_table()`, `plot_facets_chisq()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
chi <- facets_chisq_table(fit)
summary(chi)
p_chi <- plot(chi, draw = FALSE)
p_chi$data$plot
```

---

facets_feature_coverage

FACETS Feature Coverage Matrix

Description

`facets_feature_coverage()` summarizes how the current `mfrmr` release maps the main FACETS output-table, output-file, and graph-menu surface to package functions.

Use this helper before migration work when you need a public, user-facing answer to three questions:

- which FACETS outputs have a close `mfrmr` route,
- which outputs are only partially covered by structured R objects,
- which FACETS-specific outputs are not implemented or intentionally outside the current package scope.

Usage

```
facets_feature_coverage(
  status = c("all", "implemented", "partial", "not_implemented", "not_targeted")
)
```

Arguments

status	Which rows to return. "all" returns the full matrix. Other values filter by the Status column.

Details

The matrix is based on the FACETS 64-bit output index, which lists output Tables 1–14, DIF/bias plots, R/Web plots, output files, and graph-menu curves. mfrmr intentionally prioritizes structured R tables and reusable plot data over exact FACETS line-printer output.

Status meanings:

- implemented: a package-native route covers the substantive output.
- partial: the concept is covered, but not the full FACETS formatting, option surface, file type, or external integration.
- not_implemented: a FACETS feature has no direct package-native route in the current release.
- not_targeted: the feature is tied to FACETS UI, Web/Excel handoff, or another external program format and is not a release goal.

Value

A data.frame with columns:

- FACETSArea
- FACETSFeature
- FACETSReference
- mfrmrRoute
- Status
- Scope
- GapOrBoundary
- Priority

References

Linacre, J. M. (2026). *A user's guide to FACETS, version 4.5.0*. Output tables - files - plots - graphs: https://www.winsteps.com/facetman64/outputtableindex.htm.

See Also

facets_positioning_guide(), mfrmr_output_guide(), facets_fit_df_guide(), read_facets_fit_table(), facets_fit_review(), gpcm_capability_matrix()

Examples

```
facets_feature_coverage()
facets_feature_coverage("partial")
facets_feature_coverage("not_implemented")
```

`facets_fit_df_guide` *Guide FACETS-style fit df and ZSTD standardization*

Description

`facets_fit_df_guide()` gives a compact user-facing guide to the degrees of freedom and ZSTD standardization choices used when comparing mfrmr fit output with FACETS-style fit tables.

Usage

```
facets_fit_df_guide(include_references = TRUE)
```

Arguments

`include_references`
: If `TRUE`, include source-reference rows for the FACETS/Winsteps documentation and Rasch measurement texts that motivate the guide.

Details

The guide separates mean-square size from ZSTD standardization. Infit and outfit MnSq values answer how large the residual noise or predictability signal is. ZSTD values standardize those MnSq values using a degrees-of- freedom convention and a Wilson-Hilferty-style transformation, so ZSTD can differ even when the underlying MnSq values are nearly identical.

Two boundaries sit upstream of any df comparison. First, the residual basis: `method = "MML"` fits evaluate residuals at shrunken EAP person measures, whereas FACETS evaluates them at JMLE estimates, so MnSq values themselves can differ before any standardization is applied; refit with `method = "JML"` when the comparison requires a JMLE-style residual basis. Second, small df: `mfrmr` returns `NA` ZSTD when `df < 1` because the Wilson-Hilferty transformation is numerically unstable there, while FACETS/Winsteps under `WHEXACT` can continue with a linear approximation, so sparse cells can show `NA` against a finite external value without indicating a fit difference.

Value

A bundle of class `mfrm_facets_fit_df_guide` with:

- `summary`: one-row scope summary
- `formula_guide`: formulas and package columns
- `column_guide`: where engine and FACETS-style columns appear
- `decision_guide`: recommended comparison steps
- `interpretation_guide`: how to read common difference patterns
- `references`: optional source-reference rows
- `settings`: guide metadata

See Also

diagnose_mfrm(), fit_measures_table(), facets_fit_review()

Examples

```
facets_fit_df_guide()
facets_fit_df_guide()$decision_guide
```

---

facets_fit_review *Review fit standardization against FACETS-style ZSTD conventions*

Description

Review fit standardization against FACETS-style ZSTD conventions

Usage

```
facets_fit_review(
  fit,
  diagnostics = NULL,
  facets_fit = NULL,
  facet_col = NULL,
  level_col = NULL,
  mnsq_tolerance = 0.01,
  external_zstd_tolerance = 0.05,
  df_tolerance = 0.5,
  df_zstd_tolerance = 0.05,
  df_zstd_large_shift = 0.5,
  df_ratio_tolerance = 0.05
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). If it does not contain FACETS-style fit columns, diagnostics are recomputed with fit_df_method = "both" and residual_pca = "none".
facets_fit	Optional external FACETS fit table, or a list of such tables. The helper matches rows by Facet and Level; a person-only table with a Person column is also accepted.
facet_col, level_col	Optional explicit column names for the external FACETS table when automatic detection is not sufficient.
mnsq_tolerance, external_zstd_tolerance, df_tolerance	Numeric tolerances used to classify external FACETS-vs-mfrmr differences.

df_zstd_tolerance
: Smallest absolute engine-vs-FACETS-style ZSTD difference treated as interpretively visible rather than rounding noise in df_sensitivity. Default 0.05.

df_zstd_large_shift
: Absolute engine-vs-FACETS-style ZSTD difference labeled large_zstd_shift when the |ZSTD| flag status is unchanged. Default 0.5.

df_ratio_tolerance
: Relative df-difference tolerance used to classify the internal engine-vs-FACETS-style df difference; for example, 0.05 means a 5 percent df difference.

Details

This helper separates two questions that are often conflated when comparing mfrmr output with FACETS:

- how much the package-native engine ZSTD changes when the same MnSq values are standardized with the FACETS/Wright-Masters fourth-moment df convention;
- when an external FACETS table is supplied, whether the FACETS-reported rows match mfrmr's FACETS-style companion columns closely enough for practical reporting.

The review is row-matched by Facet and Level. It treats MnSq, ZSTD, and df differences separately because FACETS documentation makes the df convention and Wilson-Hilferty/WHEXACT handling central to ZSTD interpretation.

Two upstream boundaries also apply. For method = "MML" fits, residuals are evaluated at shrunken EAP person measures while FACETS uses JMLE estimates, so MnSq itself can differ before standardization; refit with method = "JML" for a JMLE-style residual basis. And mfrmr withholds ZSTD as NA when the applicable df falls below 1 (Wilson-Hilferty instability), while FACETS under WHEXACT can report a value on the same sparse cell; such NA-vs-finite pairs are availability differences, not fit differences. Both notes are repeated in the returned guidance table.

Value

An mfrm_facets_fit_review bundle with:

- summary: one-row overview of internal and external comparison counts
- standardization: the fit-standardization guide from diagnostics
- df_sensitivity: engine-vs-FACETS-style df/ZSTD comparison using the same row-level status taxonomy as fit_measures_table()$df_sensitivity
- df_sensitive: subset of df_sensitivity whose df convention changes the |ZSTD| flag or materially changes ZSTD interpretation
- df_sensitivity_summary: counts by df-sensitivity status
- external_table_quality: completeness and duplicate-key review for the supplied FACETS fit table
- external_comparison: optional external FACETS-vs-mfrmr comparison
- df_conversion_guide: formulas, column map, and comparison decisions for FACETS-style df/ZSTD review
- guidance: interpretation notes
- settings: tolerances and review metadata

See Also

diagnose_mfrm(), facets_output_contract_review(), mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
review <- facets_fit_review(fit)
summary(review)
```

---

facets_output_contract_review

Build a FACETS output-contract review

Description

Build a FACETS output-contract review

Usage

```
facets_output_contract_review(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  branch = c("facets", "original"),
  contract_file = NULL,
  include_metrics = TRUE,
  top_n_missing = 15L
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). If omitted, diagnostics are computed internally with residual_pca = "none".
bias_results	Optional output from estimate_bias(). If omitted and at least two facets exist, a 2-way bias run is computed internally.
branch	Contract branch. "facets" checks legacy-compatible columns. "original" adapts branch-sensitive contracts to the package's compact naming.
contract_file	Optional path to a custom contract CSV.
include_metrics	If TRUE, run additional numerical consistency checks.
top_n_missing	Number of lowest-coverage contract rows to keep in missing_preview.

Details

This function checks produced report components against a FACETS-style output-contract specification (`inst/references/facets_column_contract.csv`) and returns:

- column-level coverage per contract row
- table-level coverage summaries
- optional metric-level consistency checks

It is intended for output-contract QA and regression review. It does not establish external validity or software equivalence beyond the specific schema/metric contract encoded in the contract file.

Value

An object of class `mfrm_facets_contract_review` with:

- `overall`: one-row output-contract review summary
- `column_summary`: coverage summary by table ID
- `column_review`: row-level output-contract review
- `missing_preview`: lowest-coverage rows
- `metric_summary`: one-row metric-check summary
- `metric_by_table`: metric-check summary by table ID
- `metric_checks`: row-level metric checks
- `settings`: branch/contract metadata

Bounded GPCM boundary

This helper remains blocked for bounded `GPCM` fits in 0.2.1. The FACETS output contract includes score-side rows whose measure-to-score and uncertainty semantics are validated for the current Rasch-family route, not for free-discrimination bounded `GPCM`. Use `gpcm_capability_matrix()` before routing a bounded `GPCM` fit into score-side compatibility-output helpers.

Coverage interpretation in `overall`:

- `MeanColumnCoverage` and `MinColumnCoverage` are computed across all contract rows (unavailable rows count as 0 coverage).
- `MeanColumnCoverageAvailable` and `MinColumnCoverageAvailable` summarize only rows whose source component is available.

`summary(out)` is supported through `summary()`. `plot(out)` is dispatched through `plot()` for class `mfrm_facets_contract_review` (`type = "column_coverage"`, `"table_coverage"`, `"metric_status"`, `"metric_by_table"`).

Interpreting output

- `overall`: high-level output-contract coverage and metric-check pass rates.
- `column_summary` / `column_review`: where output-schema mismatches occur.
- `metric_summary` / `metric_checks`: numerical consistency checks tied to the current contract.
- `missing_preview`: direct path to unresolved output-contract gaps.

Typical workflow

1. Run `facets_output_contract_review(fit, branch = "facets")`.
2. Inspect `summary(contract_review)` and `missing_preview`.
3. Patch upstream table builders, then rerun the output-contract review.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `build_fixed_reports()`, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
contract_review <- facets_output_contract_review(fit, diagnostics = diag, branch = "facets")
summary(contract_review)
p <- plot(contract_review, draw = FALSE)
```

---

facets_output_file_bundle

Build a legacy-compatible output-file bundle (GRAPH= / SCORE=)

Description

Build a legacy-compatible output-file bundle (GRAPH= / SCORE=)

Usage

```
facets_output_file_bundle(
  fit,
  diagnostics = NULL,
  include = c("graph", "score"),
  theta_range = c(-6, 6),
  theta_points = 241,
  digits = 4,
  score_se_method = c("both", "native", "score_side", "none"),
  include_fixed = FALSE,
  fixed_max_rows = 400,
  write_files = FALSE,
  output_dir = NULL,
  file_prefix = "mfrmr_output",
  overwrite = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() (used for score file).
include	Output components to include: "graph" and/or "score".
theta_range	Theta/logit range for graph coordinates.
theta_points	Number of points on the theta grid for graph coordinates.
digits	Rounding digits for numeric fields.
score_se_method	For bounded GPCM scorefile exports, which observation-level score uncertainty columns to compute. "both" (default) includes native structural expected-score SEs and score-side delta-method SEs; "native" includes only the structural expected-score route; "score_side" includes only the score-side delta route; "none" records explicit not_requested status columns.
include_fixed	If TRUE, include fixed-width text mirrors of output tables.
fixed_max_rows	Maximum rows shown in fixed-width text blocks.
write_files	If TRUE, write selected outputs to files in output_dir.
output_dir	Output directory used when write_files = TRUE.
file_prefix	Prefix used for output file names.
overwrite	If FALSE, existing output files are not overwritten.

Details

Legacy-compatible output files often include:

- graph coordinates for Table 8 curves (GRAPH= / Graphfile=), and
- observation-level modeled score lines (SCORE=-style inspection).

This helper returns both as data frames and can optionally write CSV/fixed-width text files to disk.

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_output_bundle (type = "graph_expected", "score_residuals", "obs_probability", "score_se").

Value

A named list including:

- graphfile / graphfile_syntactic when "graph" is requested
- scorefile when "score" is requested
- graphfile_fixed / scorefile_fixed when include_fixed = TRUE
- written_files when write_files = TRUE
- settings: applied options

Interpreting output

- `graphfile`: legacy-compatible wide curve coordinates (human-readable labels).
- `graphfile_syntactic`: same curves with syntactic column names for programmatic use.
- `scorefile`: observation-level observed/expected/residual diagnostics.
- `written_files`: traceability record of files produced when `write_files = TRUE`.

For reproducible pipelines, prefer `graphfile_syntactic` and keep `written_files` in run logs.

Preferred route for new analyses

For new scripts, prefer `category_curves_report()` or `category_structure_report()` for scale outputs, then use `export_mfrm_bundle()` for file handoff. Use `facets_output_file_bundle()` only when a legacy-compatible graphfile or scorefile contract is required.

Bounded GPCM boundary

For bounded `GPCM`, graph output and package-native scorefile output are available with caveats. `include = "score"` returns observation-level fitted expected score, residual, standardized residual, observed-category probability, GPCM slope fields, and native structural delta-method expected-score uncertainty and/or score-side delta-method SEs when the required MML diagnostics are available. Use `score_se_method` to choose `"both"` (default), `"native"`, `"score_side"`, or `"none"`. The scorefile also carries explicit score-side caveat columns. It is not a FACETS score-side equivalence file, does not export FACETS-equivalent score-side standard errors, and does not establish an operational score-scale decision. Use `gpcm_score_side_contract()` and `gpcm_capability_matrix()` for the current scope.

Typical workflow

1. Fit and diagnose model.
2. Generate bundle with `include = c("graph", "score")`.
3. Validate with `summary(out)` / `plot(out)`.
4. Export with `write_files = TRUE` for reporting handoff.

See Also

`category_curves_report()`, `diagnose_mfrm()`, `unexpected_response_table()`, `export_mfrm_bundle()`, mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- facets_output_file_bundle(fit, diagnostics = diagnose_mfrm(fit, residual_pca = "none"))
summary(out)
p_out <- plot(out, draw = FALSE)
p_out$data$plot
```

`facets_positioning_guide` *FACETS Positioning Guide*

Description

`facets_positioning_guide()` gives user-facing wording for the relationship between `mfrmr` and FACETS. Use it when a report, migration note, or methods appendix must make clear that `mfrmr` is not a FACETS numerical clone.

Usage

```
facets_positioning_guide()
```

Details

The guide separates four ideas that are easy to conflate:

- estimation authority: fitted values come from `mfrmr` unless external FACETS output is explicitly supplied;
- compatibility purpose: FACETS-style names and files are transition, handoff, and report-organization surfaces;
- external comparison: FACETS comparisons require a supplied external table and should separate MnSq differences from df/ZSTD convention differences;
- extension surface: native R tables, plot data, GPCM diagnostics, network views, and G/D-study helpers are package extensions, not promises of FACETS menu-level reproduction.

Value

A data.frame with columns:

- `Topic`
- `Position`
- `RecommendedWording`
- `PrimaryRoute`

See Also

`facets_feature_coverage()`, `mfrmr_output_guide()`, `read_facets_fit_table()`, `facets_fit_review()`

Examples

```
facets_positioning_guide()
```

facet_quality_dashboard *Facet-quality dashboard for facet-level screening*

Description

Build a compact dashboard for one facet at a time, combining facet severity, misfit, central-tendency screening, and optional bias counts.

Usage

```
facet_quality_dashboard(
  fit,
  diagnostics = NULL,
  facet = NULL,
  bias_results = NULL,
  severity_warn = 1,
  misfit_warn = NULL,
  central_tendency_max = 0.25,
  bias_count_warn = 1L,
  bias_abs_t_warn = 2,
  bias_abs_size_warn = 0.5,
  bias_p_max = 0.05
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
facet	Optional facet name. When NULL, the function tries to infer a rater-like facet and otherwise falls back to the first modeled facet.
bias_results	Optional output from estimate_bias() or a named list of such outputs. Non-matching bundles are skipped quietly.
severity_warn	Absolute estimate cutoff used to flag severity outliers.
misfit_warn	Mean-square cutoff used to flag misfit. Values above this cutoff or below its reciprocal are flagged.
central_tendency_max	Absolute estimate cutoff used to flag central tendency. Levels near zero are marked.
bias_count_warn	Minimum flagged-bias row count required to flag a level.
bias_abs_t_warn	Absolute t cutoff used when deriving bias-row flags from a raw bias bundle.

`bias_abs_size_warn`
: Absolute bias-size cutoff used when deriving bias-row flags from a raw bias bundle.

`bias_p_max`
: Probability cutoff used when deriving bias-row flags from a raw bias bundle.

Details

The dashboard screens individual facet elements across four complementary criteria:

- **Severity**: elements with $|\text{Estimate}| >$ `severity_warn` logits are flagged as unusually harsh or lenient.
- **Misfit**: elements with Infit or Outfit MnSq outside the acceptance band are flagged. The band defaults to the package pair returned by `mfrm_misfit_thresholds()` (Linacre 0.5-1.5); pass `misfit_warn = 1.5` to keep the older symmetric $[1/$`misfit_warn`, `misfit_warn`$]$ form (0.67-1.5).
- **Central tendency**: elements with $|\text{Estimate}| <$ `central_tendency_max` logits are flagged. Near-zero estimates may indicate a rater who avoids extreme categories, producing artificially narrow score ranges.
- **Bias**: elements involved in $\geq$ `bias_count_warn` screen-positive interaction cells (from `estimate_bias()`) are flagged.

A **flag density** score counts how many of the four criteria each element triggers. Elements flagged on multiple criteria warrant priority review (e.g., rater retraining, data exclusion).

Default thresholds are designed for moderate-stakes rating contexts. Adjust for your application: stricter thresholds for high-stakes certification, more lenient for formative assessment.

Value

An object of class `mfrm_facet_dashboard` (also inheriting from `mfrm_bundle` and `list`). The object summarizes one target facet: `overview` reports the facet-level screening totals, `summary` provides aggregate estimates and flag counts, `detail` contains one row per facet level with the computed screening indicators, `ranked` orders levels by review priority, `flagged` keeps only levels requiring follow-up, `bias_sources` records which bias-result bundles contributed to the counts, `settings` stores the resolved thresholds, and `notes` gives short interpretation messages about how to read the dashboard.

Output

The returned object is a bundle-like list with class `mfrm_facet_dashboard` and components:

- `facet`: character scalar naming the dashboard's target facet
- `facet_source`: character scalar describing whether the target facet was inferred from the fit configuration or supplied explicitly
- `overview`: one-row structural overview
- `summary`: one-row screening summary
- `detail`: level-level detail table
- `ranked`: detail ordered by flag density / severity

- flagged: flagged levels only
- bias_sources: per-bundle bias aggregation metadata
- settings: resolved threshold settings
- notes: short interpretation notes
- diagnostics: the mfrm_diagnostics bundle the dashboard was built from (echoed for downstream helpers that need to traverse the same diagnostics object)
- bias_results: the mfrm_bias bundle (or list of bundles) when bias_results was supplied; NULL otherwise

See Also

diagnose_mfrm(), estimate_bias(), plot_qc_dashboard()

Examples

```
toy <- load_mfrmr_data("example_core")
toy <- toy[toy$Person %in% unique(toy$Person)[1:8], ]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
dash <- facet_quality_dashboard(fit, diagnostics = diag)
summary(dash)
```

---

facet_small_sample_review

Review per-facet-level sample adequacy

Description

Reports per-level observation counts, SE, and fit statistics for every level of every facet in a fitted MFRM model, and classifies each level as "sparse", "marginal", "standard", or "strong" against the Linacre sample-size bands.

Usage

```
facet_small_sample_review(
  fit,
  diagnostics = NULL,
  thresholds = c(sparse = 10, marginal = 30, standard = 50)
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output. When supplied, per-level Infit, Outfit, and ModelSE are added to the report.

`thresholds` Named numeric vector of count bands. Defaults are `c(sparse = 10, marginal = 30, standard = 50)`. These are adapted from Linacre (1994): the 30-level band preserves Linacre's approximately `+-1.0 logit at 95% CI` line, while the `sparse < 10` floor and the `standard = 50` watermark are mfrmr-specific screening choices below Linacre's 30-examinee minimum and between Linacre's 30 and 100 thresholds.

Details

In mfrmr every facet is a fixed effect (see `?fit_mfrm`, "Fixed effects assumption"), so a level with very few ratings contributes an estimate with wide SE but no shrinkage toward the facet mean. This helper surfaces those levels up front so users can decide whether to drop them, pool them, or move to a hierarchical model outside mfrmr.

Value

A list of class `mfrm_facet_sample_review` with:

- `table`: one row per `(Facet, Level)` with `N`, `Estimate`, `SE`, `Infit`, `Outfit`, and `SampleCategory`.
- `summary`: counts of levels in each sample-size category, by facet.
- `facet_summary`: smallest observed level count per facet.
- `thresholds`: the applied count bands.

Interpreting output

- `"sparse"` (n < 10): level-level estimate is unstable; SE will be wide; consider combining with adjacent levels or treating as exploratory only.
- `"marginal"` (10 <= n < 30): below Linacre (1994) 95% CI +-1.0 logit threshold; usable as screening only.
- `"standard"` (30 <= n < 50): meets baseline stability; reasonable for publication if fit statistics are acceptable.
- `"strong"` (n >= 50): well-targeted; facet estimate is robust.

Because mfrmr has no shrinkage by default, sparse and marginal levels do not "borrow strength" from other levels. Jones and Wind (2018) report that rater estimates are particularly sensitive to thin linking; the `Facet = "Person"` row is usually less of a concern because the person prior integrates out the uncertainty.

Typical workflow

1. Fit with `fit_mfrm()`; optionally also produce `diagnostics` with `diagnose_mfrm()` if you want per-level Infit/Outfit.
2. Call `facet_small_sample_review(fit, diagnostics)`.
3. Read the `facet_summary` first: it highlights the worst level per facet. The `summary` table gives counts in each band.
4. If any facet is flagged as sparse or marginal, discuss it in the Methods section; `build_apa_outputs()` already adds a sentence about the band when `fit$summary$FacetSampleSizeFlag` is set.

References

Linacre, J. M. (2026). *A User's Guide to FACETS, Version 4.5.0*. Winsteps.com. https://www.winsteps.com/facets.htm

Linacre, J. M. (1994). Sample size and item calibration stability. *Rasch Measurement Transactions, 7*(4), 328. https://www.rasch.org/rmt/rmt74m.htm

Jones, E., & Wind, S. A. (2018). Using repeated ratings to improve measurement precision in incomplete rating designs. *Journal of Applied Measurement, 19*(2), 148-161.

See Also

detect_facet_nesting(), analyze_hierarchical_structure(), compute_facet_icc(), compute_facet_design_eff
reporting_checklist().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
review <- facet_small_sample_review(fit)
summary(review)

# Custom thresholds (e.g. a stricter protocol).
strict <- facet_small_sample_review(
  fit,
  thresholds = c(sparse = 15, marginal = 40, standard = 100)
)
strict$facet_summary
```

---

facet_statistics_report

Build a facet statistics report (preferred alias)

Description

Build a facet statistics report (preferred alias)

Usage

```
facet_statistics_report(
  fit,
  diagnostics = NULL,
  metrics = c("Estimate", "Infit", "Outfit", "SE"),
  ruler_width = 41,
  distribution_basis = c("both", "sample", "population"),
  se_mode = c("both", "model", "fit_adjusted")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
metrics	Numeric columns in diagnostics$measures to summarize.
ruler_width	Width of the fixed-width ruler used for M/S/Q/X marks.
distribution_basis	Which distribution basis to keep in the appended precision summary: "both" (default), "sample", or "population".
se_mode	Which standard-error mode to keep in the appended precision summary: "both" (default), "model", or "fit_adjusted".

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_facet_statistics (type = "means", "sds", "ranges").

Value

A named list with facet-statistics components. Class: mfrm_facet_statistics.

Interpreting output

- facet-level means/SD/ranges of selected metrics (Estimate, fit indices, SE).
- fixed-width ruler rows (M/S/Q/X) for compact profile scanning.

Typical workflow

1. Run facet_statistics_report(fit).
2. Inspect summary/ranges for anomalous facets.
3. Cross-check flagged facets with fit and chi-square diagnostics. The returned bundle now includes:

- precision_summary: facet precision/separation indices by DistributionBasis and SEMode
- variability_tests: fixed/random variability tests by facet
- se_modes: compact list of available SE modes by facet

See Also

diagnose_mfrm(), summary.mfrm_fit(), plot_facets_chisq(), mfrmr_reports_and_tables

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- facet_statistics_report(fit)
summary(out)
p_fs <- plot(out, draw = FALSE)
p_fs$data$plot
```

`fair_average_table` *Build an adjusted-score reference table bundle*

Description

Build an adjusted-score reference table bundle

Usage

```
fair_average_table(
  fit,
  diagnostics = NULL,
  facets = NULL,
  totalscore = TRUE,
  umean = 0,
  uscale = 1,
  udecimals = 2,
  reference = c("both", "mean", "zero"),
  label_style = c("both", "native", "legacy"),
  omit_unobserved = FALSE,
  xtreme = 0,
  fair_se = FALSE,
  ci_level = 0.95
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`facets`	Optional subset of facets.
`totalscore`	Include all observations for score totals (`TRUE`) or apply legacy extreme-row exclusion (`FALSE`).
`umean`	Additive score-to-report origin shift.
`uscale`	Multiplicative score-to-report scale.
`udecimals`	Rounding digits used in formatted output.
`reference`	Which adjusted-score reference to keep in formatted outputs: `"both"` (default), `"mean"`, or `"zero"`.
`label_style`	Column-label style for formatted outputs: `"both"` (default), `"native"`, or `"legacy"`.
`omit_unobserved`	If `TRUE`, remove unobserved levels.
`xtreme`	Extreme-score adjustment amount.

fair_se	Logical. When TRUE and fit is an MML bounded-GPCM fit, add structural delta-method standard errors and confidence limits for Fair(M) / AdjustedAverage and Fair(Z) / StandardizedAdjustedAverage. Person rows remain NA because MML person EAP estimates are not part of the structural Hessian. For RSM, PCM, and JML fits this option leaves fair-average SE columns unavailable.
ci_level	Confidence level used when fair_se = TRUE; default 0.95.

Details

This function wraps the package's adjusted-score calculations and returns both facet-wise and stacked tables. Historical display columns such as Fair(M) Average and Fair(Z) Average are retained for compatibility, and package-native aliases such as AdjustedAverage, StandardizedAdjustedAverage, ModelBasedSE, and FitAdjustedSE are appended to the formatted outputs.

For the Rasch-family RSM / PCM branch, these tables follow the standard FACETS Linacre construction: fair averages are Rasch-measure-to-score transformations evaluated in a standardized mean/zero-facet environment.

Bounded GPCM fits are supported under a slope-aware element-conditional construction. For each slope-facet element $j^\star$ the per-row fair-average is the GPCM expected score

$$\mathrm{FA}_{p,j^\star} = \sum_k k \cdot P_{GPCM}(X = k \mid \theta_p, a_{j^\star}, \boldsymbol{\delta}_{j^\star})$$

computed at that element's own discrimination $a_{j^\star}$ and threshold structure. Rows for non-slope facets (Person, Rater, ...) use the geometric-mean-one slope by the GPCM identification convention, so those rows remain continuous with the standard PCM Linacre fair-average and reduce to it exactly when all slopes equal one. This is an identification-based reporting convention for the package's bounded GPCM route, not a unique free-discrimination score-side analogue to FACETS fair averages. Do not report it as FACETS score-side equivalence or as an operational scoring rule unless that convention is substantively justified.

Standard errors on the fair-average value itself are opt-in for MML bounded GPCM fits via fair_se = TRUE. The original SE, Model S.E., ModelBasedSE, Real S.E., and FitAdjustedSE columns retain the same meaning as for PCM (scaled facet-measure SEs); fair-average uncertainty is reported under distinct columns such as Fair(M) S.E., Fair(M) CI Lower, and AdjustedAverageSE.

Value

A named list with:

- by_facet: named list of formatted data.frames
- stacked: one stacked data.frame across facets
- raw_by_facet: unformatted component tables
- settings: resolved options

Interpreting output

- stacked: cross-facet table for global comparison.
- by_facet: per-facet formatted tables for reporting.

- `raw_by_facet`: unformatted values for custom analyses/plots.
- `settings`: scoring-transformation and filtering options used.

Larger observed-vs-fair gaps can indicate systematic scoring tendencies by specific facet levels.

Typical workflow

1. Run `fair_average_table(fit, ...)`.
2. Inspect `summary(t12)` and `t12$stacked`.
3. Visualize with `plot_fair_average()`.

Output columns

The `stacked` data.frame contains:

Facet Facet name for this row.

Level Element label within the facet.

Obsvd Average Observed raw-score average.

Fair(M) Average Model-adjusted reference average on the reported score scale.

Fair(Z) Average Standardized adjusted reference average.

ObservedAverage, AdjustedAverage, StandardizedAdjustedAverage Package-native aliases for the three average columns above.

AdjustedAverageSE, AdjustedAverageCI_Lower, AdjustedAverageCI_Upper Optional structural delta-method uncertainty for `AdjustedAverage` when `fair_se = TRUE` and available.

StandardizedAdjustedAverageSE, StandardizedAdjustedAverageCI_Lower, StandardizedAdjustedAverageCI_Uppe Optional structural delta-method uncertainty for `StandardizedAdjustedAverage` when `fair_se = TRUE` and available.

Measure Estimated logit measure for this level.

SE Compatibility alias for the model-based standard error.

ModelBasedSE, FitAdjustedSE Package-native aliases for `Model S.E.` and `Real S.E.`.

Infit MnSq, Outfit MnSq Fit statistics for this level.

Standard-error caveat (read before quoting CIs)

The `SE`, `Model S.E.`, `ModelBasedSE`, `Real S.E.`, and `FitAdjustedSE` columns in this table are the **measure-level** standard errors of the underlying facet element (the same SE that would appear in `summary(fit)$facets`), rescaled by the fair-average score scale factor so the units line up with the reported `Fair(M) Average` / `Fair(Z) Average` columns. They are **not** delta-method standard errors of the fair-average values themselves. When `fair_se = TRUE`, the distinct `Fair(M) S.E.` / `Fair(Z) S.E.` columns are computed by propagating the joint covariance of the relevant facet element, the threshold parameters, and the slope parameters through the gradient of $\mathrm{E}[X \mid \theta_p, j^\star]$. This is a structural covariance calculation: MML person EAP estimates are conditioned on rather than included in the Hessian, so person rows receive unavailable fair-average SEs. **Do not use the measure-level `SE` / `Model S.E.` columns as $\pm 1.96 \cdot \mathrm{SE}$ confidence-interval bounds on the fair-average value.**

References

- Linacre, J. M. (1989). *Many-Facet Rasch Measurement*. MESA Press.
- Linacre, J. M. (1994). *Many-facet Rasch Measurement* (2nd ed.). MESA Press.
- Linacre, J. M. (2026). *A user's guide to FACETS, version 4.5.0*. Winsteps.com. https://www.winsteps.com/facets.htm (FACETS Table 12 corresponds to the fair-average construction implemented here for RSM / PCM fits; the slope-aware element-conditional construction for bounded GPCM is documented in this help page.)
- Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573. doi:10.1007/BF02293814
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika, 47*(2), 149-174. doi:10.1007/BF02296272
- Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*(2), 159-176. (Cited for the bounded GPCM slope-aware extension.)

See Also

diagnose_mfrm(), unexpected_response_table(), displacement_table()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
t12 <- fair_average_table(fit, udecimals = 2)
t12_native <- fair_average_table(fit, reference = "mean", label_style = "native")
summary(t12)
p_t12 <- plot(t12, draw = FALSE)
p_t12$data$plot
```

---

fit_measures_table *Build a FACETS-style fit-measures review table*

Description

Build a FACETS-style fit-measures review table

Usage

```
fit_measures_table(
  x,
  diagnostics = NULL,
  facet = NULL,
  include_person = FALSE,
  lower = NULL,
```

```
  upper = NULL,
  zstd_cut = 2,
  ci_level = 0.95,
  threshold_profiles = c("literature", "active", "all", "none"),
  fit_df_method = c("engine", "facets", "both"),
  df_zstd_tolerance = 0.05,
  df_zstd_large_shift = 0.5,
  df_ratio_tolerance = 0.05,
  sort_by = c("status", "abs_zstd", "facet", "level"),
  top_n = Inf
)
```

Arguments

x	Output from fit_mfrm() or diagnose_mfrm().
diagnostics	Optional diagnostics object. If supplied, x may be the fitted object used only for provenance.
facet	Optional facet-name filter, for example "Rater".
include_person	Logical; if FALSE (default), excludes the Person facet so operational facet elements are shown first.
lower, upper	Optional mean-square review band. Defaults to mfrm_misfit_thresholds().
zstd_cut	Absolute ZSTD cutoff used for directional underfit/overfit flags. Default 2.
ci_level	Confidence level used to add approximate Wald intervals for facet measures. Default 0.95.
threshold_profiles	Which mean-square threshold profiles to summarize in addition to the active table band. "literature" (default) returns commonly cited bands from Linacre, Bond & Fox, and Wright & Linacre; "active" returns only the active band; "all" returns both; "none" suppresses profile summaries.
fit_df_method	Degrees-of-freedom convention used when diagnostics is computed inside the helper. "engine" keeps the package-native fit df, "facets" makes primary ZSTD columns use the FACETS/Wright-Masters fourth-moment df convention, and "both" keeps engine columns primary while adding FACETS-style companion df/ZSTD columns for comparison.
df_zstd_tolerance	Smallest absolute engine-vs-FACETS-style ZSTD difference treated as interpretively visible rather than rounding noise in df_sensitivity. Default 0.05.
df_zstd_large_shift	Absolute engine-vs-FACETS-style ZSTD difference labeled large_zstd_shift when the zstd_cut flag status is unchanged. Default 0.5.
df_ratio_tolerance	Relative df-difference threshold used to label df_convention_difference; for example, 0.05 means a 5 percent engine-vs-FACETS-style df difference. Default 0.05.
sort_by	Sorting rule: "status" prioritizes underfit/overfit rows, "abs_zstd" sorts by largest absolute ZSTD, and "facet" / "level" sort alphabetically.
top_n	Optional maximum number of rows in the returned main table.

Details

This helper gives users a direct table route for the common FACETS-style question: which raters, criteria, or other facet elements show underfit or overfit? It uses the fit statistics already computed by `diagnose_mfrm()`.

Directional labels are based on both mean-square and ZSTD evidence: high MnSq or positive large ZSTD is labeled `underfit`; low MnSq or negative large ZSTD is labeled `overfit`. Rows with conflicting directions are labeled `mixed`. Treat the table as a review screen and inspect substantive context before removing raters or changing an instrument.

FACETS-style ZSTD comparison is controlled by `fit_df_method`. MnSq values should be compared first; df and ZSTD columns explain how the same MnSq values are standardized. Use `fit_df_method = "both"` when preparing a table for FACETS users or when explaining why |ZSTD| flags change across df conventions. The `df_zstd_tolerance`, `df_zstd_large_shift`, and `df_ratio_tolerance` arguments make the df-sensitivity screen explicit so the same table can be reproduced under stricter or more permissive review rules.

Value

A bundle of class `mfrm_fit_measures` with:

- `table`: R-friendly fit-measure table with status columns
- `facets_table`: FACETS-style column labels for reporting/review
- `status_summary`: counts by facet and fit status
- `profile_summary_by_facet`: underfit/overfit rates for each threshold profile and facet
- `profile_summary_overall`: threshold-profile rates pooled over facets
- `df_sensitivity`: row-level engine-vs-FACETS-style df/ZSTD comparison
- `df_sensitive`: subset of rows where df convention changes the ZSTD flag or materially changes ZSTD interpretation
- `df_sensitivity_summary`: counts of df-sensitive rows
- `underfit`, `overfit`, `mixed`: filtered row subsets
- `df_conversion_guide`: FACETS-style df/ZSTD comparison guide
- `settings`: thresholds and filters used

See Also

`diagnose_mfrm()`, `facets_fit_review()`, `plot_bubble()`, `mfrm_misfit_thresholds()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
fm <- fit_measures_table(fit, facet = "Rater")
fm$facets_table
fm$underfit

# Include FACETS-style df/ZSTD companion columns for comparison.
fm_facets <- fit_measures_table(fit, facet = "Rater", fit_df_method = "both")
```

```
fm_facets$df_conversion_guide$decision_guide
```

`fit_mfrm` *Fit many-facet ordered-response models with a flexible number of facets*

Description

This is the package entry point. It wraps `mfrm_estimate()` and defaults to `method = "MML"`. Any number of facet columns can be supplied via `facets`. The `RSM` / `PCM` branches are the package's many-facet Rasch-family reference route; the bounded `GPCM` branch is available where explicitly documented.

Usage

```
fit_mfrm(
  data,
  person,
  facets,
  score,
  rating_min = NULL,
  rating_max = NULL,
  weight = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  model = c("RSM", "PCM", "GPCM"),
  method = c("MML", "JML", "JMLE"),
  step_facet = NULL,
  slope_facet = NULL,
  facet_interactions = NULL,
  min_obs_per_interaction = 10,
  interaction_policy = c("warn", "error", "silent"),
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  positive_facets = NULL,
  anchor_policy = c("warn", "error", "silent"),
  min_common_anchors = 5L,
  min_obs_per_element = 30,
  min_obs_per_category = 10,
  quad_points = 31,
  maxit = 400,
  reltol = 1e-06,
  mml_engine = c("direct", "em", "hybrid"),
  population_formula = NULL,
```

```
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
  facet_shrinkage = c("none", "empirical_bayes", "laplace"),
  facet_prior_sd = NULL,
  shrink_person = FALSE,
  attach_diagnostics = FALSE,
  checkpoint = NULL
)
```

Arguments

data	A data.frame in long format with one row per observed rating event.
person	Column name for the person (character scalar).
facets	Character vector of facet column names.
score	Column name for the observed ordered category score. Values must be coercible to numeric integer category codes. Fractional values are rejected. Binary `0/1` or `1/2` responses are supported as the ordered two-category special case. When `keep_original = FALSE`, unused intermediate categories are collapsed to a contiguous internal scale and the mapping is recorded in `fit$prep$score_map`. If `rating_min` / `rating_max` are supplied and the observed scores are a contiguous subset of that range (for example a 1-5 scale with only 2-5 observed), the supplied full range is retained so zero-count boundary categories remain part of the fitted score support.
rating_min	Optional minimum category value. Supply this with `rating_max` when the intended score scale includes unobserved boundary categories.
rating_max	Optional maximum category value. Supply this with `rating_min` when the intended score scale includes unobserved boundary categories.
weight	Optional weight column name.
keep_original	Logical. `FALSE` (the current default) collapses non-consecutive observed categories to a contiguous internal scale and records the mapping in `fit$prep$score_map` (the downstream Count = 0 rows are consequently absent). `TRUE` preserves the declared scale so unused intermediate categories remain visible in `rating_scale_table()` and APA outputs, which is recommended for publication reporting.
missing_codes	Optional pre-processing step that converts sentinel missing-code values to `NA` across the `person`, `facets`, and `score` columns before any downstream logic. One of: • `NULL` (default): no recoding; strictly backward-compatible. • `TRUE` or `"default"`: FACETS / SPSS / SAS convention set (`"99"`, `"999"`, `"-1"`, `"N"`, `"NA"`, `"n/a"`, `"."`, `""`). • Character vector: an explicit code set, e.g. `c("99", "999", ".a")`. Replacement counts are recorded in `fit$prep$missing_recoding` and surfaced by `build_mfrm_manifest()`. Equivalent to calling `recode_missing_codes()` manually before the fit.
model	`"RSM"`, `"PCM"`, or bounded `"GPCM"`.

`method` "MML" (default) or "JML". "JMLE" is accepted as a backward-compatible alias for the same joint-maximum-likelihood path.

`step_facet` Step facet for PCM and the bounded GPCM branch. For GPCM, this should be supplied explicitly rather than relying on an implicit default.

`slope_facet` Slope facet for the bounded GPCM branch. The current release requires `slope_facet == step_facet` and uses a positive-slope identification convention on the log scale with geometric mean discrimination fixed to 1.

`facet_interactions`
Optional confirmatory two-way interaction terms between non-person facets, supplied as explicit character terms such as "Rater:Criterion" or as a list of length-two character vectors. These interactions are estimated simultaneously as fixed effects in RSM and PCM fits. Person-involving interactions, higher-order interactions, and random-effect interaction terms are outside the current scope.

`min_obs_per_interaction`
Minimum weighted observations recommended for each interaction cell. Cells below this value are flagged in `interaction_effect_table()` and handled according to `interaction_policy`.

`interaction_policy`
How to handle sparse interaction cells: "warn" (default), "error", or "silent".

`anchors` Optional anchor table.

`group_anchors` Optional group-anchor table.

`noncenter_facet`
One facet to leave non-centered.

`dummy_facets` Facets to fix at zero.

`positive_facets`
Facets with positive orientation.

`anchor_policy` How to handle anchor-review issues: "warn" (default), "error", or "silent".

`min_common_anchors`
Minimum anchored levels per linking facet used in anchor-review recommendations.

`min_obs_per_element`
Minimum weighted observations per facet level used in anchor-review recommendations.

`min_obs_per_category`
Minimum weighted observations per score category used in anchor-review recommendations.

`quad_points` Integer number of Gauss-Hermite quadrature points used for MML integration over the person distribution. Default is 31, chosen so that marginal log-likelihood values are stable enough for direct manuscript reporting. Recommended tiers:

7 fast exploratory scan; in-package helpers such as `predict_mfrm_population()` and `reference_case_benchmark()` u
15 intermediate analysis when runtime matters.
31 default / publication tier.
61+ ultra-precise runs when benchmarking or working on very narrow score supports.

Internal benchmarks show the marginal log-likelihood still drifts by ~0.5-1 logit between quad_points = 15 and quad_points = 61 on moderately sized designs, which is why the default now sits at the publication tier; set a lower value explicitly for exploratory runs.

maxit
: Maximum optimizer iterations.

reltol
: Optimization tolerance.

mml_engine
: MML optimization engine for method = "MML": "direct" (default) uses direct BFGS on the marginal log-likelihood, "em" uses an EM loop for RSM / PCM with population = NULL, and "hybrid" uses EM as a warm start before the direct optimizer. Unsupported combinations currently fall back to "direct" and record that fallback in fit$summary.

population_formula
: Optional one-sided formula for a person-level latent-regression population model, for example ~ grade + ses. In the current release, latent regression is implemented only for method = "MML" with a unidimensional conditional-normal population model.

person_data
: Optional one-row-per-person data.frame holding background variables for population_formula. Numeric, logical, factor, ordered factor, and character predictors are expanded through stats::model.matrix(); categorical xlevels and contrasts are stored for replay and scoring. Required when population_formula is supplied.

person_id
: Optional person-ID column in person_data. Defaults to person when that column exists in person_data.

population_policy
: How missing background data are handled for a latent-regression fit. "error" (default) requires complete person-level covariates; "omit" fits the model on the complete-case subset and records omitted persons / omitted response rows in the returned population metadata while retaining the observed-person-aligned pre-omit table for replay/export provenance.

facet_shrinkage
: Character. "none" (default) keeps the 0.1.5 fixed-effects behaviour. "empirical_bayes" applies a post-hoc James-Stein / empirical-Bayes shrinkage to each non-person facet (Efron & Morris, 1973); fit$facets$others gains ShrunkEstimate, ShrunkSE, and ShrinkageFactor columns, and fit$shrinkage_report records the per-facet prior variance and effective degrees of freedom. "laplace" currently aliases to "empirical_bayes" and is reserved for a future penalised-MML implementation.

facet_prior_sd
: Optional numeric scalar. When supplied, the shrinkage prior variance is fixed at facet_prior_sd^2 instead of being estimated by method of moments. Useful for eliciting a prior from domain knowledge or a previous fit.

shrink_person
: Logical. When TRUE and facet_shrinkage is active, the same empirical-Bayes shrinkage is applied to fit$facets$person. Default FALSE, since MML already integrates over an N(0, 1) prior on theta; the option mainly benefits JML.

attach_diagnostics
: Logical. When TRUE, diagnose_mfrm() is run once after the fit with residual_pca = "none", and the per-level SE, Infit, Outfit, InfitZSTD, OutfitZSTD, and PtMeaCorr columns from diagnostics$measures are merged onto fit$facets$others

(non-person facets) and fit$facets$person (Person rows). This is convenient when downstream code expects a FACETS Table 7 style facet table with fit statistics in one place, and lets summary(fit) show per-person fit columns alongside the measure. For person rows, an existing posterior SE (typical for method = "MML") is preserved and the diagnostic SE is only attached when the existing column is empty. Adds diagnostic runtime (typically +1-2 s on moderate designs) and sets fit$config$attached_diagnostics = TRUE. Default FALSE preserves the minimal Facet / Level / Estimate layout.

checkpoint — Optional list(file = ..., every_iter = ...). When supplied, the MML EM engine writes its state to file every every_iter outer EM iterations using saveRDS(). If the file already exists when the fit starts, the engine resumes from the recorded iteration. Only the EM engine (mml_engine = "em" or the EM warm-start step of mml_engine = "hybrid") honours the checkpoint; the direct optim() engine ignores it. Use this to make long MML EM fits crash-resilient on shared compute environments.

Details

Data must be in **long format** (one row per observed rating event).

Value

An object of class mfrm_fit (named list) with:

- summary: one-row model summary (LogLik, AIC, BIC, convergence) including public Method, internal MethodUsed, and MMLEngineRequested, MMLEngineUsed, and EMIterations for MML fits
- facets$person: person estimates (Estimate; plus SD for MML)
- facets$others: facet-level estimates for each facet
- steps: estimated threshold/step parameters as a one-row-per-step tibble with Estimate. Bare fits keep this table as point estimates. diagnose_mfrm() exposes MML observed-information step uncertainty in diagnostics$parameter_uncertainty$steps; when attach_diagnostics = TRUE, those SE, confidence-limit, and status columns are attached to fit$steps when the Hessian is available. For step-structure quality, also use the step-collapse and disordering warnings from diagnose_mfrm() and category_structure_report().
- slopes: estimated discrimination parameters for GPCM fits as a one-row-per-slope-element tibble with LogEstimate and Estimate. Bare fits keep this table as point estimates. For MML bounded-GPCM fits, diagnose_mfrm() exposes log-slope SEs plus positive-scale delta-method SEs and confidence limits in diagnostics$parameter_uncertainty$slopes; when attach_diagnostics = TRUE, those columns are attached to fit$slopes when the Hessian is available. The identification convention pins the geometric mean of slopes at 1.
- interactions: model-estimated facet interaction effects and metadata when facet_interactions is supplied
- population: population-model metadata. Ordinary fits keep an inactive scaffold (active = FALSE, posterior_basis = "legacy_mml"). Active latent-regression fits store the fitted design matrix, regression coefficients, residual variance, omission review, the complete-case estimation table (person_table), and the observed-person-aligned replay/export provenance table retained before complete-case omission (person_table_replay), plus stored categorical

xlevels / contrasts for model-matrix replay and scoring, together with posterior_basis = "population_model".

- config: resolved model configuration used for estimation, including config$anchor_review
- prep: preprocessed data/level metadata
- opt: raw optimizer result from stats::optim()

Model

fit_mfrm() estimates many-facet ordered-response models. The RSM and PCM branches follow the many-facet Rasch-family tradition (Linacre, 1989); the bounded GPCM branch extends the partial-credit kernel with estimated positive slopes under the package's documented identification constraints. For the equal-slope RSM/PCM branch, a two-facet design (rater j, criterion i) is:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k-1)} = \theta_n - \delta_j - \beta_i - \tau_k$$

where θ_n is person ability, δ_j rater severity, β_i criterion difficulty, and τ_k the k-th Rasch-Andrich threshold. Any number of facets may be specified via the facets argument; each enters as an additive term in the linear predictor η.

With model = "RSM", thresholds τ_k are shared across all levels of all facets. With model = "PCM", each level of step_facet receives its own threshold vector $\tau_{i,k}$ on the package's shared observed score scale.

With bounded model = "GPCM", the adjacent-category kernel is multiplied by a positive slope for the designated slope-facet level:

$$\ln \frac{P(X_{nij} = k)}{P(X_{nij} = k-1)} = \alpha_g(\eta - \tau_{g,k}), \quad \alpha_g > 0.$$

The current implementation requires slope_facet == step_facet and identifies slopes by a sum-to-zero constraint on log slopes, so their geometric mean is 1.

With only two ordered categories ($K = 1$), the RSM/PCM branch reduces to the usual binary Rasch logit for the single category boundary:

$$\ln \frac{P(X_{n\cdot} = 1)}{P(X_{n\cdot} = 0)} = \eta - \tau_1$$

Bounded GPCM uses the slope-scaled counterpart $\alpha_g(\eta - \tau_{g,1})$.

With method = "MML", person parameters are integrated out using Gauss-Hermite quadrature and EAP estimates are computed post-hoc. With method = "JML", all parameters are estimated jointly as fixed effects. "JMLE" remains an accepted compatibility alias, but package output now uses "JML" as the public label. See the "Estimation methods" section of mfrmr-package for details.

Weighting policy

`mfrmr` treats `RSM` / `PCM` as the equal-weighting reference route for operational many-facet measurement. In that Rasch-family branch, discrimination is fixed, so the scoring model does not differentially reweight item-facet combinations through estimated slopes.

Bounded `GPCM` is supported as an alternative when users explicitly accept discrimination-based reweighting. This often improves model fit, but the package does not treat better fit alone as a sufficient reason to replace an equal-weighting Rasch-family model.

The `weight` argument is separate from that modeling choice. It supplies an observation-weight column; it does not create a free-form facet-weighting scheme and does not change the fixed-discrimination contract of `RSM` / `PCM`.

Input requirements

Minimum required columns are:

- person identifier (`person`)
- one or more facet identifiers (`facets`)
- observed score (`score`)

Scores are treated as ordered categories. Non-numeric score labels are dropped with a warning after coercion, whereas fractional numeric scores are rejected with an error instead of being silently truncated.

The fitted many-facet ordered-response model assumes conditional independence of observations given the person and facet parameters (Linacre, 1989). Repeated ratings of the same person-criterion combination by the same rater violate this assumption. When such structures may be present, follow fitting with `diagnose_mfrm(fit, diagnostic_mode = "both")`; its `strict_pairwise_local_dependence` screen is an exploratory check for residual dependence beyond what the additive linear predictor absorbs.

Binary responses are therefore supported as ordered two-category scores (for example `0/1` or `1/2`) under the same ordered-response interface. If your observed categories do not start at 0, set `rating_min`/`rating_max` explicitly to avoid unintended recoding assumptions. For example, if the intended instrument is a 1-5 scale but the current sample only uses 2-5, set `rating_min = 1, rating_max = 5` to retain the zero-count category 1 in the score support. If these bounds are omitted, the observed score range is used and the provenance is stored in `fit$prep` and `summary(fit)$settings_overview`. Set `options(mfrmr.show_inferred_rating_range = TRUE)` when you want an interactive reminder whenever a bound is inferred. Data-preparation events such as row drops, ID trimming, duplicate person-by-facet cells, and single-level facets are stored in `fit$prep$row_retention` and `fit$prep$preparation_notes`. Routine row-drop/trim/single-level messages are quiet by default; set `options(mfrmr.show_preparation_messages = TRUE)` to show them during interactive checks.

When `keep_original = FALSE`, observed gaps such as `1, 3, 5` are recoded internally to a contiguous scale (`1, 2, 3`) and the mapping is stored in `fit$prep$score_map`. To retain zero-count intermediate categories as part of the original scale, set `keep_original = TRUE` in addition to supplying the full `rating_min` / `rating_max` range.

Fixed effects assumption (facets have no prior)

`fit_mfrm()` follows the Linacre (1989) many-facet Rasch specification: person ability is integrated out under a `N(0, 1)` prior (or under the `N(X\beta, \sigma^2)` latent-regression population model when `population_formula` is supplied), but every facet parameter (`Rater`, `Criterion`, `Task`, ...) is estimated as a fixed effect identified by a sum-to-zero constraint. There is no hierarchical prior, no shrinkage, and no variance component for the facets.

Practical implication: when a facet has very few observed levels (for example 3 raters) or some of its levels have very few ratings (for example 5 ratings per rater), the fixed-effect estimates retain wide SEs, and extreme estimates are not pulled toward the facet mean. Jones and Wind (2018) note that rater estimates in particular are "more sensitive to link reductions" than examinee or task estimates. For a publication-workflow review of this, use:

- `facet_small_sample_review()` for per-level N and SE bands against Linacre (1994) sample-size guidelines.
- `detect_facet_nesting()` and `analyze_hierarchical_structure()` when raters are nested in regions, schools, or other strata that the additive fixed-effects MFRM cannot partition out.
- `compute_facet_icc()` and `compute_facet_design_effect()` for descriptive variance-component summaries based on `lme4` (optional).

`fit$summary$FacetSampleSizeFlag` summarizes the worst Linacre band across non-person facet levels (`"sparse"` < 10, `"marginal"` < 30, `"standard"` < 50, `"strong"` >= 50).

JML estimator caveat (use MML for final reporting)

Joint maximum likelihood (`method = "JML" / "JMLE"`) estimates both the structural parameters (facets, thresholds, slopes) and every person measure as fixed parameters in one optimization. This is the **incidental-parameter problem** of Neyman & Scott (1948): the structural parameter estimates are inconsistent as the number of persons grows with the number of items per person held fixed, carrying a bias of order $1/L$ (where L is the number of items per person) that does not vanish with sample size. Wright & Stone (1979) and Wright & Masters (1982, ch. 5) document an empirical $(L-1)/L$ correction that approximately removes the bias for the dichotomous Rasch model; mfrmr does **not** apply that correction (no `bias_correction` argument exists). The JML branch also does not produce a profile-likelihood Hessian for the structural parameters: SEs reported under JML are observation-table approximations ($1/\sqrt{\sum \mathrm{Var}(X_{pi})}$) and are marked as exploratory in the diagnostics output.

Practical recommendation:

- Use `method = "MML"` for any value reported in a manuscript or operational decision. MML integrates the person measures out under a population prior and produces consistent structural estimates with marginal observed-information SEs.
- Use `method = "JML"` only for fast exploratory iteration, the classical FACETS-style workflow, or contexts where the bias is tolerable (large L per person, descriptive screening, or teaching).
- When a third-party CML estimator is needed (the only consistent Rasch-family estimator under the incidental-parameter setting), fit with `eRm` and import via `import_erm_fit()`.

Model-estimated facet interactions

`facet_interactions` adds confirmatory fixed-effect interaction terms to the linear predictor. For example, `facet_interactions = "Rater:Criterion"` estimates a rater-by-criterion deviation matrix in the same likelihood as the main MFRM fit. The additive reference is

$$\eta_{nij} = \theta_n - \delta_j - \beta_i$$

and the interaction extension is

$$\eta_{nij} = \theta_n - \delta_j - \beta_i + \gamma_{ji}$$

where the interaction block is identified by zero marginal sums:

$$\sum_j \gamma_{ji} = 0, \quad \sum_i \gamma_{ji} = 0.$$

With J levels of the first facet and I levels of the second facet, this contributes $(J-1)(I-1)$ free parameters. Positive interaction estimates indicate scores higher than expected under the additive main-effects model for that facet-level combination; negative estimates indicate lower-than-expected scores.

This is a model-estimated interaction term, not the residual screening reported by `estimate_bias()` or `estimate_all_bias()`. In line with the MFRM bias-interaction literature, the facet pair should be named explicitly before fitting. Exploratory use is possible, but should be reported as screening, with sparse-cell and multiplicity caveats. The current implementation is intentionally narrow: two-way non-person facet interactions for `RSM` and `PCM` only, estimated as fixed effects. GPCM interactions, person interactions, higher-order interactions, and random-effect facet interactions are deferred.

This is ordered binary support, not a separate nominal-response model. In `PCM`, a binary fit still uses one threshold per `step_facet` level on the shared observed-score scale.

Supported model/estimation combinations in the current release:

- `model = "RSM"` with `method = "MML"` or `"JML"/"JMLE"`
- `model = "PCM"` with a designated `step_facet` (defaults to first facet)
- `facet_interactions` with `model = "RSM"` or `"PCM"` for explicit two-way non-person facet interactions
- `model = "GPCM"` is currently implemented only for the narrow bounded branch with `slope_facet == step_facet`; `MML` and `JML` fitting, core summaries, fixed-calibration posterior scoring, `compute_information()`, Wright/pathway/CCC fit plots, `diagnose_mfrm()`, residual-PCA follow-up, `interrater_agreement_table()`, `unexpected_response_table()`, `displacement_table()`, `measurable_summary_table()`, `rating_scale_table()`, `facet_quality_dashboard()`, `reporting_checklist()` `category_structure_report()`, `category_curves_report()`, and graph/scorefile `facets_output_file_bundle()` routes are available with score-side caveats. Direct simulation specifications and data generation are also supported through `build_mfrm_sim_spec()`, `extract_mfrm_sim_spec()`, and `simulate_mfrm_data()` when the slope-aware generator contract is stored explicitly; direct recovery checks are available through `evaluate_mfrm_recovery()` and `assess_mfrm_recovery()`.

Slope-aware fair_average_table() and estimate_bias() are available with their documented caveats. Role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers are available as caveated sensitivity evidence. Full FACETS-style score-side contract review, posterior predictive checks, and heavy backend routes should be treated as unsupported unless documented otherwise. Use gpcm_capability_matrix() as the formal boundary statement for the current GPCM scope.

Latent-regression status:

- population_formula = NULL keeps the legacy unconditional MML / JML behavior.
- Supplying population_formula activates a first-version latent-regression branch for method = "MML" only.
- The current branch assumes a one-dimensional conditional-normal population model with person-specific quadrature nodes $\theta_{nq} = x_n^\top \beta + \sigma z_q$.
- Background variables must be supplied in person_data; numeric/logical columns and categorical factor/character columns are expanded through stats::model.matrix().
- Current overlap with the ConQuest latent-regression documentation is limited to direct estimation from response data under a unidimensional MML population model with package-built model-matrix covariates. It should not be described as numerical equivalence for arbitrary imported design matrices, multidimensional models, or the full ConQuest plausible-values workflow.
- predict_mfrm_units() and sample_mfrm_plausible_values() can score latent-regression fits under the fitted population model, but they require one-row-per-person background data for scored units when the fitted population model includes covariates. Intercept-only latent-regression fits (population_formula = ~ 1) can reconstruct that minimal person table internally during scoring.

Latent-regression workflow

For an initial latent-regression run, keep the setup explicit:

1. Put response data in data, with one row per rating event.
2. Put background variables in person_data, with exactly one row per person. The ID column must match person, or be supplied through person_id.
3. Use method = "MML" and a one-sided formula such as population_formula = ~ Grade + Group.
4. Numeric/logical and factor/character predictors are expanded with stats::model.matrix(). After fitting, inspect summary(fit)$population_coding to see the fitted levels, contrasts, and encoded design columns that will be reused for scoring/replay.
5. Start with population_policy = "error" while preparing data. Use "omit" only when complete-case removal is intended, and then inspect summary(fit)$population_overview and summary(fit)$caveats before reporting results.
6. Report summary(fit)$population_coefficients as coefficients of the conditional-normal latent population model, not as a post hoc regression on EAP or MLE scores.

Latent-regression standard-error caveat

`summary(fit)$population_coefficients` reports point estimates of $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ only. mfrmr does **not** currently compute standard errors, confidence intervals, or asymptotic z / Wald statistics for the population-model parameters: no Hessian on $(\boldsymbol{\beta}, \log\sigma^2)$ is extracted from the marginal log-likelihood, and no `vcov()` method is exposed for these coefficients. Treat the coefficient table as point estimates suitable for descriptive reporting; **do not** quote $\hat{\beta}_j \pm 1.96 \cdot \mathrm{SE}$ bounds because the SE column is not provided. A marginal-Hessian-based SE for $(\boldsymbol{\beta}, \sigma^2)$ is planned for a future release.

Identification: the latent-regression intercept is identifiable only under the default `noncenter_facet = "Person"` (which sum-to-zero- centers all non-Person facets). If you re-anchor identification on a non-Person facet, the intercept becomes confounded with the freed Person-facet mean and the coefficient table becomes unidentified; mfrmr does not currently warn about this failure mode in the design-matrix check.

Anchor inputs are optional:

- `anchors` should contain facet/level/fixed-value information.
- `group_anchors` should contain facet/level/group/group-value information. Both are normalized internally, so column names can be flexible (`facet`, `level`, `anchor`, `group`, `groupvalue`, etc.).

Anchor review behavior:

- `fit_mfrm()` runs an internal anchor review.
- invalid rows are removed before estimation.
- duplicate rows keep the last occurrence for each key.
- `anchor_policy` controls whether detected issues are warned, treated as errors, or kept silent.

Facet sign orientation:

- facets listed in `positive_facets` are treated as `+1`
- all other facets are treated as `-1` This affects interpretation of reported facet measures.

Performance tips

For exploratory work, `method = "JML"` is usually faster than `method = "MML"`, but it may require a larger `maxit` to converge on larger datasets.

For MML runs, `quad_points` is the main accuracy/speed trade-off. The `@param quad_points` tier table is the authoritative reference; in short:

- `quad_points = 7` is a lightweight setting for quick iteration.
- `quad_points = 15` is an intermediate option when runtime matters.
- `quad_points = 31` is the package default and the publication tier: the marginal log-likelihood is stable enough for direct manuscript reporting.
- `quad_points = 61` (or higher) is reserved for ultra-precise benchmarking on very narrow score supports.
- `mml_engine = "direct"` remains the most stable general-purpose path.
- `mml_engine = "em"` or `"hybrid"` currently target `RSM` / `PCM` fits without a latent-regression population model.

- Benchmark your own workload before using `mml_engine = "em"` or `"hybrid"` for final reporting; `direct` remains the safer default when you have not compared engines for your data.
- For RSM and PCM fits only, an opt-in C++ MML backend can be enabled with `options(mfrmr.use_cpp11_backend = TRUE)`. The backend implements the same physicist Gauss-Hermite quadrature and sum-to-zero identification as the pure-R engine, validated against the pure-R reference at `tolerance = 1e-12` on a fixed regression fixture. It is opt-in for this release; the default flip to ON is planned for a follow-up release after a cycle of community testing. GPCM fits stay on the pure-R engine regardless of the option.

Downstream diagnostics can also be staged:

- use `diagnose_mfrm(fit, residual_pca = "none")` for a quick first pass
- add residual PCA only when you need exploratory residual-structure evidence

Downstream diagnostics report `ModelSE` / `RealSE` columns and related reliability indices. For `MML`, non-person facet `ModelSE` values are based on the observed information of the marginal log-likelihood and person rows use posterior SDs from EAP scoring. For `JML`, these quantities remain exploratory approximations and should not be treated as equally formal.

For bounded `GPCM`, residual-based mean-square fit screens are also best treated as exploratory diagnostics rather than strict Rasch-style invariance tests, because the discrimination parameter is free.

Interpreting output

A typical first-pass read is:

1. `fit$summary` for convergence and global fit indicators.
2. `summary(fit)` for human-readable overviews.
3. for `RSM` / `PCM`, `diagnose_mfrm(fit)` for element-level fit, approximate separation/reliability, and warning tables.
4. for bounded `GPCM`, use `diagnose_mfrm()` and the residual-based table helpers as exploratory screens, together with posterior scoring / `compute_information()` where documented.

Typical workflow

1. Fit the model with `fit_mfrm(...)`.
2. Validate convergence and scale structure with `summary(fit)`.
3. For `RSM` / `PCM`, run `diagnose_mfrm()` and proceed to reporting with `build_apa_outputs()`.
4. For bounded `GPCM`, use the fitted object, slope summary, `diagnose_mfrm()`, residual-based table helpers, posterior scoring helpers, `compute_information()`, direct simulation/recovery helpers, `fair_average_table()`, and `estimate_bias()` with their documented caveats. Use `gpcm_capability_matrix()` to confirm which helper families are currently supported, caveated, blocked, or deferred.

References

The ordered-category many-facet formulation follows Linacre (1989), with the RSM and PCM branches grounded in Andrich (1978) and Masters (1982). The bounded GPCM branch follows the generalized partial credit formulation of Muraki (1992) under a package-specific positive log-slope identification convention. The MML route follows the quadrature-based marginal-likelihood framework of Bock and Aitkin (1981).

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573.
- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Linacre, J. M. (1989). *Many-facet Rasch measurement*. MESA Press.
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174.
- Myford, C. M., & Wolfe, E. W. (2003). Detecting and measuring rater effects using many-facet Rasch measurement: Part I. *Journal of Applied Measurement*, 4(4), 386-422.
- Myford, C. M., & Wolfe, E. W. (2004). Detecting and measuring rater effects using many-facet Rasch measurement: Part II. *Journal of Applied Measurement*, 5(2), 189-227.
- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm*. Applied Psychological Measurement, 16(2), 159-176.
- Robitzsch, A., & Steinfeld, J. (2018). *Item response models for human ratings: Overview, estimation methods, and implementation in R*. Psychological Test and Assessment Modeling, 60(1), 101-139.

See Also

diagnose_mfrm(), estimate_bias(), build_apa_outputs(), gpcm_capability_matrix, mfrmr_workflow_methods, mfrmr_reporting_and_apa

Examples

```
# Fast smoke run: a JML fit on the bundled `example_core` toy
# dataset finishes in well under a second and returns a populated
# `summary` overview ready for inspection.
toy <- load_mfrmr_data("example_core")
fit_quick <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                      method = "JML", maxit = 30)
fit_quick$summary[, c("Model", "Method", "N", "Converged")]


# Full run with the package default MML estimator (recommended for
# final reporting because person parameters are integrated out under
# an N(0, 1) prior). The default `quad_points = 31` is the
# publication tier; `quad_points = 7` below is an exploratory speed
# setting and should not be used as the final manuscript fit.
fit <- fit_mfrm(
  data = toy,
  person = "Person",
```

```
  facets = c("Rater", "Criterion"),
  score = "Score",
  model = "RSM",
  quad_points = 7,
  maxit = 30
)
fit$summary
s_fit <- summary(fit)
s_fit$overview[, c("Model", "Method", "Converged")]
# Look for: Converged = TRUE. If FALSE, raise `maxit`, relax `reltol`,
#   or inspect `summary(fit)$key_warnings` for sparse-cell or
#   identification flags.
s_fit$person_overview
# Look for: Mean ~ 0 logits and SD ~ 1 logit are typical when the
#   sample is centred on the test difficulty. SD < 0.5 suggests the
#   test is too easy / hard for this group; SD > 1.5 suggests strong
#   targeting mismatch or extreme-score persons (see `Extreme` flag).
s_fit$targeting
# Look for: |Targeting| < ~0.5 logits is comfortable; larger absolute
#   values mean persons sit systematically above or below the facet
#   means under the package's sum-to-zero identification.
p_fit <- plot(fit, draw = FALSE)
p_fit$wright_map$data$plot

# JML is available for exploratory / fast iteration passes:
fit_jml <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "JML",
  model = "RSM",
  maxit = 30
)
summary(fit_jml)$overview[, c("Model", "Method", "Converged")]

# Latent regression (MML only) uses person-level background variables:
person_tbl <- unique(toy[c("Person")])
person_tbl$Grade <- seq_len(nrow(person_tbl))
person_tbl$Group <- rep(c("A", "B"), length.out = nrow(person_tbl))
fit_pop <- fit_mfrm(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  population_formula = ~ Grade + Group,
  person_data = person_tbl
)
summary(fit_pop)$population_overview
summary(fit_pop)$population_coding

# Binary responses are supported as ordered two-category scores:
```

```
set.seed(1)
binary_toy <- expand.grid(
  Person = paste0("P", 1:30),
  Item = paste0("I", 1:4),
  stringsAsFactors = FALSE
)
theta <- stats::rnorm(length(unique(binary_toy$Person)))
beta <- seq(-0.8, 0.8, length.out = length(unique(binary_toy$Item)))
eta <- theta[match(binary_toy$Person, unique(binary_toy$Person))] -
  beta[match(binary_toy$Item, unique(binary_toy$Item))]
binary_toy$Score <- stats::rbinom(nrow(binary_toy), 1, stats::plogis(eta))
fit_binary <- fit_mfrm(
  data = binary_toy,
  person = "Person",
  facets = "Item",
  score = "Score",
  model = "RSM",
  method = "JML",
  maxit = 30
)
fit_binary$summary[, c("Model", "Categories", "Converged")]

# Next steps after fitting:
diag <- diagnose_mfrm(fit, residual_pca = "none")
chk <- reporting_checklist(fit, diagnostics = diag)
head(chk$checklist[, c("Section", "Item", "DraftReady")])
```

`gpcm_capability_matrix` *Bounded GPCM Support Matrix*

Description

Public capability map for the current `GPCM` scope in `mfrmr`.

Use this helper when you need to answer a practical question quickly: which `GPCM` workflows are supported in this release, which are available only with explicit caveats, and which helpers remain blocked or deferred, plus the route to use instead when the requested helper is outside the current boundary.

The matrix is intentionally conservative. It is a release-scope statement, not a promise that every lower-level helper can be combined with `GPCM`. If a helper is not yet covered by the current validation boundary, it is listed as `blocked` or `deferred` even when related components already exist.

Usage

```
gpcm_capability_matrix(
  status = c("all", "supported", "supported_with_caveat", "blocked", "deferred")
)
```

Arguments

status	Which rows to return: "all" (default), "supported", "supported_with_caveat", "blocked", or "deferred".

Details

The current release treats GPCM as a bounded supported scope inside the core R package:

- fitting and core summaries are supported,
- posterior-scoring and information helpers are supported,
- residual-based diagnostics and strict marginal follow-up are supported as exploratory screens,
- direct slope-aware simulation-spec generation and parameter-recovery simulation are supported with caveats,
- fair_average_table() is supported with an explicit slope-aware element-conditional caveat,
- estimate_bias() is supported as conditional screening evidence with slope-aware information and profile-likelihood follow-up columns,
- summary-table appendix export is available for supported direct outputs,
- APA writer, visual summaries, QC pipelines, manifests, replay scripts, and fit-based export bundles are available only as caveated sensitivity-reporting surfaces with an explicit gpcm_boundary,
- package-native scorefile export is available with score-side caveats,
- role-based design evaluation and population forecasting are available as caveated bounded-GPCM sensitivity evidence,
- role-based diagnostic and signal-detection design screening helpers are available as caveated bounded-GPCM sensitivity evidence,
- full FACETS output-contract score-side review remains outside the validated GPCM boundary.

Why some helpers remain blocked:

- full FACETS output-contract score-side review depends on Rasch-family measure-to-score semantics plus delta-method SE machinery that are not yet generalized to the free-discrimination GPCM branch;
- APA writer, fit-based report/export bundles, visual summaries, and QC pipelines stay caveated because they must not turn unsupported score-side semantics into narrative or pass/fail outputs;
- diagnostic, signal-detection, design-forecast, and linking helpers stay caveated because their simulation/refit summaries must not become operational screening, scoring, or arbitrary-facet planning claims.

This boundary is aligned with the package's current validation evidence, including the targeted GPCM recovery snapshot and the public workflow checks.

Value

A data.frame with one row per public helper family and columns:

- `Area`
- `Helpers`
- `Status`
- `PrimaryUse`
- `Boundary`
- `Evidence`
- `RecommendedRoute`
- `NextValidationStep`

Typical workflow

1. Call `gpcm_capability_matrix()` before using `GPCM` in a new workflow.
2. Stay on rows marked `supported` or `supported_with_caveat` for the current release.
3. For `blocked` and `deferred` rows, read `RecommendedRoute` before choosing a substitute workflow.
4. Treat `blocked` rows as explicit non-support, not as temporary omissions.
5. Treat `deferred` rows as future-extension targets rather than part of the current user-facing support.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `compute_information()`, `predict_mfrm_units()`, `sample_mfrm_plausible_values()`
`reporting_checklist()`, mfrmr_workflow_methods, mfrmr-package

Examples

```
gpcm_capability_matrix()
gpcm_capability_matrix("supported")
gpcm_capability_matrix("blocked")
```

`gpcm_runtime_guard_coverage`

Bounded GPCM Route-Boundary Coverage

Description

Public table showing how blocked or deferred bounded-`GPCM` capability rows are handled by the current release.

Usage

```
gpcm_runtime_guard_coverage()
```

Details

gpcm_capability_matrix() is the user-facing support matrix. This helper records which public helpers stop with mfrmr_gpcm_scope_error when called on a bounded GPCM path and which capability rows have no public route yet and are therefore documented as future-extension scope.

Package checks use this table to keep out-of-scope GPCM behavior aligned with the capability matrix. A row with GuardMode = "runtime_error" should have ExpectedConditionClass = "mfrmr_gpcm_scope_error". A row with GuardMode = "roadmap_only" records a documented future-extension target with no public helper to call in the current release.

Value

A data.frame with columns:

- Area
- Helper
- Status
- GuardMode
- ExpectedConditionClass
- RecommendedRoute
- NextValidationStep
- TestRoute
- Notes

See Also

gpcm_capability_matrix(), mfrmr_workflow_methods, mfrmr-package

Examples

```
gpcm_runtime_guard_coverage()
```

---

gpcm_score_side_contract

Bounded GPCM Score-Side Export Contract

Description

Minimal contract table for the caveated bounded-GPCM scorefile route and the still-blocked full FACETS-style score-side review route.

Usage

```
gpcm_score_side_contract(
  status = c("all", "implemented_with_caveat", "required_for_full_facets_review",
    "validated_dependency")
)
```

Arguments

status	Which rows to return: "all" (default), "implemented_with_caveat", "required_for_full_facets_r or "validated_dependency".

Details

This helper does not enable full FACETS-style score-side review. It records the requirements that separate the current caveated facets_output_file_bundle(include = "score") route from a future facets_output_contract_review() route for bounded GPCM.

Use it as a release-maintenance checklist. Rows marked implemented_with_caveat support the current package-native bounded-GPCM scorefile route. Rows marked required_for_full_facets_review are still blockers for full FACETS-style output-contract review. Rows marked validated_dependency are already available in the package but are not sufficient by themselves to justify full FACETS score-side equivalence.

Value

A data.frame with columns:

- ContractArea
- Requirement
- CurrentStatus
- ReleaseBoundary
- ValidationTarget
- ExitCriterion

See Also

gpcm_capability_matrix(), gpcm_runtime_guard_coverage(), facets_output_contract_review(), facets_output_file_bundle()

Examples

```
gpcm_score_side_contract()
gpcm_score_side_contract("implemented_with_caveat")
```

---

import_erm_fit *Import an* eRm *fit to an mfrmr-compatible bundle*

Description

Extracts item / person parameters from an eRm::PCM() / eRm::RM() fit. Same caveats as import_mirt_fit().

Usage

```
import_erm_fit(fit, model = c("RSM", "PCM", "GPCM"), item_facet = "Item")
```

Arguments

fit	An object returned by eRm::PCM(), eRm::RM(), or eRm::RSM().
model	Same as import_mirt_fit().
item_facet	Name to assign to the item facet.

Value

An mfrm_imported_fit object.

See Also

import_mirt_fit(), import_tam_fit()

---

import_mirt_fit *Import an* mirt *fit to an mfrmr-compatible bundle*

Description

Extracts item, step, and person parameters from a mirt::mirt() fit and returns an mfrm_imported_fit object. The returned object has the public slots summary, facets$person, facets$others, steps, config, and source that the mfrmr plot and table helpers expect. With compute_fit = TRUE the importer also runs mirt::itemfit() and mirt::personfit() so Infit / Outfit columns are populated, and synthesises a mfrm_diagnostics-shape diagnostics slot consumable by downstream plot helpers (Wright map, QC dashboard, etc.).

Usage

```
import_mirt_fit(
  fit,
  model = c("RSM", "PCM", "GPCM"),
  item_facet = "Item",
  compute_fit = FALSE
)
```

Arguments

fit	An object returned by mirt::mirt() (a SingleGroupClass).
model	One of "RSM", "PCM", "GPCM". The importer does not infer the model from the mirt object; pass the model that was estimated.
item_facet	Name to assign to the item facet in the imported bundle (default "Item").
compute_fit	Logical. When TRUE, run mirt::itemfit() and mirt::personfit() to populate Infit / Outfit / OutfitZSTD columns on the returned facet tables, plus build a measurement-side mfrm_diagnostics bundle consumable by summary(), plot.mfrm_fit(), plot_qc_dashboard(), etc. Default FALSE keeps the importer fast (skeleton only).

Value

An `mfrm_imported_fit` object. Slots:

`summary` Model / method / N / LogLik / AIC / BIC.

`facets$person` Person ID, Estimate, SE, Extreme, plus Infit / Outfit / OutfitZSTD / Zh when `compute_fit = TRUE`.

`facets$others` Item-level estimates and slopes; with `compute_fit = TRUE`, also Infit / Outfit / S_X2 / RMSEA / df from `mirt::itemfit()`.

`steps` Per-item threshold parameters extracted from the IRT parameterisation (`b1`, ..., `b(K-1)`).

`config` List with the resolved `model` and `item_facet` used for the import; downstream plot and table helpers consult this to dispatch correctly on the imported bundle.

`diagnostics` `mfrm_diagnostics`-shape bundle when `compute_fit = TRUE`; `NULL` otherwise.

`source` Imported-from metadata.

Scope

Bundles bias / DIF / anchor / replay slots are explicitly not populated; full bidirectional import / export is planned for a future release.

See Also

`import_tam_fit()`, `import_erm_fit()`

`import_tam_fit` *Import a* `TAM` *fit to an mfrmr-compatible bundle*

Description

Extracts item / step / person parameters from a `TAM::tam.mml()`, `TAM::tam.jml()`, or `TAM::tam.mml.mfr()` fit. The multi-facet `tam.mml.mfr()` path is detected automatically and each non-person facet is mapped onto a row of `fit$facets$others` so downstream MFRM helpers (e.g. `plot_qc_dashboard()`) work on the imported object.

Usage

```
import_tam_fit(
  fit,
  model = c("RSM", "PCM", "GPCM"),
  item_facet = "Item",
  compute_fit = FALSE
)
```

Arguments

fit	An object returned by TAM::tam.mml(), TAM::tam.jml(), or TAM::tam.mml.mfr().
model	Same as import_mirt_fit().
item_facet	Name to assign to the item facet for the single-facet path. Ignored when the input is a multi-facet tam.mml.mfr fit (the original facet names are preserved).
compute_fit	Logical. When TRUE, run TAM::tam.fit() and TAM::tam.personfit() to populate Infit / Outfit columns on the returned facet tables, plus build a measurement-side mfrm_diagnostics bundle. Default FALSE.

Value

An mfrm_imported_fit object. Slots mirror import_mirt_fit().

See Also

import_mirt_fit(), import_erm_fit()

---

interaction_effect_table

Extract model-estimated facet interaction effects

Description

interaction_effect_table() returns the fixed-effect interaction block estimated by fit_mfrm() when facet_interactions is supplied. These are model-estimated deviations from the additive main-effects MFRM, not the residual screening statistics returned by estimate_bias().

Usage

```
interaction_effect_table(fit)
```

Arguments

fit	An mfrm_fit object returned by fit_mfrm().

Details

The current release supports two-way interactions between non-person facets, for example facet_interactions = "Rater:Criterion". Each interaction matrix is identified by zero marginal sums across both participating facets, so the interaction estimates are separable from the two main effects. Positive values indicate higher-than-expected scores for the facet-level combination under the additive model; negative values indicate lower-than-expected scores.

Use this table for confirmatory model review after specifying the facet pair of substantive interest. For exploratory screening without adding parameters to the fitted model, use estimate_bias() or estimate_all_bias().

Value

A tibble with one row per interaction cell. Returns an empty tibble when the fit has no model-estimated facet interactions.

See Also

`fit_mfrm()`, `estimate_bias()`, `compare_mfrm()`

`interrater_agreement_table` *Build an inter-rater agreement report*

Description

Build an inter-rater agreement report

Usage

```
interrater_agreement_table(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  context_facets = NULL,
  exact_warn = 0.5,
  corr_warn = 0.3,
  include_precision = TRUE,
  top_n = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`rater_facet`	Name of the rater facet. If `NULL`, inferred from facet names.
`context_facets`	Optional context facets used to match observations for agreement. If `NULL`, all remaining facets (including `Person`) are used.
`exact_warn`	Warning threshold for exact agreement.
`corr_warn`	Warning threshold for pairwise correlation.
`include_precision`	If `TRUE`, append rater severity spread indices from the facet precision summary when available.
`top_n`	Optional maximum number of pair rows to keep.

Details

This helper computes pairwise rater agreement on matched contexts and returns both a pair-level table and a one-row summary. The output is package-native and does not require knowledge of legacy report numbering.

Value

A named list with:

- `summary`: one-row inter-rater summary
- `pairs`: pair-level agreement table
- `settings`: applied options and thresholds

Interpreting output

- `summary`: overall agreement level, number/share of flagged pairs.
- `pairs`: pairwise exact agreement, correlation, and direction/size gaps.
- `settings`: applied facet matching and warning thresholds.

Pairs flagged by both low exact agreement and low correlation generally deserve highest calibration priority.

Typical workflow

1. Run with explicit `rater_facet` (and `context_facets` if needed).
2. Review `summary(ir)` and top flagged rows in `ir$pairs`.
3. Visualize with `plot_interrater_agreement()`.

Output columns

The `pairs` data.frame contains:

Rater1, Rater2 Rater pair identifiers.

N Number of matched-context observations for this pair.

Exact Proportion of exact score agreements.

ExpectedExact Expected exact agreement under chance.

Adjacent Proportion of adjacent (+/- 1 category) agreements.

MeanDiff Signed mean score difference (Rater1 - Rater2).

MAD Mean absolute score difference.

Corr Pearson correlation between paired scores.

Flag Logical; `TRUE` when Exact < `exact_warn` or Corr < `corr_warn`.

OpportunityCount, ExactCount, ExpectedExactCount, AdjacentCount Raw counts behind the agreement proportions.

The `summary` data.frame contains:

RaterFacet Name of the rater facet analyzed.

TotalPairs Number of rater pairs evaluated.

ExactAgreement Mean exact agreement across all pairs.

AgreementMinusExpected Observed exact agreement minus expected exact agreement.

MeanCorr Mean pairwise correlation.

FlaggedPairs, FlaggedShare Count and proportion of flagged pairs.

RaterSeparation, RaterReliability Severity-spread indices for the rater facet, reported separately from agreement.

See Also

diagnose_mfrm(), facets_chisq_table(), plot_interrater_agreement(), mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
ir <- interrater_agreement_table(fit, rater_facet = "Rater")
# One-row overview: ExactAgreement, ExpectedExactAgreement, MeanCorr,
# RaterSeparation, and RaterReliability are the headline reportable
# statistics.
ir$summary
# Per-pair detail (Rater1 vs Rater2 with Exact, Adjacent, Corr, MAD).
head(ir$pairs)
p_ir <- plot(ir, draw = FALSE)
p_ir$data$plot
```

---

launch_mfrmr_viewer *Launch a local Shiny viewer for an mfrm_results object*

Description

launch_mfrmr_viewer() opens a local Shiny app for reading the object returned by mfrm_results(). It is intentionally a viewer over an existing comprehensive results object, not a new estimation interface.

Usage

```
launch_mfrmr_viewer(
  x,
  top_n = 100L,
  launch.browser = TRUE,
  port = NULL,
  host = getOption("shiny.host", "127.0.0.1"),
  display.mode = c("auto", "normal", "showcase"),
  return_app = FALSE,
  ...
)
```

Arguments

x	An mfrm_results() object.
top_n	Maximum number of table-index rows shown in the summary payload.
launch.browser	Passed to shiny::runApp().
port	Optional port passed to shiny::runApp(). NULL lets Shiny choose its default.
host	Host passed to shiny::runApp(). Defaults to the local host.
display.mode	Passed to shiny::runApp().
return_app	Logical; if TRUE, return the Shiny app object without running it. This is useful for embedding or testing.
...	Additional arguments passed to shiny::runApp() when return_app = FALSE.

Details

The viewer assumes that fitting, diagnostics, and section selection have already happened through mfrm_results(). This keeps GUI exploration separate from reproducible analysis setup: the Replay tab displays the mfrm_results() scaffold stored in the result object.

The app includes tabs for overview/triage, QC evidence, APA-style report text when include = "publication" or "apa" was used, available bias screens, pathway plotting, unexpected-response inspection, generic tables, generic plot routes, and replay code. QC, Report, Bias, and Pathway/Misfit tabs show local section-status tables so unavailable or not-requested sections are visible where users look for them. Bias-interaction follow-up still requires an explicit facet-pair decision outside the viewer.

shiny is an optional dependency. Install it before using this viewer: install.packages("shiny").

Value

Invisibly returns the value from shiny::runApp(), or the Shiny app object when return_app = TRUE.

See Also

mfrm_results(), mfrm_results_interactive()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "JML",
  maxit = 30
)
res <- mfrm_results(fit, include = c("fit", "diagnostics", "tables"))

if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
```

```
  launch_mfrmr_viewer(res)
}
```

`list_mfrmr_data` *List packaged simulation datasets*

Description

List packaged simulation datasets

Usage

```
list_mfrmr_data()
```

Details

Use this helper when you want to select packaged data programmatically (e.g., inside scripts, loops, or shiny/streamlit wrappers).

Typical pattern:

1. call `list_mfrmr_data()` to see available keys.
2. pass one key to `load_mfrmr_data()`.

Value

Character vector of dataset keys accepted by `load_mfrmr_data()`.

Interpreting output

Returned values are canonical dataset keys accepted by `load_mfrmr_data()`.

Typical workflow

1. Capture keys in a script (`keys <- list_mfrmr_data()`).
2. Select one key by index or name.
3. Load data via `load_mfrmr_data()` and continue analysis.

See Also

`load_mfrmr_data()`, ej2021_data

Examples

```
keys <- list_mfrmr_data()
keys
d <- load_mfrmr_data(keys[1])
head(d)
```

`load_mfrmr_data` *Load a packaged simulation dataset*

Description

Load a packaged simulation dataset

Usage

```
load_mfrmr_data(
  name = c("example_core", "example_bias", "study1", "study2", "combined",
    "study1_itercal", "study2_itercal", "combined_itercal")
)
```

Arguments

`name`	Dataset key. One of values from `list_mfrmr_data()`.

Details

`load_mfrmr_data("<key>")` is the canonical loader for the packaged datasets and the entry point used across the package help and vignettes. The equivalent base-R alternative `data("mfrmr_<key>", package = "mfrmr")` remains available for users who prefer the full `data()` spelling; both paths return identical long-format data frames and are supported long-term.

All returned datasets include the core long-format columns `Study`, `Person`, `Rater`, `Criterion`, and `Score`. Some datasets, such as the packaged documentation examples, also include auxiliary variables like `Group` for DIF/bias demonstrations.

Value

A data.frame in long format.

Interpreting output

The return value is a plain long-format `data.frame`, ready for direct use in `fit_mfrm()` without additional reshaping.

Typical workflow

1. list valid names with `list_mfrmr_data()`.
2. load one dataset key with `load_mfrmr_data(name)`.
3. fit a model with `fit_mfrm()` and inspect with `summary()` / `plot()`.

See Also

`list_mfrmr_data()`, ej2021_data

Examples

```
data("mfrmr_example_core", package = "mfrmr")
head(mfrmr_example_core)

d <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  data = d,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "JML",
  maxit = 30
)
summary(fit)
```

---

make_anchor_table *Build an anchor table from fitted estimates*

Description

Build an anchor table from fitted estimates

Usage

```
make_anchor_table(fit, facets = NULL, include_person = FALSE, digits = 6)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`facets`	Optional subset of facets to include.
`include_person`	Include person estimates as anchors.
`digits`	Rounding digits for anchor values.

Details

This function exports estimated facet parameters as an anchor table for use in subsequent calibrations. This is the standard approach for **linking** across administrations: a reference run establishes the measurement scale, and anchored re-analyses place new data on that same scale.

Anchor values should be exported from a well-fitting reference run with adequate sample size. If the reference model has convergence issues or large misfit, the exported anchors may propagate instability. Re-run `review_mfrm_anchors()` on the receiving data to verify compatibility before estimation.

The `digits` parameter controls rounding precision. Use at least 4 digits for research applications; excessive rounding (e.g., 1 digit) can introduce avoidable calibration error.

Value

A data.frame with `Facet`, `Level`, and `Anchor`.

Interpreting output

- `Facet`: facet name to be anchored in later runs.
- `Level`: specific element/level name inside that facet.
- `Anchor`: fixed logit value (rounded by `digits`).

Typical workflow

1. Fit a reference run with `fit_mfrm()`.
2. Export anchors with `make_anchor_table(fit)`.
3. Pass selected rows back into `fit_mfrm(..., anchors = ...)`.

See Also

`fit_mfrm()`, `review_mfrm_anchors()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
anchors_tbl <- make_anchor_table(fit)
head(anchors_tbl)
summary(anchors_tbl$Anchor)
```

---

measurable_summary_table

Build a measurable-data summary

Description

Build a measurable-data summary

Usage

```
measurable_summary_table(fit, diagnostics = NULL)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.

Details

This helper consolidates measurable-data diagnostics into a dedicated report bundle: run-level summary, facet coverage, category usage, and subset (connected-component) information.

`summary(t5)` is supported through `summary()`. `plot(t5)` is dispatched through `plot()` for class `mfrm_measurable` (`type = "facet_coverage"`, `"category_counts"`, `"subset_observations"`).

Value

A named list with:

- `summary`: one-row measurable-data summary
- `facet_coverage`: per-facet coverage summary
- `category_stats`: category-level usage/fit summary
- `subsets`: subset summary table (when available)

Interpreting output

- `summary`: overall measurable design status.
- `facet_coverage`: spread/precision by facet.
- `category_stats`: category usage and fit context.
- `subsets`: connectivity diagnostics (fragmented subsets reduce comparability).

Typical workflow

1. Run `measurable_summary_table(fit)`.
2. Check `summary(t5)` for subset/connectivity warnings.
3. Use `plot(t5, ...)` to inspect facet/category/subset views.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `summary` data.frame (one row) contains:

Observations, TotalWeight Total observations and summed weight.

Persons, Facets, Categories Design dimensions.

ConnectedSubsets Number of connected subsets.

LargestSubsetObs, LargestSubsetPct Largest subset coverage.

The `facet_coverage` data.frame contains:

Facet Facet name.

Levels Number of estimated levels.

MeanSE Mean standard error across levels.

MeanInfit, MeanOutfit Mean fit statistics across levels.

MinEstimate, MaxEstimate Measure range for this facet.

The `category_stats` data.frame contains:

Category Score category value.

Count, Percent Observed count and percentage.

Infit, Outfit, InfitZSTD, OutfitZSTD Category-level fit.

ExpectedCount, DiffCount, LowCount Expected-observed comparison and low-count flag.

See Also

`diagnose_mfrm()`, `rating_scale_table()`, `describe_mfrm_data()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
t5 <- measurable_summary_table(fit)
summary(t5)
p_t5 <- plot(t5, draw = FALSE)
p_t5$data$plot
```

---

mfrmr_compatibility_layer

mfrmr Compatibility Layer Map

Description

Guide to the legacy-compatible wrappers and text/file exports in `mfrmr`. Use this page when you need continuity with older compatibility-oriented workflows, fixed-width reports, or graph/score file style outputs.

This compatibility layer currently applies mainly to diagnostics-based `RSM` / `PCM` workflows. First-release `GPCM` fits now also support graph-only compatibility-style exports, while scorefile and diagnostics-driven compatibility outputs remain limited to `RSM` / `PCM`. Treat this layer as a presentation/contract surface, not as a claim of FACETS or ConQuest numerical equivalence.

SPSS is treated differently from FACETS and ConQuest: `mfrmr` currently supports table/data-frame/CSV handoff for SPSS-oriented reporting workflows, but it does not generate SPSS syntax, write native SPSS system files, execute SPSS estimators, or claim SPSS numerical equivalence.

When to use this layer

- You are reproducing an older workflow that expects one-shot wrappers.
- You need fixed-width text blocks for console, logs, or archival handoff.
- You need graphfile or scorefile style outputs for downstream legacy tools.
- You are checking column coverage and metric consistency against a FACETS-style output contract.

When not to use this layer

- For standard estimation, use `fit_mfrm()` plus `diagnose_mfrm()`.
- For report bundles, use mfrmr_reports_and_tables.
- For manuscript text, use `build_apa_outputs()` and `reporting_checklist()`.
- For visual follow-up, use mfrmr_visual_diagnostics.

Compatibility map

`facets_positioning_guide()` User-facing wording for the package's relationship to FACETS. Use before describing compatibility outputs in a report or migration note.

`run_mfrm_facets()` One-shot legacy-compatible wrapper that fits, diagnoses, and returns key tables in one object.

`mfrmRFacets()` Alias for `run_mfrm_facets()` kept for continuity.

`build_fixed_reports()` Fixed-width interaction and pairwise text blocks. Best when a text-only compatibility artifact is required.

`facets_output_file_bundle()` Graphfile/scorefile style CSV and fixed-width exports for legacy pipelines.

`write_mfrm_residual_file()` Package-native observation-level residual CSV/TSV export for reviewer, spreadsheet, or external QC handoff.

`write_mfrm_subset_file()` Package-native connected-subset summary and node-membership CSV/TSV export for linking review handoff.

`facets_output_contract_review()` Column and metric review against the FACETS-style output-contract specification. Use only when an explicit output-contract review is part of the task; it checks the package output contract and does not imply external FACETS equivalence.

Preferred replacements

- Instead of `run_mfrm_facets()`, prefer: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()`.
- Instead of `build_fixed_reports()`, prefer: `bias_interaction_report()` -> `build_apa_outputs()`.
- Instead of `facets_output_file_bundle()`, prefer: `category_curves_report()` or `category_structure_report()` plus `export_mfrm_bundle()`.
- For residual or subset handoff, prefer `write_mfrm_residual_file()` and `write_mfrm_subset_file()` over reusing graph/score compatibility exports.
- Instead of `facets_output_contract_review()` for routine QA, prefer: `reference_case_review()` for package-native completeness review or `reference_case_benchmark()` for packaged benchmark cases.

Practical migration rules

- Start FACETS-facing reports with `facets_positioning_guide()` when readers might otherwise assume FACETS numerical reproduction.
- Keep compatibility wrappers only where a downstream consumer truly needs the old layout or fixed-width format.

- For new scripts, start from package-native bundles and add compatibility outputs only at the export boundary.
- Treat compatibility outputs as presentation contracts, not as the primary analysis objects.
- Use `compatibility_alias_table()` when you need to check which aliases are still retained and which package-native names should be used in new code.
- Use `reporting_checklist(fit)$software_scope` to review the current FACETS, ConQuest, and SPSS relationship wording for a fitted analysis.

Retained table-field names

- `row_review` is the data-quality table field used to document row filtering. FACETS-style column and metric contract results are exposed as `column_review` and `metric_checks`.
- Prediction outputs expose row-preparation and person-omission traceability as `row_review` and `population_review`.
- `hierarchical_review`, `shrinkage_review`, and `nesting_review` are manifest or model-comparison traceability fields. They are not callable helper names; user-facing helper names use review/check terminology.

Typical workflow

- Legacy handoff: `run_mfrm_facets()` -> `build_fixed_reports()` -> `facets_output_file_bundle()` plus `write_mfrm_residual_file()` / `write_mfrm_subset_file()` only when those standalone files are needed.
- Mixed workflow: `RSM` / `PCM`: `fit_mfrm()` -> `diagnose_mfrm()` -> `build_apa_outputs()` -> compatibility export only if required. bounded `GPCM`: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()` -> graph-only compatibility export only when a legacy handoff truly requires it.
- FACETS output-contract review: `fit_mfrm()` -> `diagnose_mfrm()` -> `facets_output_contract_review()`.

Companion guides

- For FACETS coverage and boundary wording, see `facets_positioning_guide()` and `facets_feature_coverage()`
- For standard reports/tables, see mfrmr_reports_and_tables.
- For manuscript-draft reporting, see mfrmr_reporting_and_apa.
- For visual diagnostics, see mfrmr_visual_diagnostics.
- For linking and DFF workflows, see mfrmr_linking_and_dff.
- For end-to-end routes, see mfrmr_workflow_methods.

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]

run <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
```

```
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
summary(run)
compatibility_alias_table("functions")

fixed <- build_fixed_reports(
  estimate_bias(
    run$fit,
    run$diagnostics,
    facet_a = "Rater",
    facet_b = "Criterion",
    max_iter = 1
  ),
  branch = "original"
)
names(fixed)
```

`mfrmr_example_data` *Purpose-built example datasets for package help pages*

Description

Compact synthetic many-facet datasets designed for documentation examples. Both datasets are large enough to avoid tiny-sample toy behavior while remaining fast in `R CMD check` examples.

Format

A data.frame with 6 columns:

Study Example dataset label (`"ExampleCore"` or `"ExampleBias"`).

Person Person/respondent identifier.

Rater Rater identifier.

Criterion Criterion facet label.

Score Observed category score on a four-category scale (1–4).

Group Balanced grouping variable used in DFF/DIF examples (`"A"` / `"B"`).

Details

Available data objects:

- `mfrmr_example_core`
- `mfrmr_example_bias`

mfrmr_example_core is generated from a single latent trait plus rater and criterion main effects, making it suitable for general fitting, plotting, and reporting examples.

mfrmr_example_bias starts from the same basic design but adds:

- a known Group x Criterion effect (Group B is advantaged on Language)
- a known Rater x Criterion interaction (R04 x Accuracy)

This lets differential-functioning and bias-analysis help pages demonstrate non-null findings.

Data dimensions

Dataset	Rows	Persons	Raters	Criteria	Groups
example_core	768	48	4	4	2
example_bias	384	48	4	4	2

Suggested usage

- Use mfrmr_example_core for fitting, diagnostics, design-weighted precision curves, and generic plots/reports.
- Use mfrmr_example_bias for analyze_dff(), analyze_dif(), dif_interaction_table(), plot_dif_heatmap(), and estimate_bias().

Both objects can be loaded either with load_mfrmr_data() or directly via data("mfrmr_example_core", package = "mfrmr") / data("mfrmr_example_bias", package = "mfrmr").

Source

Synthetic documentation data generated from rating-scale Rasch facet designs with fixed seeds in data-raw/make-example-data.R.

Examples

```
data("mfrmr_example_core", package = "mfrmr")
table(mfrmr_example_core$Score)
table(mfrmr_example_core$Group)
```

---

mfrmr_interval_guide *Confidence-interval and uncertainty route guide*

Description

Return a compact map of the public mfrmr routes that can expose confidence intervals or interval-like uncertainty displays. Use this when you need to know which helper accepts show_ci or ci_level, which columns to look for in draw = FALSE output, and how strongly the resulting interval should be interpreted.

Usage

```
mfrmr_interval_guide(
  scope = c("all", "visual", "table", "reporting", "fit", "bias", "linking", "gpcm",
    "equivalence", "hierarchical", "shrinkage")
)
```

Arguments

scope	Which rows to return: "all" (default), "visual", "table", "reporting", "fit", "bias", "linking", "gpcm", "equivalence", "hierarchical", or "shrinkage".

Details

The guide is deliberately conservative. It is a namespace and interpretation map, not a fitted result and not proof that a given interval is available for a particular run. For run-specific availability, call the listed helper with `draw = FALSE` or inspect the relevant result table.

Most rows use `ci_level = 0.95` by default. Some intervals are model-based Wald intervals, some are delta-method intervals, some are profile or profile-like intervals when available, and some are plotting overlays around already-estimated quantities. The `Basis` and `InterpretationBoundary` columns are the important guardrails.

Value

A data.frame with columns:

- `Route`
- `Scope`
- `PrimaryHelper`
- `DisplayRoute`
- `DefaultLevel`
- `IntervalColumns`
- `Basis`
- `UseFor`
- `InterpretationBoundary`
- `GPCMStatus`
- `Notes`

See Also

mfrmr_visual_diagnostics, `visual_reporting_template()`, `plot_fair_average()`, `plot_bias_interaction()`, `plot_displacement()`, `plot_wright_unified()`, `plot_rater_severity_profile()`, `plot_apa_figure_one()`, `fit_measures_table()`

Examples

```
mfrmr_interval_guide()
mfrmr_interval_guide("visual")[, c("Route", "DisplayRoute", "Basis")]
mfrmr_interval_guide("gpcm")[, c("Route", "GPCMStatus", "InterpretationBoundary")]
```

mfrmr_linking_and_dff *mfrmr Linking and DFF Guide*

Description

Package-native guide to checking connectedness, building anchor-based links, monitoring drift, and screening differential facet functioning (DFF) in `mfrmr`.

Start with the linking question

- "Is the design connected enough to support a common scale?" Use `subset_connectivity_report()` and `plot(..., type = "design_matrix")`.
- "Which elements can I export as anchors from an existing fit?" Use `make_anchor_table()` and `review_mfrm_anchors()`.
- "How do I anchor a new administration to a baseline?" Use `anchor_to_baseline()`.
- "Have common elements drifted across separately fitted waves?" Use `detect_anchor_drift()` and `plot_anchor_drift()`.
- "Can I synthesize anchor review, drift, and chain evidence into one review?" Use `build_linking_review()`.
- "Do specific facet levels function differently across groups?" Use `analyze_dff()`, `plot_dif_heatmap()`, and `plot_dif_summary()`.

Recommended linking route

1. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`.
2. Check connectedness with `subset_connectivity_report()`.
3. Build or review anchors with `make_anchor_table()` and `review_mfrm_anchors()`.
4. Use `anchor_to_baseline()` when you need to place raw new data onto a baseline scale.
5. Use `build_equating_chain()` only as a screened linking aid across already fitted waves.
6. Use `detect_anchor_drift()` for stability monitoring on separately fitted waves.
7. Use `build_linking_review()` when you need one operational synthesis object rather than separate anchor/drift/chain tables.
8. Run `analyze_dff()` only after checking connectivity and common-scale evidence.

Which helper answers which task

`subset_connectivity_report()` Summarizes connected subsets, bottleneck facets, and design-matrix coverage.

`make_anchor_table()` Extracts reusable anchor candidates from a fit.

`anchor_to_baseline()` Anchors new raw data to a baseline fit and returns anchored diagnostics plus a consistency check against the baseline scale.

`detect_anchor_drift()` Compares fitted waves directly to flag unstable anchor elements.

`build_equating_chain()` Accumulates screened pairwise links across a series of administrations or forms.

`build_linking_review()` Synthesizes anchor review, drift, and screened-chain evidence into one operational review surface.

`analyze_dff()` Screens differential facet functioning with residual or refit methods, using screening-only language unless linking and precision support stronger interpretation.

Practical linking rules

- Check connectedness before interpreting subgroup or wave differences.
- Use DFF outputs as screening results when common-scale linking is weak.
- Always name the facet, facet level, and group pair involved in a DFF contrast. A generic "DIF exists" statement is not interpretable in a many-facet design.
- Residual-method DFF classifications are screening labels. ETS A/B/C labels require refit output whose `ClassificationSystem` is `"ETS"`.
- Treat drift flags as prompts for review, not automatic evidence that an anchor must be removed.
- Treat `LinkSupportAdequate = FALSE` as a weak-link warning: at least one linking facet retained fewer than 5 common elements after screening.
- Rebuild anchors from a defensible baseline rather than chaining unstable links by hand.

Typical workflow

- Cross-sectional linkage review: `fit_mfrm()` -> `diagnose_mfrm()` -> `subset_connectivity_report()` -> `plot(..., type = "design_matrix")`.
- Baseline placement review: `make_anchor_table()` -> `anchor_to_baseline()` -> `diagnose_mfrm()`.
- Multi-wave drift review: fit each wave separately -> `detect_anchor_drift()` -> `build_linking_review()` -> `plot_anchor_drift()`.
- Group comparison route: `subset_connectivity_report()` -> `analyze_dff()` -> `dif_report()` -> `plot_dif_heatmap()` / `plot_dif_summary()`.

Companion guides

- For visual follow-up, see mfrmr_visual_diagnostics.
- For report/table selection, see mfrmr_reports_and_tables.
- For end-to-end routes, see mfrmr_workflow_methods.
- For a longer walkthrough, see `vignette("mfrmr-linking-and-dff", package = "mfrmr")`.

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
```

```
  quad_points = 7,
  maxit = 30
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")

subsets <- subset_connectivity_report(fit, diagnostics = diag)
subsets$summary[, c("Subset", "Observations", "ObservationPercent")]

dff <- analyze_dff(fit, diag, facet = "Rater", group = "Group", data = toy)
head(dff$dif_table[, c("Level", "Group1", "Group2",
                       "Classification", "ClassificationSystem")])
```

`mfrmr_output_guide` *Choose an mfrmr output helper by user goal*

Description

`mfrmr_output_guide()` returns a compact table for choosing among the main table, report, review, bundle, export, and compatibility helpers. It is a user-facing map, not an analysis result.

Usage

```
mfrmr_output_guide(
  scope = c("all", "public", "entry", "viewer", "binary", "tables", "reports", "reviews",
    "bundles", "exports", "compatibility", "gpcm", "simulation", "linking", "network",
      "response_time", "facets", "conquest", "r")
)
```

Arguments

`scope`	Which rows to return. `"all"` returns the full guide. `"public"` returns the small top-level public surface for most users. `"entry"` returns the recommended first-screen routes. `"viewer"` returns local-viewer routes built around `mfrm_results(include = ...)`. `"binary"` returns the two-category person-item Rasch route and checks. Other values filter to one output family or to bounded-GPCM-relevant routes. `"linking"` returns anchor, drift, and equating route rows. `"simulation"` and `"network"` return advanced design-review rows. `"response_time"` returns descriptive response-time QC rows. `"facets"`, `"conquest"`, and `"r"` return user-pathway rows for people arriving from those workflows.

Details

Naming convention used by the guide:

- `*_table`: focused table or table-like result for one evidence source
- `*_report`: multi-table evidence bundle for a reporting question
- `*_review`: status, interpretation, or decision-support object

- `*_bundle`: reusable collection of tables/metadata for handoff
- `export_*`: writes files or appendix artifacts

Value

A data.frame with one row per recommended route and columns:

- `Scope`
- `Question`
- `OutputFamily`
- `Lifecycle`
- `UserLevel`
- `APILayer`
- `ObjectRole`
- `DecisionBoundary`
- `RecommendedEntry`
- `MainFunction`
- `UseWhen`
- `TypicalInput`
- `NextStep`
- `GPCMStatus`
- `Notes`

First-screen route

Use `mfrmr_output_guide("public")` when you want the shortest top-level API map. Use `mfrmr_output_guide("entry")` when you specifically want first-screen creation routes. The entry guide points new scripts to `fit_mfrm()` -> `diagnose_mfrm()` -> `mfrm_results()`, existing fits to `mfrm_results()`, and exploratory console work to `mfrm_results_interactive()`. After creating `res`, use `summary(res)$next_actions` to choose the next purpose-specific helper. Use `mfrmr_output_guide("viewer")` when the next step is the optional local Shiny reader; it shows which `include` preset to use before calling `launch_mfrmr_viewer()`.

How to use this guide

Treat `MainFunction` as the route to try next and `UseWhen` as the guardrail. The guide is not a replacement for the help pages of the listed functions; it is a namespace map for deciding which page to open. For bounded `GPCM`, use `scope = "gpcm"` to find both the support matrix and the table that explains how out-of-scope routes are handled.

Examples

```
public <- mfrmr_output_guide("public")
public[, c("Question", "APILayer", "ObjectRole", "MainFunction")]

entry <- mfrmr_output_guide("entry")
entry[, c("Question", "Lifecycle", "UserLevel", "MainFunction")]
```

```
reviews <- mfrmr_output_guide("reviews")
reviews[, c("Question", "MainFunction", "UseWhen")]

mfrmr_output_guide("gpcm")[, c("Question", "MainFunction", "GPCMStatus")]
mfrmr_output_guide("simulation")[, c("Question", "Lifecycle")]
mfrmr_output_guide("linking")[, c("Question", "MainFunction")]
mfrmr_output_guide("facets")[, c("Question", "MainFunction")]
mfrmr_output_guide("binary")[, c("Question", "MainFunction")]
mfrmr_output_guide("viewer")[, c("Question", "MainFunction")]
mfrmr_output_guide("response_time")[, c("Question", "MainFunction")]
```

`mfrmr_reporting_and_apa`

mfrmr Reporting and APA Guide

Description

Package-native guide to moving from fitted model objects to manuscript-draft text, tables, notes, and revision checklists in `mfrmr`.

This guide currently applies fully to diagnostics-based `RSM` / `PCM` workflows. First-release `GPCM` fits now support `reporting_checklist()`, `precision_review_report()`, direct curve/graph and residual table helpers, and caveated APA/QC/export bundles. Use `gpcm_capability_matrix()` when you need the formal boundary for the current `GPCM` reporting path.

In particular, bounded `GPCM` `build_apa_outputs()`, `build_visual_summaries()`, `run_qc_pipeline()`, `build_mfrm_manifest()`, `build_mfrm_replay_script()`, and `export_mfrm_bundle()` outputs include explicit `gpcm_boundary` caveats. Full FACETS-style score-side contract review remains blocked. Scorefile export, design forecasting, diagnostic/signal-detection screening, and linking synthesis use their own caveated `GPCM` routes and should not be treated as automatic operational-scoring evidence.

Start with the reporting question

- "Which parts of this run are draft-complete, and with what caveats?" Use `reporting_checklist()`.
- "How should I phrase the model, fit, and precision sections?" For `RSM` / `PCM`, use `build_apa_outputs()`.
- "Which tables should I hand off to a manuscript or appendix?" Use `build_summary_table_bundle()`, `export_summary_appendix()`, `apa_table()`, and `facet_statistics_report()`.
- "How do I explain model-based vs exploratory precision?" Use `precision_review_report()` and `summary(diagnose_mfrm(...))`.
- "Which caveats need to appear in the write-up?" Use `reporting_checklist()` first, then `build_apa_outputs()`.
- "How should I start figure captions or visual-results wording?" Use `visual_reporting_template()` for conservative caption and results sentence starters, then verify availability with `reporting_checklist()$visual_sc`

Recommended reporting route

1. Fit with `fit_mfrm()`.
2. Build diagnostics with `diagnose_mfrm()`.
3. Review precision strength with `precision_review_report()` when inferential language matters.
4. Run `reporting_checklist()` to identify missing sections, caveats, and next actions. Use the `"Visual Displays"` rows as the figure-routing layer for the current run.
5. When strict marginal rows are available, follow up with `plot_marginal_fit()` and `plot_marginal_pairwise()` before finalizing the narrative around local misfit.
6. Create manuscript-draft prose and metadata with `build_apa_outputs()`. For bounded `GPCM`, treat the APA/QC/export stack as caveated sensitivity-reporting output and keep its `gpcm_boundary` visible.
7. Convert summary outputs to reusable table bundles with `build_summary_table_bundle()`, review the bundle with `summary()` / `plot()`, then convert specific components to handoff tables with `apa_table()` or export them directly with `export_summary_appendix()`.

Which helper answers which task

`reporting_checklist()` Turns current analysis objects into a prioritized revision guide with `DraftReady`, `Priority`, and `NextAction`. `DraftReady` means "ready to draft with the documented caveats"; `ReadyForAPA` is retained as a backward-compatible alias, and neither field means "formal inference is automatically justified". The `"Visual Displays"` rows also mirror the public plot family, so the checklist doubles as a figure-routing surface.

`build_apa_outputs()` Builds shared-contract prose, table notes, captions, and a section map from the current fit and diagnostics.

`build_summary_table_bundle()` Turns supported `summary()` outputs into named `data.frame` tables plus an index for manuscript or appendix handoff, and now also supports bundle-level `summary()` / `plot()` for role coverage and numeric QC.

`export_summary_appendix()` Writes those validated summary-table bundles to CSV and optional HTML appendix artifacts without requiring a full fit-based export bundle.

`apa_table()` Produces reproducible base-R tables with APA-oriented labels, notes, and captions.

`precision_review_report()` Summarizes whether precision claims are model-based, hybrid, or exploratory.

`facet_statistics_report()` Provides facet-level summaries that often feed result tables and appendix material.

`build_visual_summaries()` Prepares publication-oriented figure data that can be cited from the report text.

`visual_reporting_template()` Provides conservative figure placement, caption-starter, results-wording, and overclaim-avoidance guidance for public visual helpers.

Practical reporting rules

- Treat `reporting_checklist()` as the gap finder and `build_apa_outputs()` as the writing engine.

- Use the checklist's "Visual Displays" rows to decide whether the next follow-up should be plot_qc_dashboard(), plot_marginal_fit(), plot_residual_pca(), plot_bias_interaction(), or another public plot.
- Use visual_reporting_template() to draft visual captions and results-sentence starters, but do not paste the skeletons without checking the actual fit, diagnostics, and study context.
- Phrase formal inferential claims only when the precision tier is model-based.
- Keep bias and differential-functioning outputs in screening language unless the current precision layer and linking evidence justify stronger claims.
- Treat DraftReady (and the legacy alias ReadyForAPA) as a drafting-readiness flag, not as a substitute for methodological review.
- Rebuild APA outputs after major model changes instead of editing old text by hand.
- For bounded GPCM, use APA/QC/export helpers only as caveated sensitivity-reporting surfaces and keep full FACETS-style score-side review outside this route.

Typical workflow

- Manuscript-first route: fit_mfrm() -> diagnose_mfrm() -> reporting_checklist() -> build_apa_outputs() -> build_summary_table_bundle() -> summary() / plot() -> apa_table(), export_summary_appendix(), or export_mfrm_bundle()(include = c("summary_tables", "html")). For RSM / PCM final reports, prefer method = "MML" and diagnostic_mode = "both" in the diagnostics step. For bounded GPCM, use the same fit-based reporting/export family only as caveated sensitivity-reporting output and inspect its gpcm_boundary rows before writing claims.
- Appendix-first route: facet_statistics_report() -> apa_table() -> build_visual_summaries() -> build_apa_outputs().
- Precision-sensitive route: diagnose_mfrm() -> precision_review_report() -> reporting_checklist() -> build_apa_outputs().
- bounded GPCM route: diagnose_mfrm() -> precision_review_report() -> reporting_checklist() -> direct residual/category/information helpers -> caveated build_apa_outputs(), build_visual_summaries(), run_qc_pipeline(), or export_mfrm_bundle() as needed.

Companion guides

- For report/table selection, see mfrmr_reports_and_tables.
- For end-to-end analysis routes, see mfrmr_workflow_methods.
- For visual follow-up, see mfrmr_visual_diagnostics.
- For the bounded GPCM support statement, see gpcm_capability_matrix.
- For a longer walkthrough, see vignette("mfrmr-reporting-and-apa", package = "mfrmr").

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  person = "Person",
```

```
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")

checklist <- reporting_checklist(fit, diagnostics = diag)
visual_reporting_template("manuscript")[, c("FigureFamily", "CaptionSkeleton")]
head(checklist$checklist[, c("Section", "Item", "DraftReady", "NextAction")])
subset(
  checklist$checklist,
  Section == "Visual Displays",
  c("Item", "Available", "NextAction")
)

apa <- build_apa_outputs(fit, diagnostics = diag)
apa$section_map[, c("SectionId", "Available")]

tbl <- apa_table(fit, which = "summary")
tbl$caption
bundle <- build_summary_table_bundle(checklist)
bundle$table_index
apa_from_bundle <- apa_table(bundle, which = "section_summary")
apa_from_bundle$caption
```

`mfrmr_reports_and_tables` *mfrmr Reports and Tables Map*

Description

Quick guide to choosing the right report or table helper in `mfrmr`. Use this page when you know the reporting question but have not yet decided which bundle, table, or reporting helper to call.

Start with the question

- "How should I document the model setup and run settings?" Use `specifications_report()`.
- "Was data filtered, dropped, or mapped in unexpected ways?" Use `data_quality_report()` and `describe_mfrm_data()`.
- "Did estimation converge cleanly and how formal is the precision layer?" Use `estimation_iteration_report()` and `precision_review_report()`.
- "Which facets are measurable, variable, or weakly separated?" Use `facet_statistics_report()`, `measurable_summary_table()`, and `facets_chisq_table()`.

- "Are score categories functioning in a usable sequence?" Use `rating_scale_table()`, `category_structure_report` and `category_curves_report()`.
- "Is the design linked well enough across subsets, forms, or waves?" Use `subset_connectivity_report()` and `plot_anchor_drift()`.
- "What should go into the manuscript text and tables?" For `RSM` / `PCM`, use `reporting_checklist()`, `build_apa_outputs()`, and `build_summary_table_bundle()` or `export_summary_appendix()`. For bounded `GPCM`, use the same route only where `gpcm_capability_matrix()` marks it as `supported_with_caveat`: direct table/plot helpers, summary-table appendix export, caveated `build_apa_outputs()`, and caveated `export_mfrm_bundle()` are available with a `gpcm_boundary`; score-side exports and design-forecasting evidence use their own caveated or blocked `GPCM` routes.
- "Did a simulation recover the known generating parameters well enough?" Use `evaluate_mfrm_recovery()` for the recovery study, `assess_mfrm_recovery()` for the adequacy checklist, and then `build_summary_table_bundle` or `export_summary_appendix()` for the appendix handoff.

Recommended report route

1. Start with `specifications_report()` and `data_quality_report()` to document the run and confirm usable data.
2. Continue with `estimation_iteration_report()` and `precision_review_report()` to judge convergence and inferential strength.
3. Use `facet_statistics_report()` and `subset_connectivity_report()` to describe spread, linkage, and measurability.
4. Add `rating_scale_table()`, `category_structure_report()`, and `category_curves_report()` to document scale functioning.
5. For `RSM` / `PCM`, finish with `reporting_checklist()` and `build_apa_outputs()` for manuscript-oriented output, then `build_summary_table_bundle()` for reusable handoff tables or `export_summary_appendix()` for direct appendix export. For bounded `GPCM`, the same report/export route is available only as a caveated sensitivity-reporting layer with `gpcm_boundary`; keep FACETS-style score-side review and design forecasting on their separate capability rows.

If you are unsure which helper to call, start with `mfrmr_output_guide()`. It returns a compact purpose-to-helper table that separates `*_table`, `*_report`, `*_review`, `*_bundle`, `export_*`, and compatibility routes.

Which output answers which question

`specifications_report()` Documents model type, estimation method, anchors, and core run settings. Best for method sections and reproducibility records.

`data_quality_report()` Summarizes retained and dropped rows, missingness, and unknown elements. Best for data cleaning narratives.

`estimation_iteration_report()` Shows replayed convergence trajectories. Best for diagnosing slow or unstable estimation.

`precision_review_report()` Summarizes whether `SE`, `CI`, and reliability indices are model-based, hybrid, or exploratory. Best for deciding how strongly to phrase inferential claims.

`facet_statistics_report()` Bundles facet summaries, precision summaries, and variability tests. Best for facet-level reporting.

`subset_connectivity_report()` Summarizes disconnected subsets and coverage bottlenecks. Best for linking and anchor strategy review.

`rating_scale_table()` Gives category counts, average measures, and threshold diagnostics. Best for first-pass category evaluation.

`category_structure_report()` Adds transition points and compact category warnings. Best for category-order interpretation.

`category_curves_report()` Returns category-probability, cumulative-probability, expected-ogive, total-information, and category-specific information coordinates. Best for downstream graphics and report drafts.

`write_mfrm_residual_file()` Writes an observation-level residual file, optionally with modeled category probabilities. Best for external case review or reproducible handoff.

`write_mfrm_subset_file()` Writes connected-subset summary and node-membership files. Best for scale-linking review outside R.

`reporting_checklist()` Turns analysis status into an action list with priorities and next steps. Best for closing reporting gaps.

`build_apa_outputs()` Creates manuscript-draft text, notes, captions, and section maps from a shared reporting contract.

`build_summary_table_bundle()` Converts supported `summary()` outputs into named `data.frame` tables with a compact index for appendix or manuscript handoff, including recovery simulation and recovery assessment outputs. It also supports bundle-level `summary()` / `plot()` for QC before export.

`export_summary_appendix()` Exports those validated summary-table bundles as CSV and optional HTML appendix artifacts without requiring the broader fit-based export bundle. This is the preferred export route for recovery simulation evidence.

`apa_table()` Can now take those summary-table bundles directly, so a selected component can move from `summary()` to a formatted handoff table without rebuilding the analysis object path.

Practical interpretation rules

- Use bundle summaries first, then drill down into component tables.
- Treat `precision_review_report()` as the gatekeeper for formal inference.
- Treat category and bias outputs as complementary layers rather than substitutes for overall fit review.
- Treat zero-count score categories as scale-functioning caveats. Boundary zero-count categories can be retained with explicit `rating_min` / `rating_max`; intermediate zero-count categories require `keep_original = TRUE` and make adjacent thresholds weakly identified. `summary(describe_mfrm_data(...))` exposes these in `Notes`, printed `Caveats`, and `$caveats`; `summary(fit)` carries full structured caveats into printed `Caveats` and `$caveats`, with `Key warnings` as a short triage subset. Summary-table exports use `score_category_caveats` and `analysis_caveats`.
- Use `reporting_checklist()` before `build_apa_outputs()` when a report still needs missing diagnostics or clearer caveats.

Typical workflow

- Run documentation: fit_mfrm() -> specifications_report() -> data_quality_report().
- Precision and facet review: diagnose_mfrm() -> precision_review_report() -> facet_statistics_report().
- Scale review: rating_scale_table() -> category_structure_report() -> category_curves_report().
- Manuscript handoff (RSM / PCM): reporting_checklist() -> build_apa_outputs() -> build_summary_table_bund -> summary() / plot() -> apa_table() or export_summary_appendix() / export_mfrm_bundle()(include = "summary_tables").
- Bounded GPCM handoff: reporting_checklist() -> direct summaries/plots -> build_apa_outputs() or build_summary_table_bundle() -> export_summary_appendix() or caveated export_mfrm_bundle(), with gpcm_boundary retained in report/export objects.
- Recovery simulation handoff: evaluate_mfrm_recovery() -> plot() / assess_mfrm_recovery() -> build_summary_table_bundle() -> export_summary_appendix().

Companion guides

- For visual follow-up, see mfrmr_visual_diagnostics.
- For one-shot analysis routes, see mfrmr_workflow_methods.
- For manuscript assembly, see mfrmr_reporting_and_apa.
- For linking and DFF review, see mfrmr_linking_and_dff.
- For legacy-compatible wrappers and exports, see mfrmr_compatibility_layer.

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
fit <- fit_mfrm(
  toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")

spec <- specifications_report(fit)
summary(spec)$overview

prec <- precision_review_report(fit, diagnostics = diag)
summary(prec)$checks

checklist <- reporting_checklist(fit, diagnostics = diag)
subset(checklist$checklist, Section == "Visual Displays", c("Item", "NextAction"))

apa <- build_apa_outputs(fit, diagnostics = diag)
apa$section_map[, c("Heading", "Available")]
bundle <- build_summary_table_bundle(checklist)
```

```
bundle$table_index
```

`mfrmr_visual_diagnostics` *mfrmr Visual Diagnostics Map*

Description

Quick guide to choosing the right base-R diagnostic plot in `mfrmr`. Use this page when you know the analysis question but do not yet know which plotting helper or `plot()` method to call.

If you are preparing figures for a report, start with `reporting_checklist()` and inspect the `"Visual Displays"` rows first. Those rows now map directly onto the public plotting family covered on this page, so the checklist can act as a plot-readiness router rather than just a manuscript checklist.

This guide is primarily written for diagnostics-based `RSM` / `PCM` workflows. `GPCM` fits also use the residual-based diagnostics stack through `diagnose_mfrm()`, `plot_unexpected()`, `plot_displacement()`, `plot_interrater_agreement()`, `plot_facets_chisq()`, `plot_residual_pca()`, and `plot_qc_dashboard()`, plus the posterior-scoring, design-weighted-information path via `compute_information()` / `plot_information()`, and the Wright / pathway / CCC fit plots. Two `GPCM`-specific caveats apply when interpreting these residual-based screens:

- The free discrimination parameter means MnSq mean-square screens carry weaker invariance evidence than they do under `RSM` / `PCM`. Treat MnSq flags from `GPCM` as exploratory pointers to cells that merit closer inspection rather than as Rasch-style violations of strict invariance.
- FACETS-style fair averages are a Rasch-family measure-to-score transformation. Under `GPCM` the fair-average panel of `plot_qc_dashboard()` therefore renders with an explicit "unavailable" status, and the broader compatibility-export helpers stay outside the validated `GPCM` boundary.

Use `gpcm_capability_matrix()` for the formal per-helper boundary before choosing a `GPCM` follow-up plot route.

Start with the question

- "Do persons and facet levels overlap on the same logit scale?" Use `plot(fit, type = "wright")` or `plot_wright_unified()`.
- "Where do score categories transition across theta?" Use `plot(fit, type = "pathway")` and `plot(fit, type = "ccc")`.
- "Is the design linked well enough across subsets or administrations?" Use `plot(subset_connectivity_report(...), type = "design_matrix")`, `mfrm_network_analysis()`, `build_mfrm_network_review()`, `plot(..., type = "network")`, and `plot_anchor_drift()`.
- "Which responses or levels look locally problematic?" Use `plot_unexpected()` and `plot_displacement()`.
- "Which facet/category cells drive strict marginal misfit?" Use `plot_marginal_fit()`.
- "Which level pairs drive strict local-dependence follow-up?" Use `plot_marginal_pairwise()`.

- "Do raters agree and do facets separate meaningfully?" Use `plot_interrater_agreement()`, `rater_network_analysis()`, and `plot_facets_chisq()`.
- "Do criteria within the same rater move together in a halo-like way?" Use `rater_halo_network_analysis()` and `plot(..., type = "edge_distribution")`.
- "Is there notable residual structure after the main Rasch dimension?" Use `plot_residual_pca()`.
- "Which interaction cells or facet levels drive bias screening results?" Use `plot_bias_interaction()`.
- "Which group-by-facet contrasts drive DFF / DIF screening results?" Use `plot_dif_heatmap()` and `plot_dif_summary()` after `analyze_dff()`.
- "Do person response rows follow the expected Guttman-style ordering once persons and items are sorted on the logit scale?" Use `plot_guttman_scalogram()` as a teaching-oriented screen.
- "Do person-level standardized residuals look Gaussian, or are there heavy tails that warrant follow-up?" Use `plot_residual_qq()`.
- "Is rater severity drifting across waves or training sessions (assuming the waves are already on a common anchored scale)?" Use `plot_rater_trajectory()` together with `plot_anchor_drift()` for the linking-scale review.
- "I have many raters and want a compact pairwise agreement / correlation overview instead of the bar chart?" Use `plot_rater_agreement_heatmap()`.
- "Do response times suggest rapid responding, slow responding, or timing patterns by person, facet, or score category?" Use `response_time_review()` and `plot_response_time_review()` as a descriptive QC layer outside the MFRM likelihood.
- "Are there pairs of facet levels whose residuals co-move beyond the main-effects MFRM? (Q3-style local-dependence screen)" Use `plot_local_dependence_heatmap()`.
- "How distinguishable is each facet on a single page (separation, strata, reliability)?" Use `plot_reliability_snapshot()`.
- "Where do persons with the largest residual aggregates accumulate across facet levels?" Use `plot_residual_matrix()`.
- "How much did empirical-Bayes shrinkage move each facet level?" Use `plot_shrinkage_funnel()` on a fit augmented via `apply_empirical_bayes_shrinkage()`.
- "I need one compact triage screen first." Use `plot_qc_dashboard()` for `RSM` / `PCM`. The bounded `GPCM` branch can also call `plot_qc_dashboard()`, but its fair-average panel reports an explicit unavailability indicator because that panel's score-metric semantics have not yet been generalized beyond the Rasch-family branch.
- "Which figures are already supported by my current run?" Use `reporting_checklist()` and review the `"Visual Displays"` rows before choosing the next plot.
- "Where should this figure go in a paper or appendix?" Use `visual_reporting_template()` for a static reporting-use table, then cross-check run-specific availability with `reporting_checklist()$visual_scope`
- "Do I need a 3D-style category probability surface?" Use `plot(fit, type = "ccc_surface", draw = FALSE)` to get theta-by-category-by-probability plot data for exploratory teaching or downstream interactive rendering. Keep 2D pathway/CCC plots as the default reporting figures.

Recommended visual route

1. If you are drafting a report, run `reporting_checklist()` first and read the `"Visual Displays"` rows as the plot-readiness layer.
2. Start with `plot_qc_dashboard()` for one-page triage.
3. Move to `plot_unexpected()`, `plot_displacement()`, `plot_marginal_fit()`, `plot_marginal_pairwise()`, and `plot_interrater_agreement()` for flagged local issues.
4. Use `plot(fit, type = "wright")`, `plot(fit, type = "pathway")`, and `plot_residual_pca()` for structural interpretation.
5. Use `plot_bias_interaction()`, `plot_dif_heatmap()`, `plot_dif_summary()`, `plot_anchor_drift()`, and `plot_information()` when the checklist or dashboard points to interaction, differential-functioning, linking, or precision follow-up.
6. Use `plot(..., draw = FALSE)` when you want reusable plot data instead of immediate graphics.
7. Use `plot(fit, type = "ccc_surface", draw = FALSE)` only when you need 3D-ready category-probability data; `mfrmr` intentionally does not add a package-native plotly/rgl renderer for this route.
8. Use `preset = "publication"` when you want the package's cleaner manuscript-oriented styling.

Visual coverage for this release

This release treats the plotting layer as sufficient when the current run supports all of the following follow-up roles through public helpers:

- First-pass triage: `plot_qc_dashboard()` or the `"Visual Displays"` rows from `reporting_checklist()`.
- Structural interpretation: `plot(fit, type = "wright")`, `plot(fit, type = "pathway")`, `plot(fit, type = "ccc")`, and `plot_residual_pca()`.
- Local issue follow-up: `plot_unexpected()`, `plot_displacement()`, `plot_interrater_agreement()`, `plot_bias_interaction()`, `plot_dif_heatmap()`, and `plot_dif_summary()`.
- Strict marginal follow-up: `plot_marginal_fit()` and `plot_marginal_pairwise()` for `diagnostic_mode = "both"`.
- Reporting/export handoff: `build_visual_summaries()` and `draw = FALSE` routes that return reusable `mfrm_plot_data` objects for downstream review and export. When step estimates are available, `build_visual_summaries()` also exposes `$plot_payloads$category_probability_surface`.
- 3D-ready exploratory handoff: `plot(fit, type = "ccc_surface", draw = FALSE)` returns a theta-by-category-by-probability `mfrm_plot_data` object. This is not a default APA/reporting figure and does not load plotly/rgl.

3D and surface data

The package currently treats 3D as an exploratory data handoff, not as a default plotting layer. The supported route is `plot(fit, type = "ccc_surface", draw = FALSE)`, which returns `surface`, `categories`, `category_support`, `groups`, `axis_contract`, `renderer_contract`, `interpretation_guide`, and `reporting_policy` tables inside an `mfrm_plot_data` object. These columns can be passed to an external renderer if needed, while `category_support` and `interpretation_guide` should be checked before interpreting retained zero-frequency categories or adjacent threshold ridges.

Do not replace the standard 2D Wright map, pathway map, CCC plot, heatmap/profile diagnostics, or information curves with 3D figures in routine reports. In particular, 3D Wright maps are discouraged because perspective and occlusion obscure the shared-scale comparison that the Wright map is meant to support.

Which plot answers which question

`plot(fit, type = "wright")` Shared logit map of persons, facet levels, and step thresholds. Best for targeting and spread.

`plot(fit, type = "pathway")` Expected score by theta, with dominant-category strips. Best for scale progression.

`plot(fit, type = "ccc")` Category probability curves. Best for checking whether categories peak in sequence.

`plot_unexpected()` Observation-level surprises. Best for case review and local misfit triage.

`plot_displacement()` Level-wise anchor movement. Best for anchor robustness and residual calibration tension.

`plot_marginal_fit()` Posterior-integrated first-order category residuals. Best for seeing which facet/category cells drive strict marginal flags.

`plot_marginal_pairwise()` Posterior-integrated exact/adjacent agreement residuals. Best for exploratory local-dependence follow-up after strict marginal flags.

`plot_interrater_agreement()` Exact agreement, expected agreement, pairwise correlation, and agreement gaps. Best for rater consistency.

`plot_facets_chisq()` Facet variability and chi-square summaries. Best for checking whether a facet contributes meaningful spread.

`plot_residual_pca()` Residual structure after the Rasch dimension is removed. Best for exploratory residual-structure review, not as a standalone unidimensionality test.

`plot_bias_interaction()` Interaction-bias screening views for cells and facet profiles. Best for systematic departure from the additive main-effects model.

`plot_dif_heatmap()` / `plot_dif_summary()` DFF / DIF screening views for facet-level x group contrasts. Best for showing which facet and group pair is involved before writing substantive interpretations.

`plot_anchor_drift()` Anchor drift and screened linking-chain visuals. Best for multi-form or multi-wave linking review after checking retained common-element support.

`plot_guttman_scalogram()` Person x facet-level response matrix with unexpected-response overlay. Best for teaching-oriented scalogram intuition and visual triage of where the data depart from the expected ordering.

`plot_residual_qq()` Normal Q-Q plot of person-level standardized residual aggregates. Best for checking the tail behavior of residuals as exploratory follow-up after a fit screen.

`plot_rater_trajectory()` Per-rater severity trajectory across named waves / occasions. Best for rater-training or drift feedback when the supplied fits have already been placed on a common anchored scale; the helper itself does not perform linking.

`plot_rater_agreement_heatmap()` Compact pairwise rater x rater heatmap of exact agreement (default) or Pearson-style correlation. Best when the rater count makes the bar-chart form of `plot_interrater_agreement()` too busy.

`response_time_review()` / `plot_response_time_review()` Descriptive response-time screening by person, facet, and score category. Best for reviewing rapid/slow response patterns alongside MFRM diagnostics; it is not a joint speed-accuracy model and does not change fitted measures.

`plot_local_dependence_heatmap()` Yen Q3-style heatmap of pairwise residual correlations between facet levels. Best for exploratory local-dependence screening; pairs with very strong off-diagonal residual correlation merit content-level review.

`plot_reliability_snapshot()` One-figure facet x reliability / separation / strata bar overview built from `diagnostics$reliability`. Best as a single small figure for "which facets are statistically distinguishable?".

`plot_residual_matrix()` Person x facet-level standardized residual heatmap. Best as a follow-up to `plot_guttman_scalogram()` when the residual sign and magnitude matter, not just the response code.

`plot_shrinkage_funnel()` Empirical-Bayes shrinkage caterpillar / funnel showing raw versus shrunken facet estimates. Best on fits produced via `apply_empirical_bayes_shrinkage()` for reviewing how much each level moved under the prior.

Cross-reference to FACETS / Winsteps tables

For users coming from the standard Rasch-measurement software packages, the closest mfrmr helper for each table or figure family is summarised below. The mapping is approximate; mfrmr is designed for many-facet workflows, so column subsets and column names differ.

Wright (variable) map `plot(fit, type = "wright")` and `plot_wright_unified()` correspond to FACETS Table 6 / Winsteps "Person-Item map".

Pathway / probability curves `plot(fit, type = "pathway")` and `plot(fit, type = "ccc")` correspond to Winsteps Table 21 ("Probability category curves") and FACETS category-probability curves.

Test / item information `compute_information()` + `plot_information()` correspond to Winsteps Table 17 ("Test characteristic curve, test information function").

Misfit / Infit / Outfit `diagnose_mfrm()` and the Largest |ZSTD| / MnSq misfit blocks of `summary(diag)` correspond to Winsteps Table 10/13/14 (Misfit order) and FACETS Tables 7/8.

Bias / interaction `estimate_bias()` + `plot_bias_interaction()` correspond to FACETS Table 14 ("Bias / Interaction calibration report").

Differential rater / item functioning `analyze_dff()` / `analyze_dif()` + `plot_dif_heatmap()` / `plot_dif_summary()` cover the FACETS DIF / bias-by-group route and the Winsteps DIF (Table 30 group differences) report.

Inter-rater agreement `interrater_agreement_table()` + `plot_interrater_agreement()` / `plot_rater_agreement_`
correspond to FACETS Table 7-style observed-vs-expected agreement reports.

Anchoring / linking `plot_anchor_drift()` and `plot_information()` cover the FACETS / Winsteps anchored-run review route; full equating-chain helpers are exposed via `build_equating_chain()`.

Practical interpretation rules

- Wright map: look for gaps between person density and facet/step locations; large gaps indicate weaker targeting.

- Pathway / CCC: look for monotone progression and clear category dominance bands; flat or overlapping curves suggest weak category separation.
- 3D-ready category surface: use as an exploratory view of the same category-probability information, not as a replacement for the 2D pathway/CCC figures in reports. Read `category_support` first when a retained category has zero observed responses.
- Unexpected / displacement: use as screening tools, not final evidence by themselves.
- Strict marginal and pairwise local-dependence plots are exploratory follow-up layers for `diagnostic_mode = "both"`, not standalone inferential tests.
- Inter-rater agreement and facet variability address different questions: agreement concerns scoring consistency, whereas variability concerns whether facet elements are statistically distinguishable.
- Residual PCA and bias plots should be interpreted as follow-up layers after the main fit screen, not as first-pass diagnostics.
- DFF residual-method plots are screening visuals. ETS A/B/C labels should be claimed only for rows whose refit output reports `ClassificationSystem == "ETS"`.

Typical workflow

- Figure-readiness route: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()` -> inspect `"Visual Displays"` rows -> chosen public plot helper.
- Quick screening: `fit_mfrm()` -> `diagnose_mfrm()` -> `plot_qc_dashboard()`.
- Strict marginal follow-up: `diagnose_mfrm()` with `diagnostic_mode = "both"` -> `plot_marginal_fit()` -> `plot_marginal_pairwise()`.
- Scale and targeting review: `plot(fit, type = "wright")` -> `plot(fit, type = "pathway")` -> `plot(fit, type = "ccc")`.
- Linking review: `subset_connectivity_report()` -> `plot(..., type = "design_matrix")` / `mfrm_network_analysis()` / `build_mfrm_network_review()` / `plot(..., type = "network")` -> `plot_anchor_drift()`.
- Interaction review: `estimate_bias()` -> `plot_bias_interaction()` -> `reporting_checklist()`.
- DFF / DIF review: `analyze_dff()` -> `plot_dif_heatmap()` / `plot_dif_summary()` -> inspect the explicit facet, level, and group-pair columns before writing interpretation.

Companion vignette

For a longer, plot-first walkthrough, run `vignette("mfrmr-visual-diagnostics", package = "mfrmr")`.

See Also

mfrmr_workflow_methods, mfrmr_reports_and_tables, mfrmr_reporting_and_apa, mfrmr_linking_and_dff, gpcm_capability_matrix, `visual_reporting_template()`, `mfrmr_interval_guide()`, `plot.mfrm_fit()`, `plot_qc_dashboard()`, `plot_unexpected()`, `plot_displacement()`, `plot_marginal_fit()`, `plot_marginal_pairwise`
`plot_interrater_agreement()`, `plot_facets_chisq()`, `plot_residual_pca()`, `plot_bias_interaction()`, `plot_dif_heatmap()`, `plot_dif_summary()`, `plot_anchor_drift()`, `plot_guttman_scalogram()`, `plot_residual_qq()`, `plot_rater_trajectory()`, `plot_rater_agreement_heatmap()`, `response_time_review()`, `plot_response_time_review()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
checklist <- reporting_checklist(fit, diagnostics = diag)
visual_reporting_template("manuscript")
subset(
  checklist$checklist,
 Section == "Visual Displays" & Item %in% c("QC / facet dashboard", "Strict marginal visuals"),
  c("Item", "Available", "NextAction")
)

qc <- plot_qc_dashboard(fit, diagnostics = diag, draw = FALSE, preset = "publication")
qc$data$plot

p_marg <- plot_marginal_fit(diag, draw = FALSE, preset = "publication")
p_marg$data$preset

wright <- plot(fit, type = "wright", draw = FALSE, preset = "publication")
wright$data$preset

pca <- analyze_residual_pca(diag, mode = "overall")
scree <- plot_residual_pca(pca, plot_type = "scree", draw = FALSE, preset = "publication")
scree$data$preset
```

---

mfrmr_workflow_methods

mfrmr Workflow and Method Map

Description

Quick reference for end-to-end mfrmr analysis and for checking which output objects support summary() and plot().

Canonical reporting route

For the clearest default route in RSM / PCM, use fit_mfrm() with method = "MML" -> diagnose_mfrm() with diagnostic_mode = "both" -> reporting_checklist() -> plot_qc_dashboard() and, when

flagged, plot_marginal_fit() / plot_marginal_pairwise() -> build_apa_outputs() -> build_summary_table_bund
-> apa_table() or export_summary_appendix().

Use JML only when you explicitly want a faster exploratory pass and are willing to defer strict marginal follow-up and formal precision language to a later MML run.

Canonical operational review route

When the main question is scale maintenance rather than manuscript reporting, branch after diagnose_mfrm() into: review_mfrm_anchors() and/or detect_anchor_drift() -> build_equating_chain() when adjacent-link review is needed -> build_linking_review() -> inspect review$group_view_index for stable wave / link / facet rollups and summary(review)$plot_routes for the next plot helper -> plot_anchor_drift() or plot(anchor_review, ...) for the specific flagged evidence family.

For bounded GPCM, use build_linking_review() as a caveated exploratory synthesis over direct anchor, drift, and chain evidence. It is not an operational GPCM linking decision or evidence that anchor drift is absent.

Canonical misfit case-review route

When the main question is which observations, facet levels, or pairwise structures deserve follow-up, branch after diagnose_mfrm() into: build_misfit_casebook() -> inspect casebook$group_view_index, casebook$group_views, and summary(casebook)$plot_routes for stable person / facet / wave rollups and the next plot helper -> plot_unexpected(), plot_displacement(), plot_marginal_fit(), or plot_marginal_pairwise() according to casebook$plot_map -> build_summary_table_bundle() / export_summary_appendix() when the flagged cases need appendix-style reporting support.

build_misfit_casebook() can still be used for bounded GPCM, but it should be read as an operational exploratory screen rather than as a strict Rasch-style invariance report.

Latent-regression route

When the fit uses population_formula = ..., keep the distinction between the estimator and the forecast helpers explicit:

- fit_mfrm() estimates the current narrow latent-regression MML branch. In the returned fit object, fit$population$person_table is the complete-case estimation table, while fit$population$person_table_r retains the observed-person-aligned pre-omit background-data table for replay/export provenance.
- predict_mfrm_units() and sample_mfrm_plausible_values() can then score under the fitted population model when scored units also supply one-row-per-person background data. That scoring-time person_data contract remains separate from the fit object's stored replay table.
- predict_mfrm_population() remains a scenario-level simulation/refit helper rather than the latent-regression estimator itself.

Score-category support

If the intended rating scale includes categories not observed in the current data, make that support explicit. For example, use rating_min = 1, rating_max = 5 for a 1-5 scale with only 2-5 observed. If an intermediate category is unobserved (for example 1, 2, 4, 5 with no 3),

also set keep_original = TRUE if the zero-count category should remain in the fitted support. summary(describe_mfrm_data(...)) reports retained zero-count categories in Notes, printed Caveats, and $caveats; summary(fit) carries full structured rows into printed Caveats and $caveats, with Key warnings as a short triage subset. Summary-table exports route those rows through score_category_caveats or analysis_caveats. Adjacent threshold estimates should still be treated as weakly identified when an intermediate category is unobserved.

Typical workflow

1. Fit a model with fit_mfrm(). For final reporting, prefer method = "MML" unless you explicitly want a fast exploratory JML pass.
2. (Optional) Use run_mfrm_facets() or mfrmRFacets() for a legacy-compatible one-shot workflow wrapper.
3. For RSM / PCM, build diagnostics with diagnose_mfrm(). For final reporting, prefer diagnostic_mode = "both" so the legacy residual path and the strict marginal screen remain visible side by side. For bounded GPCM, diagnostics are now available through diagnose_mfrm() together with analyze_residual_pca(), interrater_agreement_table(), unexpected_response_table(), displacement_table(), measurable_summary_table(), rating_scale_table(), facet_quality_dashboard(), reporting_checklist(), and plot_qc_dashboard() – the fair-average panel of the dashboard reports an explicit unavailability indicator under GPCM. Use fair_average_table() directly when you need the supported slope-aware element-conditional fair averages. Treat those residual-based summaries as exploratory screens because the discrimination parameter is free. Full FACETS-style score-side contract review remains blocked for bounded GPCM; package-native scorefile export, fit-based reporting bundles, direct fair-average tables, and bias-screening tables carry their own caveats. Posterior scoring with predict_mfrm_units() / sample_mfrm_plausible_values(), design-weighted information via compute_information() / plot_information(), Wright/pathway/CCC plots via plot.mfrm_fit(), direct category reports via category_structure_report() / category_curves_report(), and direct data generation through build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data() are also available when the simulation specification stores both thresholds and slopes. Use evaluate_mfrm_recovery() and assess_mfrm_recovery() for direct recovery checks plus caveated role-based design evaluation, population forecasting, diagnostic-screening, and signal-detection helpers. Caveated APA/QC/export bundles are available for sensitivity reporting, while score-side FACETS helpers remain outside the validated GPCM boundary. Use gpcm_capability_matrix() as the formal capability map before branching into less common helpers.
4. (Optional, RSM / PCM; bounded GPCM with caveat) Estimate interaction bias with estimate_bias().
5. Choose a downstream branch: reporting_checklist() for direct report preparation, or build_weighting_review() for Rasch-versus-bounded-GPCM weighting review, or build_misfit_casebook() / build_linking_review() for operational case review. For bounded GPCM, use build_linking_review() only as an exploratory index over direct anchor/drift/chain evidence.
6. Generate reporting bundles: build_summary_table_bundle(), apa_table(), export_summary_appendix(), build_fixed_reports(), build_visual_summaries(). For bounded GPCM, use the APA, visual, QC, and fit-based export bundles as caveated sensitivity-reporting surfaces; full score-side FACETS review stays blocked, while diagnostic/signal-detection design screening has its own caveated operating-characteristic route.
7. (Optional, RSM / PCM) Review report completeness with reference_case_review(). Use facets_output_contract_review() only when you explicitly need the compatibility layer.

8. (Optional, RSM / PCM) For operational linking follow-up, combine review_mfrm_anchors(), detect_anchor_drift(), and build_equating_chain() inside build_linking_review() before exporting appendix-style tables.
9. (Optional) Check packaged reference cases with reference_case_benchmark() when you want package-side reference checks.
10. (Optional) For design planning or future scoring, move to the simulation/prediction layer: build_mfrm_sim_spec() / extract_mfrm_sim_spec() -> evaluate_mfrm_recovery() -> assess_mfrm_recovery() / evaluate_mfrm_design() / predict_mfrm_population() -> predict_mfrm_units() / sample_mfrm_plausible_values(). Current fit-derived simulation specs include direct GPCM data generation and recovery checks. Design-evaluation, population-forecasting, diagnostic- screening, and signal-detection helpers also support bounded GPCM as caveated role-based simulation/refit evidence; inspect gpcm_boundary before using those results in design claims. Unit scoring can use an ordinary MML fit directly, a latent-regression MML fit when you also supply one-row-per-person background data for the scored units, or a JML fit when a post hoc reference-prior EAP layer is acceptable. Intercept-only latent-regression fits (population_formula = ~ 1) can reconstruct that minimal person table from the scored person IDs. Keep predict_mfrm_population() conceptually separate from that scoring layer: it is a simulation-based scenario forecast helper, not the latent-regression estimator itself. Prediction export still requires actual prediction objects in addition to include = "predictions".
11. Use summary() for compact text checks and plot() (or dedicated plot helpers) for base-R visual diagnostics.

Three practical routes

- Quick first pass: RSM / PCM: fit_mfrm() -> diagnose_mfrm() -> plot_qc_dashboard() -> reporting_checklist() when you want the package to route the next figures. bounded GPCM: fit_mfrm() -> diagnose_mfrm() -> plot_qc_dashboard() / unexpected_response_table() -> rating_scale_table() -> compute_information() -> plot_information() -> plot.mfrm_fit() / category_curves_report() -> fair_average_table() / estimate_bias() when those screening tables answer the question. For bounded GPCM, the fit-based export family (build_mfrm_manifest(), build_mfrm_replay_script(), export_mfrm_bundle()) is available as caveated sensitivity-reporting output with explicit gpcm_boundary rows.
- Linking and coverage review: subset_connectivity_report() -> plot(..., type = "design_matrix") -> plot_wright_unified().
- Manuscript prep: RSM / PCM: reporting_checklist() -> inspect the "Visual Displays" and "Method Section" rows -> build_apa_outputs() -> build_summary_table_bundle() -> apa_table() or export_summary_appendix(). First-release GPCM: reporting_checklist() -> direct table/plot helpers -> build_apa_outputs() / build_visual_summaries() -> export_mfrm_bundle() with gpcm_boundary caveats.
- Weighting-policy review: compare_mfrm() -> build_weighting_review() -> compute_information() / plot_information() when you want to inspect whether bounded GPCM is introducing substantively acceptable discrimination-based reweighting relative to the Rasch-family reference.
- Design planning and forecasting: build_mfrm_sim_spec() or extract_mfrm_sim_spec() -> evaluate_mfrm_recovery() -> assess_mfrm_recovery() for parameter-recovery checks, then evaluate_mfrm_design() -> predict_mfrm_population() -> predict_mfrm_units()

or sample_mfrm_plausible_values() under the fitted scoring basis (ordinary MML, latent-regression MML with person-level background data, or JML with the documented post hoc EAP approximation). Here again, predict_mfrm_population() is the scenario-level forecast helper, whereas predict_mfrm_units() / sample_mfrm_plausible_values() are the scoring layer. Prediction export requires actual prediction objects. First-release GPCM now supports direct data generation via build_mfrm_sim_spec(), extract_mfrm_sim_spec(), and simulate_mfrm_data(), evaluate_mfrm_recovery(), assess_mfrm_recovery(), caveated role-based design evaluation and population forecasting, diagnostic/signal-detection design screening, residual diagnostics, and direct curve/report helpers. The current planning layer remains role-based for two non-person facets even though estimation itself supports arbitrary facet counts; future arbitrary-facet planning fields should be treated as design metadata rather than finished public behavior.

Interpreting output

This help page is a map, not an estimator:

- use it to decide function order,
- confirm which objects have summary()/plot() defaults,
- identify when dedicated helper functions are needed,
- and treat reporting_checklist() as the package's readiness router for plot and report follow-up.

Objects with default summary() and plot() routes

- mfrm_fit: summary(fit) and plot(fit, ...).
- mfrm_diagnostics: summary(diag); plotting via dedicated helpers such as plot_unexpected(), plot_displacement(), plot_qc_dashboard().
- mfrm_bias: summary(bias) and plot_bias_interaction().
- mfrm_data_description: summary(ds) and plot(ds, ...).
- mfrm_anchor_review: summary(review) and plot(review, ...).
- mfrm_misfit_casebook: summary(casebook) and print(casebook), with grouping views available through casebook$group_view_index and casebook$group_views, source-specific plotting routed through summary(casebook)$plot_routes and casebook$plot_map, and appendix/report handoff available through build_summary_table_bundle() and export_summary_appendix().
- mfrm_weighting_review: summary(review) and print(review), with information follow-up routed through compute_information() and plot_information() according to review$plot_map, and appendix/report handoff available through build_summary_table_bundle() and export_summary_appendix().
- mfrm_linking_review: summary(review) and print(review), with grouping views available through review$group_view_index and review$group_views, and plotting routed through summary(review)$plot_routes, plot_anchor_drift(), and plot(anchor_review, ...) according to review$plot_map.
- mfrm_facets_run: summary(run) and plot(run, type = c("fit", "qc"), ...).
- apa_table: summary(tbl) and plot(tbl, ...).
- mfrm_apa_outputs: summary(apa) for compact diagnostics of report text.

- mfrm_summary_table_bundle: print(bundle) for manuscript-oriented table index plus named tables from supported summary() outputs, summary(bundle) for table-role/numeric coverage, and plot(bundle, ...) for table-size or numeric-column QC.
- mfrm_threshold_profiles: summary(profiles) for preset threshold grids.
- mfrm_population_prediction: summary(pred) for design-level forecast tables.
- mfrm_unit_prediction: summary(pred) for unit-level posterior summaries under the fitted scoring basis.
- mfrm_plausible_values: summary(pv) for draw-level uncertainty summaries.
- mfrm_bundle families: summary() and class-aware plot(bundle, ...). Key bundle classes now also use class-aware summary(bundle): mfrm_unexpected, mfrm_fair_average, mfrm_displacement, mfrm_interrater, mfrm_facets_chisq, mfrm_bias_interaction, mfrm_rating_scale, mfrm_category_structure, mfrm_category_curves, mfrm_measurable, mfrm_unexpected_after_bias, mfrm_output_bundle, mfrm_residual_pca, mfrm_specifications, mfrm_data_quality, mfrm_iteration_report, mfrm_subset_connectivity, mfrm_facet_statistics, mfrm_facets_contract_revie mfrm_reference_review, mfrm_reference_benchmark.

plot.mfrm_bundle() **coverage**

Default dispatch now covers:

- mfrm_unexpected, mfrm_fair_average, mfrm_displacement
- mfrm_interrater, mfrm_facets_chisq, mfrm_bias_interaction
- mfrm_bias_count, mfrm_fixed_reports, mfrm_visual_summaries
- mfrm_category_structure, mfrm_category_curves, mfrm_rating_scale
- mfrm_measurable, mfrm_unexpected_after_bias, mfrm_output_bundle
- mfrm_residual_pca, mfrm_specifications, mfrm_data_quality
- mfrm_iteration_report, mfrm_subset_connectivity, mfrm_facet_statistics
- mfrm_facets_contract_review, mfrm_reference_review, mfrm_reference_benchmark

For unknown bundle classes, use dedicated plotting helpers or custom base-R plots from component tables.

See Also

fit_mfrm(), run_mfrm_facets(), mfrmRFacets(), diagnose_mfrm(), estimate_bias(), mfrmr_visual_diagnostics, mfrmr_reports_and_tables, mfrmr_reporting_and_apa, gpcm_capability_matrix, mfrmr_linking_and_dff, mfrmr_compatibility_layer, summary.mfrm_fit(), summary(diag), summary(), plot.mfrm_fit(), plot()

Examples

```
toy_full <- load_mfrmr_data("example_core")
keep_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% keep_people, , drop = FALSE]

fit <- fit_mfrm(
  toy,
```

```
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
)
summary(fit)$next_actions

diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
summary(diag)$next_actions

chk <- reporting_checklist(fit, diagnostics = diag)
subset(
  chk$checklist,
  Section == "Visual Displays",
  c("Item", "DraftReady", "NextAction")
)

qc <- plot_qc_dashboard(fit, diagnostics = diag, draw = FALSE, preset = "publication")
qc$data$preset
p_marg <- plot_marginal_fit(diag, draw = FALSE, preset = "publication")
p_marg$data$preset

sc <- subset_connectivity_report(fit, diagnostics = diag)
p_design <- plot(sc, type = "design_matrix", draw = FALSE, preset = "publication")
p_design$data$plot

bundle <- build_summary_table_bundle(chk, appendix_preset = "recommended")
summary(bundle)$role_summary
plot(bundle, type = "appendix_presets", draw = FALSE)$data$plot
```

`mfrm_d_study` *Project G-theory coefficients under alternative D-study designs*

Description

`mfrm_d_study()` applies a practical D-study projection to the variance components from `mfrm_generalizability()`. It answers questions such as "what happens to `G` and `Phi` if we use 2, 3, or 4 raters?" without re-fitting the Rasch/MFRM model.

Usage

```
mfrm_d_study(
  x,
  design_grid = NULL,
  object_facet = "Person",
  random_facets = NULL,
```

```
  residual_scaling = c("highest_order", "single_condition", "none", "sensitivity"),
  ...
)
```

Arguments

x	Output from mfrm_generalizability() or an mfrm_fit. If an mfrm_fit is supplied, mfrm_generalizability() is called first.
design_grid	Data frame or named list giving planned counts for each random measurement facet. Column names may be the facet names themselves (for example Rater) or n_ plus the facet name (for example n_Rater). When NULL, one row using the observed number of levels is returned.
object_facet, random_facets	Passed to mfrm_generalizability() when x is an mfrm_fit.
residual_scaling	How the collapsed residual variance should be scaled when planned facet counts increase. "highest_order" treats the residual as highest-order person-by-all-conditions/error variance and divides by the product of planned counts. "single_condition" divides by the smallest planned facet count, a conservative sensitivity check when unmodeled person-by-one-facet interactions may dominate. "none" leaves the residual unscaled. "sensitivity" returns all three assumptions for each design row.
...	Additional arguments passed to mfrm_generalizability() when x is an mfrm_fit.

Details

The projection uses the variance decomposition already estimated by mfrm_generalizability(). For a random measurement facet j, main-effect variance contributes sigma2_j / n_j to the absolute-error denominator. The residual term contains unmodeled person-by-facet and higher-order interaction variance in the current simplified G-study, so the selected residual_scaling assumption is reported explicitly. The relative-decision denominator uses only this scaled residual term.

This is a pragmatic D-study planning layer, not a full p x r x i ANOVA decomposition. If person-by-rater or person-by-item interactions are a primary estimand, use residual_scaling = "sensitivity" and treat the output as planning evidence; fit a fully crossed G-theory model externally when those interaction components must be estimated separately.

The G and Phi values returned here belong to the generalizability-theory metric family. They should not be interpreted as coefficient alpha, omega, KR-20, or IRT marginal/separation reliability, even though all of those summaries may be displayed on a 0–1 scale in broader reporting dashboards.

Value

An object of class mfrm_d_study, a data.frame with one row per design scenario and columns for planned facet counts, variance terms, projected G, projected Phi, and interpretation bands.

References

Cronbach, L. J., Gleser, G. C., Nanda, H., & Rajaratnam, N. (1972). *The dependability of behavioral measurements: Theory of generalizability for scores and profiles*. Wiley.

Brennan, R. L. (2001). *Generalizability theory*. Springer.

See Also

mfrm_generalizability(), evaluate_mfrm_design(), recommend_mfrm_design(), plot_data()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("lme4", quietly = TRUE)) {
  gt <- mfrm_generalizability(fit)
  mfrm_d_study(gt, data.frame(Rater = c(2, 3, 4), Criterion = 4))
}
```

---

mfrm_generalizability *Generalizability-theory variance decomposition for an MFRM design*

Description

Re-fits the rating data underlying an mfrm_fit as a crossed random-effects model Score ~ 1 + (1 | Person) + (1 | Facet1) + ... + Residual via lme4::lmer, and returns the canonical G-theory variance components plus G / Phi coefficients. Useful when reviewers ask for a generalizability-theory complement to the Rasch-style separation / reliability statistics that diagnose_mfrm() already emits.

Usage

```
mfrm_generalizability(
  fit,
  data = NULL,
  object_facet = "Person",
  random_facets = NULL,
  reml = TRUE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
data	Optional data frame. When NULL, the rating data stored on fit$prep$data is used.
object_facet	Facet that plays the role of the "object of measurement" – typically "Person" (default).
random_facets	Character vector of non-person facets to treat as random conditions of measurement. Default uses every facet other than object_facet.
reml	Logical, passed to lme4::lmer() (default TRUE).

Value

An object of class `mfrm_generalizability` with:

`variance_components` One row per random effect plus residual, with columns `Source`, `Variance`, and `ProportionVariance`.

`coefficients` One-row data frame with `G` (generalizability coefficient, relative decision) and `Phi` (dependability coefficient, absolute decision), using the single-observation-per-cell convention.

`design` Description of the crossed-random model.

Interpretation

- `G` is appropriate for **relative** decisions (rank-ordering persons): `G = sigma2(p) / (sigma2(p) + sigma2(Residual))`.
- The reported `Phi` is appropriate for **absolute** decisions (cut-score classification): `Phi = sigma2(p) / (sigma2(p) + si` before D-study scaling.
- Use `mfrm_d_study()` to project `G` / `Phi` under planned numbers of raters, items, criteria, or other random measurement facets.
- Reporting bands follow Brennan (2001): G / Phi >= 0.8 for high-stakes decisions, >= 0.7 for routine reporting.

Limitations

This helper formulates the random-effects model with main effects only (`Score ~ 1 + (1|Person) + (1|Facet1) + ... + Residual`); no explicit `(1 | Person:Rater)`, `(1 | Person:Criterion)`, or `(1 | Rater:Criterion)` interaction terms are estimated. All two-way and higher interaction variance is therefore folded into the `Residual` term – the standard one-observation-per-cell approximation – which can bias `G` downward when person x facet interactions are substantively large. This function reports the one-observation-per-cell baseline. `mfrm_d_study()` applies D-study scaling, including residual-scaling sensitivity checks, to the same simplified variance-component decomposition. Because person-by-facet interaction terms are not estimated separately, D-study projections remain practical planning evidence rather than a replacement for a fully specified G-theory design.

References

- Cronbach, L. J., Gleser, G. C., Nanda, H., & Rajaratnam, N. (1972). *The dependability of behavioral measurements: Theory of generalizability for scores and profiles*. Wiley.
- Brennan, R. L. (2001). *Generalizability theory*. Springer.

See Also

`mfrm_d_study()`, `compute_facet_icc()`, `diagnose_mfrm()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("lme4", quietly = TRUE)) {
  gt <- mfrm_generalizability(fit)
  gt$variance_components
  # Look for: a Person variance component well above any single
  #   non-person facet's variance share. Large rater or criterion
  #   variance shares mean those conditions add measurement error
  #   relative to person spread.
  gt$coefficients
  # Look for: G >= 0.7 for routine reporting, >= 0.8 for high-stakes.
  #   G < Phi means absolute decisions are noisier than relative
  #   decisions; review whether facet main effects need anchoring.
}
```

---

mfrm_misfit_thresholds

MnSq misfit threshold pair used across mfrmr screening helpers

---

Description

Returns the lower / upper bounds that mfrmr screens treat as the acceptable mean-square (Infit / Outfit MnSq) band when flagging element-level misfit. Defaults follow Linacre's published 0.5-1.5 acceptance band; both ends can be overridden via R options.

Usage

```
mfrm_misfit_thresholds(lower = NULL, upper = NULL)
```

Arguments

lower	Optional lower bound. When NULL (default), the active option / package default is used.
upper	Optional upper bound.

Details

Helpers that consume the band include summary.mfrm_diagnostics() (misfit_flagged block and key_warnings auto-flag), build_misfit_casebook() (the new element_fit source family), the bias / misfit narrative inside build_apa_outputs(), and facet_quality_dashboard() when misfit_warn = NULL. Setting the options once at the top of an analysis script therefore changes every downstream screen at once.

Value

A named numeric vector c(lower = ..., upper = ...) with lower < upper.

Configuration

Two scalar R options drive the band:

`mfrmr.misfit_lower` Lower acceptance bound. Default `0.5`.

`mfrmr.misfit_upper` Upper acceptance bound. Default `1.5`.

Pass scalar arguments to override the options for a single call, e.g. `mfrm_misfit_thresholds(lower = 0.7, upper = 1.3)` for the tighter Bond & Fox (2015) reporting band.

See Also

`summary.mfrm_diagnostics()`, `build_misfit_casebook()`, `facet_quality_dashboard()`

Examples

```
mfrm_misfit_thresholds()
old <- options(mfrmr.misfit_lower = 0.7, mfrmr.misfit_upper = 1.3)
mfrm_misfit_thresholds()
options(old)
```

---

mfrm_network_analysis *Analyze the MFRM design network*

Description

Analyze the MFRM design network

Usage

```
mfrm_network_analysis(
  fit,
  diagnostics = NULL,
  top_n_subsets = NULL,
  min_observations = 0,
  include_graph = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`top_n_subsets`	Optional maximum number of connected-subset rows to retain before constructing the graph.
`min_observations`	Minimum observations required to keep a subset row.
`include_graph`	Logical; if `TRUE`, include the underlying `igraph` object in the returned bundle. Defaults to `FALSE` so outputs remain easy to serialize.

Details

`mfrm_network_analysis()` treats the person/facet-level observation design as an undirected weighted graph. Nodes are person or facet levels; edges connect levels that co-occur in at least one observed rating; edge weights are co-observation counts. The resulting network metrics are design diagnostics, not psychometric measures of person ability or rater quality. `plot(net, type = "centrality")`, `plot(net, type = "facet_summary")`, and `plot(net, type = "network")` provide immediate visual checks; use `draw = FALSE` to extract reusable plot data.

The most useful review columns are:

- `Components`: more than one component means the design has disconnected measurement subsets.
- `IsArticulationPoint`: a node whose removal would increase disconnectedness.
- `IsBridge`: an edge whose removal would increase disconnectedness.
- `Betweenness`: a routing-dependence indicator; high values identify levels that carry many shortest paths through the design graph.

In incomplete rater-mediated designs, these graph summaries help identify fragile linking structures before interpreting facet measures or planning additional data collection.

Value

A bundle of class `mfrm_network_analysis` containing:

- `summary`: graph-level connectedness and vulnerability metrics
- `node_metrics`: node-level degree, strength, centrality, and cutpoint flags
- `edge_metrics`: edge-level weights, betweenness, and bridge flags
- `facet_summary`: facet-level aggregation of node/bridge indicators
- `cut_nodes`: articulation-point rows from `node_metrics`
- `bridge_edges`: bridge rows from `edge_metrics`

References

- McEwen, M. R. (2015). *Development of a Software Prototype for Generating and Classifying Incomplete Many-Facet-Rasch Model Rating Designs*. Brigham Young University.
- Csardi, G., Nepusz, T., Traag, V., Horvat, S., Zanini, F., Noom, D., & Muller, K. (2026). *igraph: Network Analysis and Visualization*.

See Also

`subset_connectivity_report()`, `diagnose_mfrm()`, mfrmr_linking_and_dff, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("igraph", quietly = TRUE)) {
  net <- mfrm_network_analysis(fit)
  net$summary
  head(net$node_metrics)
  net$cut_nodes
  plot(net, type = "centrality", draw = FALSE)
}
```

---

mfrm_report *Build report-ready output from* mfrm_results()

Description

mfrm_report() is a report-synthesis layer for an existing mfrm_results() object. It does not refit the model, recompute diagnostics, or add new validity rules. Instead, it turns the comprehensive first-screen result into a first-screen table, section plan, claim-readiness table, report-gap table, report-index table, template-index table, fit-criteria table, result-specific fit evidence summaries, fit-reporting wording templates, precision/separation reporting templates, bias/DFF reporting templates, misfit/pathway reporting templates, linking/anchor reporting templates, ZSTD-convention table, evidence-boundary table, next-action table, and optional Markdown or HTML report.

Usage

```
mfrm_report(
  x,
  style = c("qc", "apa", "validation", "reviewer", "technical"),
  output = c("object", "markdown", "html", "tables")
)
```

Arguments

x	An mfrm_results() object.
style	Report emphasis. "qc" is the default first-screen report. "apa" emphasizes manuscript wording, "validation" emphasizes the validity-argument boundary, "reviewer" emphasizes reviewer response preparation, and "technical" emphasizes appendix/reproducibility routes.
output	Return format: "object" for an mfrm_report object, "markdown" for a character scalar, "html" for a temporary HTML file, or "tables" for the report's named data-frame list.

Details

The intended workflow is:

1. Create `res <- mfrm_results(fit, include = ...)`.
2. Inspect `summary(res)$triage` and `summary(res)$next_actions`.
3. Create `report <- mfrm_report(res, style = "qc")`.
4. Read `summary(report)` and `report$first_screen` before opening detailed report tables.
5. Use `report$report_index` to choose the next `PrimaryTable`, `TemplateTable`, plot route, or export route.
6. Use `report$template_index` before copying APA/QC/validation wording.
7. Use `style = "apa"`, `"validation"`, `"reviewer"`, or `"technical"` only when that reporting question is needed.

Report rows deliberately distinguish evidence from claims. The `first_screen` table is the compact entry point: it gives an overall row and one row per major evidence area with status, readiness, main issue, next action, and primary route. The `summary.mfrm_report` method summarizes that first screen into immediate actions, optional not-requested sections, claim-readiness counts, report gaps, and template-boundary rows without introducing a new pass/fail decision. The default print method follows the same short reading order and does not print every detailed evidence table. HTML output places the same reader guidance and report-summary tables before the full Markdown text so the browser view starts from the first-screen route. The `report_index` table is the detailed evidence-route index: it lists the major report areas, evidence status, readiness label, review-signal count, and the primary/template tables, evidence routes, template routes, plot routes, export route, and `mfrm_results(include = ...)` preset to inspect next. In ordinary use, open detailed tables through the `PrimaryTable` and `TemplateTable` columns rather than scanning every element of `report$tables`. The `template_index` table then stacks all fit, precision, bias, misfit, and linking wording templates into a single boundary/claim-strength index before users drill into the area-specific template tables. The `claim_readiness` table marks which report claims are ready, caveated, unavailable, or require additional requested sections. The `report_gaps` table turns those statuses into follow-up actions. The fit-specific tables keep multiple MnSq threshold profiles, observed fit-status counts, and engine-vs-FACETS-style ZSTD conventions visible, including the small-df/capping boundary used for FACETS-style ZSTD review. They summarize the stored `fit_measures` component from `mfrm_results()`; `mfrm_report()` itself does not recompute diagnostics. The `fit_reporting_templates` table turns those counts into cautious APA/QC/validation/reviewer wording scaffolds while keeping MnSq, ZSTD standardization, df sensitivity, and separation/reliability in separate sentences. All reporting-template tables share `EvidenceTable`, `EvidenceRoute`, `BoundaryType`, `ClaimStrength`, and `RecommendedUse` columns so each template can be traced back to its evidence and claim boundary. `template_index` stacks those columns across all template areas so report authors can review unsupported or caveated wording before opening the full template text. The `precision_reporting_templates` table does the same for separation, reliability, and strata using the stored precision review and `diagnostics$reliability`. The `bias_reporting_templates` table is available when the source result was built with `include = "bias"` and keeps facet-level screens, interaction-bias contrasts, DFF follow-up, and fairness conclusions in separate lanes. The `misfit_reporting_templates` table is available when the source result was built with `include = "misfit_review"` and keeps unexpected responses, displacement, pathway-map evidence, and case-review actions separate. The `linking_reporting_templates` table is available when the source result was built with `include = "linking"` and keeps anchor readiness, drift review, equating-chain review, and GPCM support boundaries separate. For example, fit

and separation are not collapsed into a single pass/fail statement; bias screens are not treated as final fairness conclusions; pathway/misfit rows are case-review prompts; and drift/equating claims require multiple fitted forms or waves.

Value

Depending on `output`, an `mfrm_report` object, a Markdown character scalar, an `mfrm_report_html` object, or a named list of data frames.

See Also

`mfrm_results()`, `export_mfrm_results()`, `build_apa_outputs()`, `reporting_checklist()`, `mfrmr_output_guide()`

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:6], , drop = FALSE]
fit <- fit_mfrm(toy_small, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
res <- mfrm_results(fit, include = c("fit", "diagnostics", "tables"))

report <- mfrm_report(res, style = "qc")
summary(report)
report$first_screen
report$report_index[, c("Area", "Readiness", "PrimaryTable",
                        "TemplateTable", "PlotRoute")]
report$template_index[, c("Area", "Topic", "BoundaryType",
                          "ClaimStrength", "EvidenceRoute")]

# Open detailed evidence only after the index points to it.
fit_primary <- report$report_index$PrimaryTable[
  report$report_index$Area == "Fit"
][1]
report$tables[[fit_primary]]

mfrm_report(res, output = "markdown")
mfrm_report(res, output = "html")
```

`mfrm_results` *Build comprehensive first-screen MFRM results*

Description

Build comprehensive first-screen MFRM results

Usage

```
mfrm_results(
  fit,
  include = "standard",
  response_time = NULL,
  response_time_data = NULL,
  response_time_facets = NULL,
  response_time_score = NULL,
  output = c("object", "summary", "tables", "html")
)
```

Arguments

fit	Output from fit_mfrm() or run_mfrm_facets(). A standard long-format data.frame is also accepted when person and score columns can be inferred unambiguously from common names such as Person and Score; all remaining columns are treated as facets.
include	Result sections or purpose presets to include. Purpose presets are "standard", "publication", "validation", "facets", "bias", "misfit_review", "linking", "network", "gpcm_review", and "all". Section names include "fit", "diagnostics", "tables", "precision", "reporting", "categories", "plots", "facets_fit", "bias", "misfit", "linking", "network", and "apa".
response_time	Optional response-time column name. When NULL and include contains "response_time", conservative column names such as ResponseTime, response_time, or RT are detected when available.
response_time_data	Optional original long-format data containing the timing column. Required for already fitted objects unless the timing column is still present in fit$prep$data.
response_time_facets	Optional facet columns for response-time summaries. Defaults to the fitted model's source facet columns when available.
response_time_score	Optional score column for response-time summaries. Defaults to the fitted model's source score column when available.
output	Return format: "object" for an mfrm_results object, "summary" for its compact summary, "tables" for a named list of available data frames, or "html" for a temporary HTML report.

Details

mfrm_results() is a high-level result object. It does not introduce a new estimator or a new validity rule. It fits only when fit is a data frame, computes diagnostics automatically when needed, and collects output from existing helpers such as diagnose_mfrm(), fit_measures_table(), precision_review_report(), and reporting_checklist(). Sections that are unsupported for a particular fit are retained in the status table as not_available rather than stopping the whole results workflow. The returned object also carries next_actions and input$reproducible_code so users can move from the comprehensive first screen to explicit reporting or replay code.

Value

Depending on `output`, an `mfrm_results` object, a `summary.mfrm_results` object, a named table list, or an `mfrm_results_html` object.

Include presets

- `"standard"`: fit, diagnostics, tables, precision, reporting, categories, and plot routes
- `"publication"`: standard sections plus APA output assembly
- `"validation"`: standard sections plus FACETS-fit/df-sensitivity review
- `"facets"`: fit, diagnostics, tables, categories, plots, and FACETS-fit review for FACETS-facing migration work
- `"bias"` / `"bias_review"`: standard sections plus facet-level bias-screen guidance; interaction bias still requires explicit facet-pair selection
- `"misfit"` / `"misfit_review"`: standard sections plus unexpected-response, displacement, and pathway-map case-review surfaces
- `"linking"` / `"anchors"`: standard sections plus anchor-readiness and operational linking-review surfaces from the fitted object's stored anchor review; drift and screened-chain review still require multiple fitted forms or waves
- `"network"`: standard sections plus network/connectivity review
- `"response_time"`: descriptive response-time QC review when timing metadata are supplied through `response_time` / `response_time_data`
- `"gpcm_review"`: standard sections with bounded-`GPCM` caveats retained in the collected summaries and reports
- `"all"`: standard sections plus FACETS-fit, network, APA, and response-time sections

Response-time metadata

Response-time review is opt-in and descriptive. It does not change fitted MFRM estimates, fit a joint speed-accuracy model, or create automatic exclusion rules. Use `include = "response_time"` together with `response_time = "ResponseTime"`. When `fit` is an already fitted object, also supply `response_time_data = original_data` because fitted objects keep only the measurement columns needed for estimation.

What to inspect first

Start with `summary(res)`. The most useful fields are:

- `overview`: input mode, model, method, table count, and plot-route count
- `triage`: first-screen signals ordered by unavailable/review/info/ok
- `status`: which sections were available, skipped, or unsupported
- `plot_map`: the supported `plot(res, type = ...)` routes for this object
- `next_actions`: recommended follow-up calls
- `reproducible_code`: replay scaffold for the first-screen route

Data-frame input

Direct data-frame input is intentionally conservative. It is intended for standard columns such as `Person`, `Score`, `Rater`, and `Criterion`. For research scripts, use `fit_mfrm()` or `run_mfrm_facets()` explicitly when column roles, model, method, anchors, or missing-data rules need to be documented. Use `mfrm_results_interactive()` only when you want an opt-in column-selection wizard in an interactive session.

Visualization and HTML

`plot(res)` routes to a FACETS-style model-level visual bundle by default. Other routes include `plot(res, type = "wright")`, `"pathway"`, `"qc"`, `"category"`, `"anchors"`, and `"tables"`. `output = "html"` writes a lightweight temporary HTML file; use `launch_mfrmr_viewer()` when you want an optional local Shiny reader for an already-created `mfrm_results` object. Use `export_mfrm_results()` for a lightweight download of the comprehensive results object, or `export_mfrm_bundle()` when a fit-centered durable analysis archive is needed.

Typical workflow

1. Fit explicitly with `fit_mfrm()` in scripts and manuscripts.
2. Call `res <- mfrm_results(fit)`.
3. Read `summary(res)$triage`, `summary(res)$status`, `summary(res)$plot_map`, and `summary(res)$next_actions`.
4. Use `plot(res, type = "qc")` for the first visual screen.
5. Optionally inspect the same result with `launch_mfrmr_viewer()` in an interactive session.
6. Use `build_summary_table_bundle()` or the helper named in `summary(res)$next_actions` for report-specific follow-up.

See Also

`fit_mfrm()`, `run_mfrm_facets()`, `diagnose_mfrm()`, `reporting_checklist()`, `build_summary_table_bundle()`, `export_mfrm_results()`, `launch_mfrmr_viewer()`, `mfrmr_output_guide()`

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:8], , drop = FALSE]

# JML keeps the help example fast; use the recommended workflow settings
# for final analyses.
fit <- fit_mfrm(toy_small, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
res <- mfrm_results(fit)

sx <- summary(res)
sx$overview
sx$triage
sx$plot_map
sx$next_actions
mfrm_results(fit, include = "validation", output = "summary")$status
```

```
plot(res, type = "qc", draw = FALSE)

# Direct data-frame input is available for conservative exploratory use
# when Person and Score columns are unambiguous.
mfrm_results(
  toy_small,
  include = c("fit", "diagnostics"),
  output = "summary"
)$mapping
```

`mfrm_results_interactive` *Interactively choose data-frame columns before calling* `mfrm_results()`

Description

Interactively choose data-frame columns before calling `mfrm_results()`

Usage

```
mfrm_results_interactive(
  data,
  include = "standard",
  output = c("object", "summary", "tables", "html")
)
```

Arguments

`data`	A long-format data frame.
`include`	Passed to `mfrm_results()`.
`output`	Passed to `mfrm_results()`.

Details

This helper is deliberately opt-in and stops in non-interactive sessions. It asks the user to choose the person, score, optional weight, and facet columns, prints reproducible code for the selected roles, then fits the default legacy-compatible `RSM`/`JML` route before calling `mfrm_results()`. Use explicit `fit_mfrm()` calls in scripts, Quarto documents, tests, and reproducible analyses.

Value

The selected `mfrm_results()` output.

Why this helper is opt-in

Interactive prompts are useful at the console but are unsafe defaults for reproducible analysis, package checks, batch scripts, and manuscripts. The helper therefore prints replay code and leaves the scripted route explicit.

See Also

mfrm_results(), fit_mfrm(), run_mfrm_facets()

Examples

```
if (interactive()) {
  toy <- load_mfrmr_data("example_core")
  res <- mfrm_results_interactive(toy)
}
```

---

mfrm_threshold_profiles

List literature-based warning threshold profiles

Description

List literature-based warning threshold profiles

Usage

```
mfrm_threshold_profiles()
```

Details

Use this function to inspect available profile presets before calling build_visual_summaries().

profiles contains thresholds used by warning logic (sample size, fit ratios, PCA cutoffs, etc.). pca_reference_bands contains literature-oriented descriptive bands used in summary text.

Value

An object of class mfrm_threshold_profiles with profiles (strict, standard, lenient) and pca_reference_bands.

Interpreting output

- profiles: numeric threshold presets (strict, standard, lenient).
- pca_reference_bands: narrative reference bands for PCA interpretation.

Typical workflow

1. Review presets with `mfrm_threshold_profiles()`.
2. Pick a default profile for project policy.
3. Override only selected fields in `build_visual_summaries()` when needed.

See Also

`build_visual_summaries()`

Examples

```
profiles <- mfrm_threshold_profiles()
s_profiles <- summary(profiles)
s_profiles$overview
```

---

normalize_conquest_overlap_files

Normalize extracted ConQuest overlap files to the `mfrmr` *review contract*

Description

Normalize extracted ConQuest overlap files to the `mfrmr` review contract

Usage

```
normalize_conquest_overlap_files(
  population_file,
  item_file,
  case_file,
  population_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  item_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  case_delimiter = c("auto", "comma", "tab", "semicolon", ",", "\t", ";"),
  conquest_population_term = "auto",
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
  conquest_item_estimate = "auto",
  conquest_case_person = "auto",
  conquest_case_estimate = "auto",
  keep_extra_columns = TRUE
)
```

Arguments

`population_file`
: Path to an extracted ConQuest population-parameter table in CSV/TSV/TXT form.

`item_file`
: Path to an extracted ConQuest item-estimate table in CSV/TSV/TXT form.

`case_file`
: Path to an extracted ConQuest case-level EAP table in CSV/TSV/TXT form.

`population_delimiter`
: Delimiter for `population_file`. `"auto"` chooses comma, tab, or semicolon from the file extension/header line.

`item_delimiter`
: Delimiter for `item_file`. `"auto"` chooses from the file extension/header line.

`case_delimiter`
: Delimiter for `case_file`. `"auto"` chooses from the file extension/header line.

`conquest_population_term`
: Column in `population_file` that stores parameter names. `"auto"` tries conservative aliases such as `Parameter` and `Term`.

`conquest_population_estimate`
: Column in `population_file` that stores parameter estimates. `"auto"` tries aliases such as `Estimate` and `Est`.

`conquest_item_id`
: Column in `item_file` that stores the item identifier as extracted by the user. `"auto"` tries aliases such as `ResponseVar`, `ItemID`, `Item`, and `Label`.

`conquest_item_estimate`
: Column in `item_file` that stores item estimates. `"auto"` tries aliases such as `Estimate`, `Est`, and `Facility`.

`conquest_case_person`
: Column in `case_file` that stores person IDs. `"auto"` tries conservative aliases such as `Person`, `PID`, and `Sequence ID`.

`conquest_case_estimate`
: Column in `case_file` that stores case EAP estimates. `"auto"` tries conservative aliases such as `Estimate`, `EAP_1`, and `EAP`.

`keep_extra_columns`
: If `TRUE`, keep all remaining columns after the standardized identifier and estimate columns.

Details

This helper is a thin file-wrapper around `normalize_conquest_overlap_tables()`. It is intentionally limited to already extracted tabular files and does not parse raw ConQuest report text.

The recommended workflow is:

1. export an exact-overlap bundle with `build_conquest_overlap_bundle()`;
2. extract the relevant ConQuest tables to CSV/TSV/TXT files;
3. call `normalize_conquest_overlap_files()` on those files;
4. pass the result to `review_conquest_overlap()`.

Read `summary(normalized)$normalization_scope` before review to confirm that the files were treated as extracted tables, not raw ConQuest report text, and to check duplicate-ID / non-numeric-estimate pre-review flags.

Value

A named list with class `mfrm_conquest_overlap_tables`.

See Also

`normalize_conquest_overlap_tables()`, `review_conquest_overlap()`

Examples

```
bundle <- build_conquest_overlap_bundle()
tmp_dir <- tempdir()
pop_path <- file.path(tmp_dir, "cq_pop.csv")
item_path <- file.path(tmp_dir, "cq_item.tsv")
case_path <- file.path(tmp_dir, "cq_case.csv")
utils::write.csv(
  data.frame(
    Term = bundle$mfrmr_population$Parameter,
    Est = bundle$mfrmr_population$Estimate
  ),
  pop_path,
  row.names = FALSE
)
utils::write.table(
  data.frame(
    Item = bundle$mfrmr_item_estimates$ResponseVar,
    Est = bundle$mfrmr_item_estimates$Estimate
  ),
  item_path,
  sep = "\t",
  row.names = FALSE
)
utils::write.csv(
  data.frame(
    PID = bundle$mfrmr_case_eap$Person,
    EAP = bundle$mfrmr_case_eap$Estimate
  ),
  case_path,
  row.names = FALSE
)
normalized <- normalize_conquest_overlap_files(
  population_file = pop_path,
  item_file = item_path,
  case_file = case_path,
  conquest_population_term = "Term",
  conquest_population_estimate = "Est",
  conquest_item_id = "Item",
  conquest_item_estimate = "Est",
  conquest_case_person = "PID",
  conquest_case_estimate = "EAP"
)
summary(normalized)$normalization_scope
review <- review_conquest_overlap(bundle, normalized)
```

```
summary(review)$summary
```

`normalize_conquest_overlap_tables` *Normalize extracted ConQuest overlap tables to the* `mfrmr` *review contract*

Description

Normalize extracted ConQuest overlap tables to the `mfrmr` review contract

Usage

```
normalize_conquest_overlap_tables(
  conquest_population,
  conquest_item_estimates,
  conquest_case_eap,
  conquest_population_term = "auto",
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
  conquest_item_estimate = "auto",
  conquest_case_person = "auto",
  conquest_case_estimate = "auto",
  keep_extra_columns = TRUE
)
```

Arguments

`conquest_population`
: Extracted ConQuest population-parameter table as a data.frame.

`conquest_item_estimates`
: Extracted ConQuest item-estimate table as a data.frame.

`conquest_case_eap`
: Extracted ConQuest case-level EAP table as a data.frame.

`conquest_population_term`
: Column in `conquest_population` that stores parameter names. `"auto"` tries conservative aliases such as `Parameter` and `Term`.

`conquest_population_estimate`
: Column in `conquest_population` that stores parameter estimates. `"auto"` tries aliases such as `Estimate` and `Est`.

`conquest_item_id`
: Column in `conquest_item_estimates` that stores the item identifier as exported or extracted by the user. `"auto"` tries aliases such as `ResponseVar`, `ItemID`, `Item`, and `Label`.

conquest_item_estimate
: Column in conquest_item_estimates that stores item estimates. "auto" tries aliases such as Estimate, Est, and Facility.

conquest_case_person
: Column in conquest_case_eap that stores person IDs. "auto" tries conservative aliases such as Person, PID, and Sequence ID.

conquest_case_estimate
: Column in conquest_case_eap that stores case EAP estimates. "auto" tries conservative aliases such as Estimate, EAP_1, and EAP.

keep_extra_columns
: If TRUE, keep all remaining columns after the standardized identifier and estimate columns.

Details

This helper does not parse raw ConQuest text output. It standardizes already extracted tables to the contract used by review_conquest_overlap():

- population parameters become columns Parameter, Estimate, and EstimateNonNumeric;
- item estimates become columns ItemID, Estimate, and EstimateNonNumeric;
- case summaries become columns Person, Estimate, and EstimateNonNumeric.

The resulting object is intentionally conservative. It does not infer whether item IDs correspond to exported response variables or original item levels; that matching step remains part of review_conquest_overlap(), where the standardized ConQuest tables are compared against a concrete overlap bundle.

Value

A named list with class mfrm_conquest_overlap_tables.

Output

The returned object has class mfrm_conquest_overlap_tables and includes:

- summary: one-row normalization summary
- conquest_population: standardized population table
- conquest_item_estimates: standardized item table
- conquest_case_eap: standardized case table
- settings: source-column metadata
- notes: interpretation notes

Read summary(normalized)$normalization_scope before review to confirm that the object contains extracted tabular inputs, not parsed raw ConQuest report text, and to check duplicate-ID / non-numeric-estimate pre-review flags.

See Also

build_conquest_overlap_bundle(), review_conquest_overlap()

Examples

```
normalized <- normalize_conquest_overlap_tables(
  conquest_population = data.frame(
    Term = c("(Intercept)", "GroupB", "sigma2"),
    Est = c(0, 0.2, 1)
  ),
  conquest_item_estimates = data.frame(
    Item = c("I1", "I2"),
    Est = c(-0.2, 0.2)
  ),
  conquest_case_eap = data.frame(
    PID = c("P001", "P002"),
    EAP = c(-0.1, 0.1)
  ),
  conquest_population_term = "Term",
  conquest_population_estimate = "Est",
  conquest_item_id = "Item",
  conquest_item_estimate = "Est",
  conquest_case_person = "PID",
  conquest_case_estimate = "EAP"
)
summary(normalized)$normalization_scope
```

---

plot.apa_table *Plot an APA/FACETS table object using base R*

Description

Plot an APA/FACETS table object using base R

Usage

```
## S3 method for class 'apa_table'
plot(
  x,
  y = NULL,
  type = c("numeric_profile", "first_numeric"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from apa_table().
y	Reserved for generic compatibility.

type	Plot type: "numeric_profile" (column means) or "first_numeric" (distribution of the first numeric column).
main	Optional title override.
palette	Optional named color overrides.
label_angle	Axis-label rotation angle for bar-type plots.
draw	If TRUE, draw using base graphics.
...	Reserved for generic compatibility.

Details

Quick visualization helper for numeric columns in apa_table() output. It is intended for table QA and exploratory checks, not final publication graphics.

Value

A plotting-data object of class mfrm_plot_data.

Interpreting output

- "numeric_profile": compares column means to spot scale/centering mismatches.
- "first_numeric": checks distribution shape of the first numeric column.

Typical workflow

1. Build table with apa_table().
2. Run summary(tbl) for metadata.
3. Use plot(tbl, type = "numeric_profile") for quick numeric QC.

See Also

apa_table(), summary()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
tbl <- apa_table(fit, which = "summary")
p <- plot(tbl, draw = FALSE)
p2 <- plot(tbl, type = "first_numeric", draw = FALSE)
if (interactive()) {
  plot(
    tbl,
    type = "numeric_profile",
    main = "APA Numeric Profile (Customized)",
    palette = c(numeric_profile = "#2b8cbe", grid = "#d9d9d9"),
    label_angle = 45
  )
}
```

plot.mfrm_anchor_review *Plot an anchor-review object*

Description

Plot an anchor-review object

Usage

```
## S3 method for class 'mfrm_anchor_review'
plot(
  x,
  y = NULL,
  type = c("issue_counts", "facet_constraints", "level_observations"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from review_mfrm_anchors().
y	Reserved for generic compatibility.
type	Plot type: "issue_counts", "facet_constraints", or "level_observations".
main	Optional title override.
palette	Optional named colors.
label_angle	X-axis label angle for bar plots.
draw	If TRUE, draw using base graphics.
...	Reserved for generic compatibility.

Details

Base-R visualization helper for anchor-review outputs.

Value

A plotting-data object of class mfrm_plot_data.

Interpreting output

- "issue_counts": volume of each issue class.
- "facet_constraints": anchored/grouped/free mix by facet.
- "level_observations": observation support across levels.

Typical workflow

1. Run `review_mfrm_anchors()`.
2. Start with `plot(review, type = "issue_counts")`.
3. Inspect constraint and support plots before fitting.

See Also

`review_mfrm_anchors()`, `make_anchor_table()`

Examples

```
toy <- load_mfrmr_data("example_core")
review <- review_mfrm_anchors(toy, "Person", c("Rater", "Criterion"), "Score")
p <- plot(review, draw = FALSE)
```

---

plot.mfrm_bundle *Plot report/table bundles with base R defaults*

Description

Plot report/table bundles with base R defaults

Usage

```
## S3 method for class 'mfrm_bundle'
plot(x, y = NULL, type = NULL, ...)
```

Arguments

x	A bundle object returned by mfrmr table/report helpers.
y	Reserved for generic compatibility.
type	Optional plot type. Available values depend on bundle class.
...	Additional arguments forwarded to class-specific plotters.

Details

`plot()` dispatches by bundle class:

- `mfrm_unexpected` -> `plot_unexpected()`
- `mfrm_fair_average` -> `plot_fair_average()`
- `mfrm_displacement` -> `plot_displacement()`
- `mfrm_interrater` -> `plot_interrater_agreement()`
- `mfrm_facets_chisq` -> `plot_facets_chisq()`
- `mfrm_bias_interaction` -> `plot_bias_interaction()`

- `mfrm_bias_count` -> bias-count plots (cell counts / low-count rates)
- `mfrm_fixed_reports` -> pairwise-contrast diagnostics
- `mfrm_visual_summaries` -> warning/summary message count plots
- `mfrm_category_structure` -> default base-R category plots
- `mfrm_category_curves` -> overview (default), ogive, CCC / category probability / conditional probability, cumulative, total-information, and category-specific-information plots
- `mfrm_rating_scale` -> category-counts/threshold plots
- `mfrm_measurable` -> measurable-data coverage/count plots
- `mfrm_unexpected_after_bias` -> post-bias unexpected-response plots
- `mfrm_output_bundle` -> graph/score output-file diagnostics, including `type = "score_se"` when scorefile SE columns are available
- `mfrm_residual_pca` -> residual PCA scree, parallel-analysis, or loadings views via `plot_residual_pca()`
- `mfrm_specifications` -> facet/anchor/convergence plots
- `mfrm_data_quality` -> dashboard, quality-flag, score-map, facet-pattern, and row/category/missing-row plots
- `mfrm_facets_fit_review` -> FACETS-style df-sensitivity plot
- `mfrm_fit_measures` -> fit-status counts, Infit/Outfit scatter, measure intervals, and FACETS-style df-sensitivity plots
- `mfrm_iteration_report` -> replayed-iteration trajectories
- `mfrm_subset_connectivity` -> subset-observation/connectivity plots
- `mfrm_facet_statistics` -> facet statistic profile plots
- `mfrm_export_bundle` / `mfrm_summary_appendix_export` -> export handoff plots (`formats`, `artifact_groups`, `selection_tables`, `selection_handoff`, `selection_handoff_bundles`, `selection_handoff_roles`, `selection_handoff_role_sections`, `selection_bundles`, `selection_roles`, `selection_sections`)

If a class is outside these families, use dedicated plotting helpers or custom base R graphics on component tables.

For `mfrm_category_curves`, pass `preset = "monochrome"` for grayscale/line-type output. Cumulative `.5` boundary lines are shown only for interpretable in-range boundaries by default; use `boundary_status = "all"` to show every finite boundary estimate or `boundary_status = "none"` / `show_cumulative_boundaries = FALSE` to suppress those vertical boundary lines. Use `plot_data(x, component = "plot_long")` on a category-curve bundle when you want one ggplot2/plotly-friendly table across all curve families.

Value

A plotting-data object of class `mfrm_plot_data`.

Interpreting output

The returned object is plotting data (`mfrm_plot_data`) that captures the selected route and reusable data; set `draw = TRUE` for immediate base graphics.

Typical workflow

1. Create bundle output (e.g., `unexpected_response_table()`).
2. Inspect routing with `summary(bundle)` if needed.
3. Call `plot(bundle, type = ..., draw = FALSE)` to obtain reusable plot data.

See Also

`summary()`, `plot_unexpected()`, `plot_fair_average()`, `plot_displacement()`

Examples

```
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
t4 <- unexpected_response_table(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 5)
p <- plot(t4, draw = FALSE)
vis <- build_visual_summaries(fit, diagnose_mfrm(fit, residual_pca = "none"))
p_vis <- plot(vis, type = "comparison", draw = FALSE)
spec <- specifications_report(fit)
p_spec <- plot(spec, type = "facet_elements", draw = FALSE)
if (interactive()) {
  plot(
    t4,
    type = "severity",
    draw = TRUE,
    main = "Unexpected Response Severity (Customized)",
    palette = c(higher = "#d95f02", lower = "#1b9e77", bar = "#2b8cbe"),
    label_angle = 45
  )
  plot(
    vis,
    type = "comparison",
    draw = TRUE,
    main = "Warning vs Summary Counts (Customized)",
    palette = c(warning = "#cb181d", summary = "#3182bd"),
    label_angle = 45
  )
}
```

---

plot.mfrm_data_description

Plot a data-description object

Description

Plot a data-description object

Usage

```
## S3 method for class 'mfrm_data_description'
plot(
  x,
  y = NULL,
  type = c("score_distribution", "facet_levels", "missing"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from describe_mfrm_data().
y	Reserved for generic compatibility.
type	Plot type: "score_distribution", "facet_levels", or "missing".
main	Optional title override.
palette	Optional named colors (score, facet, missing).
label_angle	X-axis label angle for bar plots.
draw	If TRUE, draw using base graphics.
...	Reserved for generic compatibility.

Details

This method draws quick pre-fit quality views from describe_mfrm_data():

- score distribution balance
- facet-level structure size
- missingness by selected columns

Value

A plotting-data object of class mfrm_plot_data.

Interpreting output

- "score_distribution": bar chart of weighted observation counts per score category. Y-axis is WeightedN (sum of weights for each category). Categories with very few observations (< 10) may produce unstable threshold estimates. A roughly uniform or unimodal distribution is ideal; heavy floor/ceiling effects compress the measurement range.

- "facet_levels": bar chart showing the number of distinct levels per facet. Useful for verifying that the design structure matches expectations (e.g., expected number of raters or criteria). Very large numbers of levels increase computation time and may require higher maxit in fit_mfrm().
- "missing": bar chart of missing-value counts per input column. Columns with non-zero counts should be investigated before fitting—rows with missing scores, persons, or facet IDs are dropped during estimation.

Typical workflow

1. Run describe_mfrm_data() before fitting.
2. Inspect summary(ds) and plot(ds, type = "missing").
3. Check category/facet balance with other plot types.
4. Fit model after resolving obvious data issues.

See Also

describe_mfrm_data(), plot()

Examples

```
toy <- load_mfrmr_data("example_core")
ds <- describe_mfrm_data(toy, "Person", c("Rater", "Criterion"), "Score")
p <- plot(ds, draw = FALSE)
```

---

plot.mfrm_design_evaluation

Plot a design-simulation study

Description

Plot a design-simulation study

Usage

```
## S3 method for class 'mfrm_design_evaluation'
plot(
  x,
  facet = c("Rater", "Criterion", "Person"),
  metric = c("separation", "reliability", "infit", "outfit", "misfitrate",
   "severityrmse", "severitybias", "convergencerate", "elapsedsec", "mincategorycount",
   "designdensity", "plannedmissingrate", "linkpersons", "linkfraction", "linkraters",
     "mincommonpersons", "zerocommonpairs", "pairsshorttarget"),
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_design().
facet	Facet to visualize.
metric	Metric to plot.
x_var	Design variable used on the x-axis. When x was generated from a sim_spec with custom public facet names, the corresponding aliases (for example n_judge, n_task, judge_per_person) are also accepted. Role keywords (person, rater, criterion, assignment) are accepted as an abstraction over the current two-facet schema.
group_var	Optional design variable used for separate lines. The same alias rules as x_var apply.
draw	If TRUE, draw with base graphics; otherwise return plotting data.
...	Reserved for generic compatibility.

Details

This method is designed for quick design-planning scans rather than polished publication graphics.

Useful first plots are:

- rater metric = "separation" against x_var = "n_person"
- criterion metric = "severityrmse" against x_var = "n_person" when you want aligned recovery error rather than raw location shifts
- rater metric = "convergencerate" against x_var = "raters_per_person"
- sparse linked metric = "plannedmissingrate", "mincommonpersons", "zerocommonpairs", or "pairsshorttarget" to review planned missingness and rater-pair linkage separately from recovery metrics

Value

If draw = TRUE, invisibly returns a plotting-data list. If draw = FALSE, returns that list directly. The returned list includes resolved canonical variables (x_var, group_var) together with public labels (x_label, group_label), design_variable_aliases, and design_descriptor, plus planning_scope, planning_constraints, and planning_schema.

See Also

evaluate_mfrm_design(), summary.mfrm_design_evaluation

Examples

```
sim_eval <- suppressWarnings(evaluate_mfrm_design(
  n_person = c(8, 12),
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 1,
  reps = 1,
  maxit = 30,
```

```
  seed = 123
))
p <- plot(sim_eval, facet = "Rater", metric = "separation", x_var = "n_person", draw = FALSE)
c(p$facet, p$x_var)
```

---

plot.mfrm_diagnostic_screening *Plot a diagnostic-screening validation study*

Description

Builds an integrated visual summary from evaluate_mfrm_diagnostic_screening() output. The default view combines legacy residual, strict marginal, strict pairwise, strict combined, and optional report-index review rates so simulation results can be inspected in one operating-characteristic surface.

Usage

```
## S3 method for class 'mfrm_diagnostic_screening'
plot(
  x,
  type = c("overview", "report", "contrast", "runtime"),
  metric = NULL,
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_diagnostic_screening().
type	Plot family. "overview" combines screening and optional report rates or counts. "report" focuses on mfrm_report() review signals. "contrast" plots misspecification-minus-well-specified contrasts. "runtime" plots elapsed-time summaries.
metric	Metric family. Use NULL or "auto" for the default within each type. Supported values are documented by error messages and include "rate", "count", "magnitude", "elapsed", and "per_observation" depending on type.
x_var	Design variable for the horizontal axis. Public design aliases from a simulation specification are accepted.
group_var	Optional additional design variable to include in group labels. Public design aliases are accepted.
draw	Logical; if FALSE, return the plot-data bundle without drawing.
...	Reserved for future extensions.

Value

An mfrm_plot_data object with reusable metadata, a long-form plot_long table, and interpretation handoff tables (overview, reading_order, next_actions, reporting_notes, and figure_recipes). When draw = TRUE, the object is returned invisibly after drawing.

Examples

```
diag_eval <- evaluate_mfrm_diagnostic_screening(
  design = list(person = 10, rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  include_report = TRUE,
  seed = 123
)
plot(diag_eval, type = "overview", draw = FALSE)
plot_data(diag_eval, type = "overview", component = "plot_long")
plot_data(diag_eval, type = "overview", component = "next_actions")
plot_data(diag_eval, type = "overview", component = "figure_recipes")
plot(diag_eval, type = "report", metric = "rate", draw = FALSE)
```

---

plot.mfrm_facets_run *Plot outputs from a legacy-compatible workflow run*

Description

Plot outputs from a legacy-compatible workflow run

Usage

```
## S3 method for class 'mfrm_facets_run'
plot(x, y = NULL, type = c("fit", "qc"), ...)
```

Arguments

x	A mfrm_facets_run object from run_mfrm_facets().
y	Unused.
type	Plot route: "fit" delegates to plot.mfrm_fit() and "qc" delegates to plot_qc_dashboard().
...	Additional arguments passed to the selected plot function.

Details

This method is a router for fast visualization from a one-shot workflow result:

- type = "fit" for model-level displays.
- type = "qc" for multi-panel quality-control diagnostics.

Value

A plotting object from the delegated plot route.

Interpreting output

Returns the plotting object produced by the delegated route: `plot.mfrm_fit()` for `"fit"` and `plot_qc_dashboard()` for `"qc"`.

Typical workflow

1. Run `run_mfrm_facets()`.
2. Start with `plot(out, type = "fit", draw = FALSE)`.
3. Continue with `plot(out, type = "qc", draw = FALSE)` for diagnostics.

See Also

`run_mfrm_facets()`, `plot.mfrm_fit()`, `plot_qc_dashboard()`, mfrmr_visual_diagnostics, mfrmr_workflow_methods

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
p_fit <- plot(out, type = "fit", draw = FALSE)
p_fit$wright_map$data$plot
p_qc <- plot(out, type = "qc", draw = FALSE)
p_qc$data$plot
```

---

plot.mfrm_facet_nesting

Plot the pairwise nesting index matrix

Description

Renders the directed nesting index $1 - H(B \mid A)/H(B)$ as a heatmap between facet pairs, highlighting fully nested relationships close to 1. Colour scale runs from 0 (crossed, white / cold) to 1 (fully nested, dark).

Usage

```
## S3 method for class 'mfrm_facet_nesting'
plot(x, preset = c("standard", "publication", "compact", "monochrome"), ...)
```

Arguments

x	An mfrm_facet_nesting object.
preset	Plot preset.
...	Reserved.

Value

Invisibly, the matrix rendered.

See Also

detect_facet_nesting(), analyze_hierarchical_structure().

---

plot.mfrm_facet_sample_review

Plot a facet sample-size review

Description

Per-level observation counts rendered as a horizontal bar chart coloured by the Linacre sample-size band assigned in facet_small_sample_review(). Vertical dashed lines mark the sparse / marginal / standard thresholds so reviewers see where every facet level sits relative to the Linacre (1994) guidance.

Usage

```
## S3 method for class 'mfrm_facet_sample_review'
plot(
  x,
  top_n = NULL,
  preset = c("standard", "publication", "compact", "monochrome"),
  ...
)
```

Arguments

x	An mfrm_facet_sample_review object.
top_n	Optional integer; trim the y-axis to the top_n smallest level counts per facet. NULL (default) keeps all.
preset	One of "standard", "publication", "compact", "monochrome".
...	Reserved.

Value

Invisibly, the data.frame used for the plot.

See Also

facet_small_sample_review().

---

plot.mfrm_fit *Plot fitted MFRM results with base R*

Description

Plot fitted MFRM results with base R

Usage

```
## S3 method for class 'mfrm_fit'
plot(
  x,
  type = NULL,
  facet = NULL,
  top_n = 30,
  theta_range = c(-6, 6),
  theta_points = 241,
  title = NULL,
  palette = NULL,
  label_angle = 45,
  show_ci = FALSE,
  ci_level = 0.95,
  group = NULL,
  diagnostics = NULL,
  include_fit_measures = TRUE,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  ...
)
```

Arguments

x	An mfrm_fit object from fit_mfrm().
type	Plot type. Use NULL, "bundle", or "all" for the three-part fit bundle; otherwise choose one of "facet", "person", "step", "wright", "pathway", "ccc", "ccc_surface", or "category_surface".
facet	Optional facet name for type = "facet".
top_n	Maximum number of facet/step locations retained for compact displays.
theta_range	Numeric length-2 range for pathway, CCC, and category-surface plot data.

theta_points	Number of theta grid points used for pathway, CCC, and category-surface plot data.
title	Optional custom title.
palette	Optional color overrides.
label_angle	Rotation angle for x-axis labels where applicable.
show_ci	If TRUE, add approximate confidence intervals when available.
ci_level	Confidence level used when show_ci = TRUE.
group	Optional grouping for type = "wright" to overlay per-group person-density curves (DIF / DFF screening view). Either a column name (looked up first in group_data when supplied through ..., then in fit$prep$data) or a vector aligned with fit$facets$person. Ignored for other type values. To pass the source data alongside, use plot(fit, type = "wright", group = "MyCol", group_data = <df>).
diagnostics	Optional output from diagnose_mfrm(). When supplied, pathway plot data reuse it for fit_measures, fit_status, and curve_fit_status instead of recomputing diagnostics.
include_fit_measures	If TRUE (default), pathway plot data include tidy fit-measure and fit-status tables for custom R graphics. Set to FALSE when only the curve coordinates are needed.
draw	If TRUE, draw the plot with base graphics.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
...	Additional arguments ignored for S3 compatibility.

Details

This S3 plotting method provides the core fit-family visuals for mfrmr. When type is omitted, it returns the Wright map alone as an mfrm_plot_data object (the most useful single figure for a first inspection). Pass type = "bundle" (or "all" / "default") to obtain the legacy three-plot mfrm_plot_bundle containing a Wright map, pathway map, and category characteristic curves. The returned object always carries machine-readable metadata through the mfrm_plot_data contract, even when the plot is drawn immediately.

type = "wright" shows persons, facet levels, and step thresholds on a shared logit scale. Estimates are plotted as fitted, so the sign convention follows the fit: higher person values indicate higher ability, and higher non-person facet values indicate greater severity/difficulty under the default negative facet orientation. Facets listed in fit_mfrm(positive_facets = ...) are reversed (higher values raise expected scores); state the active orientation in figure captions when reporting. type = "pathway" shows expected score traces and dominant-category regions across theta. This expected-score display is distinct from the Bond-and-Fox-style measure-versus-fit "pathway" bubble chart used around FACETS/Winsteps output; for that display, use plot_bubble(). Its draw-free plot data also includes pathway_long, pathway_annotations, fit_measures, fit_status, and curve_fit_status, so R users can rebuild the pathway map in ggplot2, plotly, or a report pipeline while keeping the same underfit/overfit labels used by fit_measures_table(). type = "ccc" shows category response probabilities. type = "ccc_surface" or type = "category_surface" returns 3D-ready category-probability surface data for external rendering; it deliberately does not add a plotly/rgl dependency or replace the 2D CCC/pathway reporting figures. The returned object includes category_support, interpretation_guide, and reporting_policy tables so retained zero-frequency categories and manuscript-use boundaries remain visible to beginners. The remaining types ("facet", "person", "step", "shrinkage") provide compact location-specific displays.

Value

Invisibly, an `mfrm_plot_data` object (default and for any single `type`), or an `mfrm_plot_bundle` when `type = "bundle"` / `"all"` / `"default"`.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Use `plot(fit)` to inspect the Wright map at a glance.
3. Switch to `type = "pathway"`, `"ccc"`, or `"shrinkage"` for the relevant follow-up figure, or `type = "bundle"` for the three-plot overview when preparing a FACETS-style summary.

Further guidance

For a plot-selection guide and extended examples, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnosti package = "mfrmr")`.

See Also

`fit_mfrm()`, `plot_wright_unified()`, `plot_bubble()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "JML",
  model = "RSM",
  maxit = 30
)
wright <- plot(fit, draw = FALSE)
wright$data$plot
# Look for: persons clustered against the facet / step rows on the
#   shared logit axis. Large gaps between the person density and
#   the step / facet rails indicate weak targeting; ceiling /
#   floor stripes mean the test is too easy / hard.
bundle <- plot(fit, type = "bundle", draw = FALSE)
bundle$wright_map$data$plot
# Look for: pathway curves rising in the expected order with
#   visible dominant-category bands; CCC curves peaking sequentially
#   without one category being completely overlapped by neighbours.
surface <- plot(fit, type = "ccc_surface", draw = FALSE)
head(surface$data$surface)
surface$data$category_support
# Look for: every retained category having `Observed > 0`; categories
#   with zero observations are returned as a zero-observation slice and
#   should not be interpreted as a real score region.
surface$data$interpretation_guide
if (interactive()) {
```

```
  plot(
    fit,
    type = "wright",
    preset = "publication",
    title = "Customized Wright Map",
    show_ci = TRUE,
    label_angle = 45
  )
  plot(
    fit,
    type = "pathway",
    title = "Customized Pathway Map",
    palette = c("#1f78b4")
  )
  plot(
    fit,
    type = "ccc",
    title = "Customized Category Characteristic Curves",
    palette = c("#1b9e77", "#d95f02", "#7570b3")
  )
}
```

---

plot.mfrm_future_branch_active_branch *Plot a future arbitrary-facet planning active branch*

Description

Plot a future arbitrary-facet planning active branch

Usage

```
## S3 method for class 'mfrm_future_branch_active_branch'
plot(
  x,
  y = NULL,
  type = c("profile_metrics", "load_balance", "coverage", "readiness_tiers",
    "table_rows", "role_tables", "appendix_roles", "appendix_sections",
    "appendix_presets", "selection_handoff_presets", "selection_tables",
   "selection_handoff", "selection_handoff_bundles", "selection_handoff_roles",
    "selection_handoff_role_sections", "selection_bundles", "selection_roles",
    "selection_sections"),
  appendix_preset = c("recommended", "compact", "all", "methods", "results",
    "diagnostics", "reporting"),
  selection_value = c("count", "fraction"),
  draw = TRUE,
  main = NULL,
  palette = NULL,
  label_angle = 45,
```

```
  ...
)
```

Arguments

x	Output from the future-branch active planning scaffold stored in planning_schema$future_branch_act
y	Unused placeholder for generic compatibility.
type	Plot type: "profile_metrics" for recommended deterministic profile values by metric, "load_balance" for recommended load/balance values by metric, "coverage" for recommended coverage/connectivity values by metric, "readiness_tiers" for counts of structural tiers across the current active-branch design grid, "table_rows" / "role_tables" / "appendix_roles" for summary-table bundle QC, "appendix_sections" / "appendix_presets" for manuscript-facing appendix selection counts, "selection_handoff_presets for preset-level appendix handoff counts, "selection_tables" for appendix-selected future-branch tables ranked by row count within a preset, "selection_handoff" for section-aware plot-ready appendix handoff counts, "selection_handoff_bundles" for section-and-bundle plot-ready appendix handoff counts, "selection_handoff_roles" for role-aware plot-ready appendix handoff counts, "selection_handoff_role_sections" for role-by-section plot-ready appendix handoff counts, or "selection_bundles" / "selection_roles" / "selection_sections" for preset-filtered appendix selection summaries.
appendix_preset	Appendix preset used for selection_* plot types.
selection_value	For selection_* plot types, whether to plot exact counts ("count") or the matching exact fraction ("fraction") when that surface exposes one. selection_tables remains count-only because it represents table row counts rather than a normalized selection surface.
draw	If TRUE, draw with base graphics; otherwise return plotting data.
main	Optional title override.
palette	Optional named color overrides.
label_angle	Axis-label rotation angle.
...	Reserved for generic compatibility.

Value

A plotting-data object of class mfrm_plot_data.

See Also

summary.mfrm_future_branch_active_branch()

plot.mfrm_recovery_simulation *Plot parameter-recovery simulation results*

Description

Plot parameter-recovery simulation results

Usage

```
## S3 method for class 'mfrm_recovery_simulation'
plot(
  x,
  y = NULL,
  type = c("summary", "coverage", "errors", "scatter", "replications"),
 metric = c("rmse", "bias", "mae", "correlation", "coverage", "mcse_bias", "mcse_rmse",
    "raw_rmse", "raw_bias", "mean_se", "se_available"),
  parameter_type = NULL,
  facet = NULL,
  comparison = c("aligned", "unaligned"),
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_recovery().
y	Reserved for S3 generic compatibility.
type	Plot route: "summary" draws a metric from x$recovery_summary, "coverage" draws 95% coverage by parameter group, "errors" draws row-level recovery-error distributions, "scatter" draws truth against estimated values, and "replications" summarizes run status.
metric	Summary metric used when type = "summary". Supported values are "rmse", "bias", "mae", "correlation", "coverage", "mcse_bias", "mcse_rmse", "raw_rmse", "raw_bias", "mean_se", and "se_available".
parameter_type	Optional parameter type filter, such as "person", "facet", "step", "slope", or "population".
facet	Optional facet filter.
comparison	Error/estimate scale for row-level routes. "aligned" uses EstimateAligned / ErrorAligned; "unaligned" uses Estimate / ErrorRaw on the same comparison scale.
draw	If TRUE, draw with base graphics. If FALSE, return an mfrm_plot_data object with reusable plot tables and metadata.
...	Reserved for future extensions.

Details

These plots are intended as simulation-review graphics. They do not replace the row-level `x$recovery` table or the ADEMP metadata; they make the main recovery estimands easier to inspect during model-development and design checks. Coverage is displayed only for parameter groups with available standard errors.

Value

An `mfrm_plot_data` object. When `draw = TRUE`, the object is returned invisibly after drawing.

See Also

`evaluate_mfrm_recovery()`, `summary()`

Examples

```
rec <- evaluate_mfrm_recovery(
  n_person = 12,
  n_rater = 2,
  n_criterion = 2,
  reps = 1,
  maxit = 30,
  seed = 123
)
plot(rec, type = "summary", metric = "rmse", draw = FALSE)
```

---

plot.mfrm_signal_detection

Plot DIF/bias screening simulation results

Description

Plot DIF/bias screening simulation results

Usage

```
## S3 method for class 'mfrm_signal_detection'
plot(
  x,
  signal = c("dif", "bias"),
  metric = c("power", "false_positive", "estimate", "screen_rate",
    "screen_false_positive"),
  x_var = c("n_person", "n_rater", "n_criterion", "raters_per_person"),
  group_var = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from evaluate_mfrm_signal_detection().
signal	Whether to plot DIF or bias screening results.
metric	Metric to plot. For signal = "bias", prefer metric = "screen_rate" for the screening hit rate. The older metric = "power" spelling is retained as a backwards-compatible alias that maps to BiasScreenRate.
x_var	Design variable used on the x-axis. When x was generated from a sim_spec with custom public facet names, the corresponding aliases (for example n_judge, n_task, judge_per_person) are also accepted. Role keywords (person, rater, criterion, assignment) are accepted as an abstraction over the current two-facet schema.
group_var	Optional design variable used for separate lines. The same alias rules as x_var apply.
draw	If TRUE, draw with base graphics; otherwise return plotting data.
...	Reserved for generic compatibility.

Value

If draw = TRUE, invisibly returns plotting data. If draw = FALSE, returns that plotting-data list directly. The returned list includes resolved canonical variables (x_var, group_var) together with public labels (x_label, group_label), design_variable_aliases, design_descriptor, planning_scope, planning_constraints, planning_schema, display_metric, and interpretation_note so callers can label bias-side plots as screening summaries rather than formal power/error-rate displays.

See Also

evaluate_mfrm_signal_detection(), summary.mfrm_signal_detection

Examples

```
sig_eval <- suppressWarnings(evaluate_mfrm_signal_detection(
  n_person = 8,
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 1,
  reps = 1,
  maxit = 30,
  bias_max_iter = 1,
  seed = 123
))
plot(sig_eval, signal = "dif", metric = "power", x_var = "n_person", draw = FALSE)
```

plot.mfrm_summary_table_bundle *Plot a summary-table bundle for manuscript QC*

Description

Plot a summary-table bundle for manuscript QC

Usage

```
## S3 method for class 'mfrm_summary_table_bundle'
plot(
  x,
  y = NULL,
  type = c("table_rows", "role_tables", "appendix_roles", "appendix_sections",
    "appendix_presets", "selection_handoff_presets", "selection_tables",
   "selection_handoff", "selection_handoff_bundles", "selection_handoff_roles",
    "selection_handoff_role_sections", "selection_bundles", "selection_roles",
    "selection_sections", "numeric_profile", "first_numeric"),
  which = NULL,
  selection_value = c("count", "fraction"),
  appendix_preset = c("recommended", "compact", "all", "methods", "results",
    "diagnostics", "reporting"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from build_summary_table_bundle().
y	Reserved for generic compatibility.
type	Plot type: "table_rows" for returned-table sizes, "role_tables" for returned-table counts by reporting role, "appendix_roles" for returned-table counts by reporting role under the bundle's appendix-routing contract, "appendix_sections" for returned-table counts by manuscript-facing appendix section, "appendix_presets" for conservative appendix-preset counts, "selection_handoff_presets" for workflow-only preset-level appendix handoff counts, "selection_tables" / "selection_handoff" / "selection_handoff_bundles" / "selection_handoff_roles" / "selection_bundles" / "selection_roles" / "selection_sections" for workflow-only appendix selection surfaces when present in the bundle, "numeric_profile" for column means from a selected numeric table, or "first_numeric" for the distribution of the first numeric column in a selected table.
which	Optional table selector used for numeric plot types.

`selection_value`
: For `selection_*` plot types, whether to plot exact counts (`"count"`) or the corresponding exact fraction (`"fraction"`) when that surface exposes one.

`appendix_preset`
: Appendix preset used for `selection_*` plot types.

`main`
: Optional title override.

`palette`
: Optional named color overrides.

`label_angle`
: Axis-label rotation angle for bar-type plots.

`draw`
: If `TRUE`, draw using base graphics.

`...`
: Reserved for generic compatibility.

Details

This helper keeps summary-bundle plotting conservative. It either visualizes the bundle's own bundle-level indexes (`"table_rows"`, `"role_tables"`, `"appendix_roles"`, `"appendix_sections"`, `"appendix_presets"`) or routes a selected table through `apa_table()` and `plot.apa_table()` for numeric QC.

Value

A plotting-data object of class `mfrm_plot_data`.

Interpreting output

- `"table_rows"`: compares returned table sizes to show where reporting mass sits.
- `"role_tables"`: shows how many returned tables belong to each reporting role.
- `"appendix_roles"`: shows how returned tables contribute to conservative appendix routing by reporting role.
- `"appendix_sections"`: shows how returned tables are distributed across methods/results/diagnostics/reporting sections.
- `"appendix_presets"`: shows how many tables the current bundle contributes to the conservative appendix presets.
- `"selection_handoff_presets"`: shows plot-ready appendix handoff counts by preset for workflow-only appendix routing surfaces in the bundle.
- `"selection_tables"` / `"selection_handoff"` / `"selection_handoff_bundles"` / `"selection_handoff_roles"` / `"selection_handoff_role_sections"` / `"selection_bundles"` / `"selection_roles"` / `"selection_sections"`: show workflow-only appendix selection surfaces already materialized inside the bundle.
- `"numeric_profile"` / `"first_numeric"`: reuse the same numeric QC logic as `plot.apa_table()` but start from a summary-table bundle.

See Also

`build_summary_table_bundle()`, `apa_table()`, `plot.apa_table()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
bundle <- build_summary_table_bundle(fit)
plot(bundle, draw = FALSE)
plot(bundle, type = "numeric_profile", which = "facet_overview", draw = FALSE)
```

---

plot_anchor_drift *Plot anchor drift or a screened linking chain*

Description

Creates base-R plots for inspecting anchor drift across calibration waves or visualising the cumulative offset in a screened linking chain.

Usage

```
plot_anchor_drift(
  x,
  type = c("drift", "chain", "heatmap", "forest"),
  facet = NULL,
  ci_level = 0.95,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ...
)
```

Arguments

x	An mfrm_anchor_drift or mfrm_equating_chain object.
type	Plot type: "drift" (dot plot of element drift), "chain" (cumulative offset line plot), "heatmap" (wave-by-element drift heatmap), or "forest" (per-(Facet, Level, Wave) anchor estimate with +/- z * SE whiskers; requires mfrm_anchor_drift).
facet	Optional character vector to filter drift plots to specific facets.
ci_level	Confidence level used by type = "forest" for the anchor-estimate whiskers (default 0.95). Ignored for other plot types.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If FALSE, return the plot data invisibly without drawing.
...	Additional graphical parameters passed to base plotting functions.

Details

Three plot types are supported:

- `"drift"` (for `mfrm_anchor_drift` objects): A dot plot of each element's drift value, grouped by facet. Horizontal reference lines mark the drift threshold. Red points indicate flagged elements.
- `"heatmap"` (for `mfrm_anchor_drift` objects): A wave-by-element heat matrix showing drift magnitude. Darker cells represent larger absolute drift. Useful for spotting systematic patterns (e.g., all criteria shifting in the same direction).
- `"chain"` (for `mfrm_equating_chain` objects): A line plot of cumulative offsets across the screened linking chain. A flatter line indicates smaller between-wave shifts; steep segments suggest larger link offsets that deserve review.

Value

A plotting-data object of class `mfrm_plot_data`. With `draw = FALSE`, `result$data$table` contains the filtered drift or chain table, `result$data$matrix` contains the heatmap matrix when requested, and the returned plot data includes package-native `title`, `subtitle`, `legend`, and `reference_lines`.

Which plot should I use?

- Use `type = "drift"` with an `mfrm_anchor_drift` object to review flagged elements directly.
- Use `type = "heatmap"` with an `mfrm_anchor_drift` object to spot wave-by-element patterns.
- Use `type = "chain"` with an `mfrm_equating_chain` object after `build_equating_chain()` to inspect cumulative offsets across waves.

Interpreting plots

Drift is the change in an element's estimated measure between calibration waves, after accounting for the screened common-element link offset. An element is flagged when its absolute drift exceeds a threshold (typically 0.5 logits) **and** the drift-to-SE ratio exceeds a secondary criterion (typically 2.0), ensuring that only practically noticeable and relatively precise shifts are flagged.

- In drift and heatmap plots, red or dark-shaded elements exceed both thresholds. Common causes include rater drift over time, item exposure effects, or curriculum changes.
- In chain plots, uneven spacing between waves suggests differential shifts in the screened linking offsets. The y-axis shows cumulative logit-scale offsets; flatter segments indicate more stable adjacent links. Steep segments should be checked alongside `LinkSupportAdequate` and the retained common-element counts before making longitudinal claims.
- For drift objects, it is usually best to read `summary(x)` first and then use the plot to see where the flagged values sit.

Typical workflow

1. Build a drift or screened-linking object with `detect_anchor_drift()` or `build_equating_chain()`.
2. Start with `draw = FALSE` if you want the plotting data for custom reporting.
3. Use the base-R plot for quick screening and then inspect the underlying tables for exact values.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`detect_anchor_drift()`, `build_equating_chain()`, `plot_dif_heatmap()`, `plot_bubble()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
people <- unique(toy$Person)
d1 <- toy[toy$Person %in% people[1:12], , drop = FALSE]
d2 <- toy[toy$Person %in% people[13:24], , drop = FALSE]
fit1 <- fit_mfrm(d1, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
fit2 <- fit_mfrm(d2, "Person", c("Rater", "Criterion"), "Score",
                 method = "JML", maxit = 30)
drift <- detect_anchor_drift(list(W1 = fit1, W2 = fit2))
drift_plot <- plot_anchor_drift(drift, type = "drift", draw = FALSE)
class(drift_plot)
names(drift_plot$data)
chain <- build_equating_chain(list(F1 = fit1, F2 = fit2))
chain_plot <- plot_anchor_drift(chain, type = "chain", draw = FALSE)
head(chain_plot$data$table)
if (interactive()) {
  plot_anchor_drift(drift, type = "heatmap", preset = "publication")
}
```

---

plot_apa_figure_one *Manuscript-ready four-panel composite (Wright + severity + threshold + summary)*

Description

Builds a 2x2 publication composite for an `mfrm_fit`, suitable for a "Figure 1" in the Rasch-family RSM/PCM manuscript route. Panels: (1) Wright map, (2) rater severity profile with CI whiskers, (3) threshold ladder, (4) a one-line reliability / separation summary block. Each panel reuses the standalone plot helper so the visual language is consistent with the rest of the package.

Usage

```
plot_apa_figure_one(
  fit,
  diagnostics = NULL,
  rater_facet = "Rater",
  ci_level = 0.95,
```

```
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output.
rater_facet	Facet name to use as the "rater" axis (default "Rater").
ci_level	Confidence level for the rater severity panel.
preset	Visual preset.
draw	If TRUE, draw the composite immediately with graphics::layout().

Value

Invisibly, an mfrm_plot_data object whose data slot bundles the four panel data objects under wright, severity, threshold, and summary.

Interpreting output

Designed for a single-figure Methods or Results overview. The summary panel prints the model class, sample size, log-likelihood, AIC/BIC, and the largest non-Person facet's separation / reliability if available.

See Also

plot.mfrm_fit() (type = "wright"), plot_rater_severity_profile(), plot_threshold_ladder(), build_apa_outputs().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_apa_figure_one(fit, draw = FALSE)
names(p$data)
```

---

plot_bias_interaction *Plot bias interaction diagnostics (preferred alias)*

Description

Plot bias interaction diagnostics (preferred alias)

Usage

```
plot_bias_interaction(
  x,
  plot = c("scatter", "ranked", "heatmap", "abs_t_hist", "facet_profile"),
  diagnostics = NULL,
  facet_a = NULL,
  facet_b = NULL,
  interaction_facets = NULL,
  top_n = 40,
  abs_t_warn = 2,
  abs_bias_warn = 0.5,
  p_max = 0.05,
  sort_by = c("abs_t", "abs_bias", "prob"),
  show_ci = FALSE,
  ci_level = 0.95,
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

x	Output from estimate_bias() or fit_mfrm().
plot	Plot type: "scatter", "ranked", "heatmap", "abs_t_hist", or "facet_profile".
diagnostics	Optional output from diagnose_mfrm() (used when x is fit).
facet_a	First facet name (required when x is fit and interaction_facets is not supplied).
facet_b	Second facet name (required when x is fit and interaction_facets is not supplied).
interaction_facets	Character vector of two or more facets.
top_n	Maximum number of ranked rows to keep.
abs_t_warn	Warning cutoff for absolute t statistics.
abs_bias_warn	Warning cutoff for absolute bias size.
p_max	Warning cutoff for p-values.
sort_by	Ranking key: "abs_t", "abs_bias", or "prob".
show_ci	Logical. When TRUE and plot is "scatter" or "ranked", draw confidence-interval whiskers for Bias Size. Bounded GPCM rows use the conditional profile-likelihood limits returned by estimate_bias() when available; otherwise the interval uses the per-cell standard error from estimate_bias(). Ignored for "heatmap", "abs_t_hist", and "facet_profile".
ci_level	Confidence level used when show_ci = TRUE; default 0.95. The returned plot-data object gains CI_Lower / CI_Upper / CI_Level columns on the ranked_table and scatter_data elements for downstream reuse.

main	Optional plot title override.
palette	Optional named color overrides (normal, flag, hist, profile).
label_angle	Label angle hint for ranked/profile labels.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.

Details

Visualization front-end for bias_interaction_report() with multiple views. With draw = FALSE, the returned plot data include plot_long, plot_annotations, flag_summary, and plot_settings in addition to the view-specific ranked_table, scatter_data, facet_profile, and heatmap components. Use these fields when rebuilding the same screening view in ggplot2, plotly, Quarto, or a dashboard.

Value

A plotting-data object of class mfrm_plot_data.

Plot types

"scatter" **(default)** Scatter plot of bias size (x) vs screening t-statistic (y). Points colored by flag status. Dashed reference lines at abs_bias_warn and abs_t_warn. Use for overall triage of interaction effects.

"ranked" Ranked bar chart of top top_n interactions sorted by sort_by criterion (absolute t, absolute bias, or probability). Bars colored red for flagged cells.

"heatmap" Facet A by facet B matrix of signed bias size. Cells retain reusable matrix and flag tables for dashboards. This is a Table 13 follow-up display: it supports pattern recognition but does not turn screening rows into confirmatory tests.

"abs_t_hist" Histogram of absolute screening t-statistics across all interaction cells. Dashed reference line at abs_t_warn. Use for assessing the overall distribution of interaction effect sizes.

"facet_profile" Per-facet-level aggregation showing mean absolute bias and flag rate. Useful for identifying which individual facet levels drive systematic interaction patterns.

Interpreting output

Start with "scatter" or "ranked" for triage, then confirm pattern shape using "abs_t_hist" and "facet_profile".

Consistent flags across multiple views are stronger screening signals of systematic interaction bias than a single extreme row, but they do not by themselves establish formal inferential evidence.

Typical workflow

1. Estimate bias with estimate_bias() or pass mfrm_fit directly.
2. Plot with plot = "ranked" for top interactions.
3. Cross-check using plot = "scatter" and plot = "facet_profile".

See Also

bias_interaction_report(), estimate_bias(), plot_displacement()

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
p <- plot_bias_interaction(
  fit,
  diagnostics = diagnose_mfrm(fit, residual_pca = "none"),
  facet_a = "Rater",
  facet_b = "Criterion",
  preset = "publication",
  draw = FALSE
)
```

---

plot_bubble *Bubble chart of measure estimates and fit statistics*

Description

Produces a Rasch-convention bubble chart where each element is a circle positioned at its measure estimate (x) and fit mean-square (y). Bubble radius reflects approximate measurement precision or sample size.

Usage

```
plot_bubble(
  x,
  diagnostics = NULL,
  fit_stat = c("Infit", "Outfit"),
  view = c("measure", "infit_outfit"),
  bubble_size = NULL,
  facets = NULL,
  fit_range = c(0.5, 1.5),
  top_n = 60,
  main = NULL,
  palette = NULL,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome")
)
```

Arguments

x	Output from fit_mfrm or diagnose_mfrm.
diagnostics	Optional output from diagnose_mfrm when x is an mfrm_fit object. If omitted, diagnostics are computed automatically.

fit_stat	Fit statistic for the y-axis: "Infit" (default) or "Outfit". Ignored when view = "infit_outfit" because that view always plots Infit on x and Outfit on y.
view	Layout. "measure" (default, the historical mfrmr layout) plots Measure (logit) on x and the chosen fit_stat MnSq on y. "infit_outfit" plots Infit MnSq on x and Outfit MnSq on y, matching the Winsteps Table 30.2 "Most-misfitting Persons / Items" scatter that many MFRM and Rasch users expect, and defaults bubble_size = "N".
bubble_size	Variable controlling bubble radius: "SE" (default for view = "measure"), "N" (observation count; default for view = "infit_outfit"), or "equal" (uniform size).
facets	Character vector of facets to include. NULL (default) includes all non-person facets.
fit_range	Numeric length-2 vector defining the heuristic fit-review band shown as a shaded region (default c(0.5, 1.5)).
top_n	Maximum number of elements to plot (default 60).
main	Optional custom plot title.
palette	Optional named colour vector keyed by facet name.
draw	If TRUE (default), render the plot using base graphics.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").

Details

When x is an mfrm_fit object and diagnostics is omitted, the function computes diagnostics internally via diagnose_mfrm(). For repeated plotting in the same workflow, passing a precomputed diagnostics object avoids that extra work.

The x-axis shows element measure estimates on the **logit** scale (one logit = one unit change in log-odds of responding in a higher category). The y-axis shows the selected fit mean-square statistic. A shaded band between fit_range[1] and fit_range[2] highlights a common heuristic review range.

Bubble radius options:

- "SE": inversely proportional to standard error—larger circles indicate more precisely estimated elements under the current SE approximation.
- "N": proportional to observation count—larger circles indicate elements with more data.
- "equal": uniform size, useful when SE or N differences distract from the fit pattern.

Person estimates are excluded by default because they typically outnumber facet elements and obscure the display.

Value

Invisibly, an object of class mfrm_plot_data.

Interpreting the plot

Points near the horizontal reference line at 1.0 are closer to model expectation on the selected MnSq scale. Points above 1.5 suggest underfit relative to common review heuristics; these elements may have inconsistent scoring. Points below 0.5 suggest overfit relative to common review heuristics; these may indicate redundancy or restricted range. Points are colored by facet for easy identification.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Compute diagnostics once with `diagnose_mfrm()`.
3. Call `plot_bubble(fit, diagnostics = diag)` to inspect the most extreme elements.

See Also

`diagnose_mfrm`, `plot_unexpected`, `plot_fair_average`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
p <- plot_bubble(fit, diagnostics = diag, draw = FALSE)
head(p$data$table[, c("Facet", "Level", "Estimate", "Infit", "Outfit")])
# Look for (default `view = "measure"`): bubbles inside the shaded
#   0.5-1.5 fit-review band. Bubbles above the band are underfit
#   (noisy elements); below the band are overfit (overly predictable).
#
# For the Winsteps Table 30 layout pass `view = "infit_outfit"`:
p_io <- plot_bubble(fit, diagnostics = diag, view = "infit_outfit",
                    draw = FALSE)
p_io$data$view
# Look for: bubbles clustered inside the central [0.5, 1.5] x [0.5, 1.5]
#   square. Points outside the upper-right corner have both Infit
#   AND Outfit > 1.5 (consistent underfit); points outside the
#   lower-left have both < 0.5 (consistent overfit). Bubble size in
#   this view defaults to N (observation count) so the visual
#   weighting matches how seriously the misfit should be taken.
```

---

plot_data *Extract reusable data from an mfrmr plot object*

Description

`plot_data()` is a small accessor for users who want to build custom base-R, ggplot2, plotly, or table-based displays from mfrmr plot helpers. It accepts an existing `mfrm_plot_data` object, or any mfrmr object whose `plot()` method supports `draw = FALSE`. Use `plot_data_components()` first when you want to inspect which components are available before extracting one.

Usage

```
plot_data(x, component = NULL, type = NULL, ...)
```

Arguments

x	An mfrm_plot_data object, or a fitted/report/review object with a plot(..., draw = FALSE) method.
component	Optional single component name inside the reusable plot data. When NULL, the full plot-data list is returned.
type	Optional plot type passed to plot() when x is not already an mfrm_plot_data object.
...	Additional arguments passed to plot(..., draw = FALSE) when x is not already an mfrm_plot_data object.

Value

The full reusable plot-data list, or the selected component.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", maxit = 30)

wright_plot_data <- plot_data(fit, type = "wright")
names(wright_plot_data)

wright_table <- plot_data(fit, type = "wright", component = "locations")
head(wright_table)

curves <- category_curves_report(fit, theta_points = 51)
curve_long <- plot_data(curves, component = "plot_long")
head(curve_long[, c("PlotType", "Theta", "Series", "Value")])

pathway_long <- plot_data(fit, type = "pathway", component = "pathway_long")
head(pathway_long[, c("Layer", "CurveGroup", "Theta", "Value")])
pathway_fit <- plot_data(fit, type = "pathway", component = "fit_measures")
head(pathway_fit[, c("Facet", "Level", "Infit", "Outfit", "FitStatus")])

info <- compute_information(fit, theta_points = 51)
sem_long <- plot_data(
  plot_information(info, type = "sem", draw = FALSE),
  component = "plot_long"
)
head(sem_long[, c("Metric", "Theta", "Value", "DisplayedByDefault")])
```

plot_data_components *List reusable components in mfrmr plot data*

Description

plot_data_components() is a companion to plot_data(). It returns a compact table that tells users which plot-data components are available, what shape they have, and which ones are most useful for custom graphics, dashboards, or report assembly.

Usage

```
plot_data_components(x, type = NULL, ...)
```

Arguments

x	An mfrm_plot_data object, or a fitted/report/review object with a plot(..., draw = FALSE) method.
type	Optional plot type passed to plot() when x is not already an mfrm_plot_data object.
...	Additional arguments passed to plot(..., draw = FALSE) when x is not already an mfrm_plot_data object.

Value

A data frame with one row per reusable plot-data component.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", maxit = 30)
plot_data_components(fit, type = "pathway")

curves <- category_curves_report(fit, theta_points = 51)
plot_data_components(curves, type = "category_probability")

toy$ResponseTime <- 10 + seq_len(nrow(toy)) %% 6 + as.numeric(toy$Score)
rt <- response_time_review(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  time = "ResponseTime"
)
plot_data_components(plot_response_time_review(rt, draw = FALSE))
```

plot_dif_heatmap *Plot a differential-functioning heatmap*

Description

Visualizes the interaction between a facet and a grouping variable as a heatmap. Rows represent facet levels, columns represent group values, and cell color indicates the selected metric.

Usage

```
plot_dif_heatmap(
  x,
  metric = c("obs_exp", "t", "contrast"),
  draw = TRUE,
  show_values = TRUE,
  value_digits = 2L,
  flag_threshold = NULL,
  scale_limit = NULL,
  flag_color = "black",
  ...
)
```

Arguments

x	Output from dif_interaction_table(), analyze_dff(), or analyze_dif(). When an mfrm_dff/mfrm_dif object is passed, the cell_table element is used (requires method = "residual").
metric	Which metric to plot: "obs_exp" for observed-minus-expected average (default), "t" for the standardized residual / t-statistic, or "contrast" for pairwise differential-functioning contrast (only for mfrm_dff objects with dif_table).
draw	If TRUE (default), draw the plot.
show_values	Logical. If TRUE (default), print rounded cell values on top of the heatmap.
value_digits	Non-negative integer number of digits after the decimal point for cell labels.
flag_threshold	Optional non-negative absolute-value threshold. When supplied, cells with abs(value) >= flag_threshold are recorded in $data$flag_matrix and outlined on the drawn heatmap.
scale_limit	Optional positive scalar for a symmetric color scale from -scale_limit to +scale_limit. Use this to make several heatmaps visually comparable.
flag_color	Border color for cells meeting flag_threshold.
...	Additional graphical parameters passed to graphics::image().

Value

Invisibly, an mfrm_plot_data object whose data slot bundles the row x column metric matrix ($matrix), the source long table ($pairs), and the metric label. Earlier 0.1.x releases returned the bare matrix; consume $data$matrix to keep code forward-compatible.

Interpreting output

- Warm colors (red) indicate positive Obs-Exp values (the model underestimates the facet level for that group).
- Cool colors (blue) indicate negative Obs-Exp values (the model overestimates).
- White/neutral indicates no systematic difference.
- The `"contrast"` view is best for pairwise differential-functioning summaries, whereas `"obs_exp"` and `"t"` are best for cell-level diagnostics.

Typical workflow

1. Compute interaction with `dif_interaction_table()` or differential- functioning contrasts with `analyze_dff()`.
2. Plot with `plot_dif_heatmap(...)`.
3. Identify extreme cells or contrasts for follow-up.

See Also

`dif_interaction_table()`, `analyze_dff()`, `analyze_dif()`, `dif_report()`

Examples

```
toy <- load_mfrmr_data("example_bias")

fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
int <- dif_interaction_table(fit, diag, facet = "Rater",
                             group = "Group", data = toy, min_obs = 2)
heat <- plot_dif_heatmap(int, metric = "obs_exp", draw = FALSE)
dim(heat$data$matrix)
# Look for (`metric = "obs_exp"`): cells near 0 are aligned with
#   model expectation; |Obs - Exp| > 0.5 logits is a substantive
#   gap. With `metric = "t"` the cell scale becomes a standardized
#   residual where |t| > 2 is a screening flag, not a standalone
#   hypothesis test. With `metric = "contrast"` the layout switches
#   to Level x GroupPair and reads as the pairwise differential-
#   functioning contrast (use `analyze_dff()`).
```

---

plot_dif_summary *Summary plot of differential functioning effect sizes*

Description

Compact effect-size summary for a `analyze_dff()` / `analyze_dif()` result. Shows each contrast's signed effect size as a horizontal bar with a vertical reference at zero, coloured by the method-appropriate classification. ETS-style A / B / C colours are used only when they are actually available; residual-method screening labels otherwise use the neutral colour.

Usage

```
plot_dif_summary(
  x,
  top_n = 30L,
  sort_by = c("abs_effect", "effect", "classification"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ci_level = NULL,
  effect_thresholds = NULL,
  effect_axis_label = NULL
)
```

Arguments

x	Output from analyze_dff() or analyze_dif().
top_n	Maximum rows shown (default 30).
sort_by	"abs_effect" (default), "effect", or "classification".
preset	Visual preset.
draw	If TRUE, draw with base graphics.
ci_level	Optional confidence level for approximate normal intervals drawn from Effect +/- z * SE when finite standard errors are available. Use NULL (default) to omit intervals.
effect_thresholds	Optional numeric vector of absolute effect-size guide lines to draw at +/- threshold. These are display aids; only use ETS-like values when the source rows support ETS interpretation.
effect_axis_label	Optional x-axis label override. When NULL, the label is chosen from the DFF method.

Value

An mfrm_plot_data object whose data slot contains columns Pair, Effect, SE, Classification, Color.

Interpreting output

Bars are anchored at zero. Width corresponds to effect size on the contrast's native scale. For method = "residual", this is the observed-minus-expected average screening contrast between groups. For method = "refit", this is the subgroup parameter difference on the fitted logit scale when linking support allows a comparable contrast. The ETS classification (A negligible, B moderate, C large) drives bar colour only when ClassificationSystem == "ETS"; otherwise the bar uses the preset's neutral.

See Also

analyze_dff(), analyze_dif(), plot_dif_heatmap().

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
dff <- analyze_dff(fit, diagnostics = diag,
                   facet = "Rater", group = "Group", data = toy)
unique(dff$dif_table$ClassificationSystem)
p <- plot_dif_summary(dff, draw = FALSE)
head(p$data$data)
```

`plot_displacement` *Plot displacement diagnostics using base R*

Description

Plot displacement diagnostics using base R

Usage

```
plot_displacement(
  x,
  diagnostics = NULL,
  anchored_only = FALSE,
  facets = NULL,
  plot_type = c("lollipop", "hist"),
  top_n = 40,
  show_ci = FALSE,
  ci_level = 0.95,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ...
)
```

Arguments

`x`	Output from `fit_mfrm()` or `displacement_table()`.
`diagnostics`	Optional output from `diagnose_mfrm()` when `x` is `mfrm_fit`.
`anchored_only`	Keep only anchored/group-anchored levels.
`facets`	Optional subset of facets.
`plot_type`	`"lollipop"` or `"hist"`.
`top_n`	Maximum levels shown in `"lollipop"` mode.
`show_ci`	Logical. When `TRUE` and `plot_type = "lollipop"`, draw approximate confidence-interval whiskers from `DisplacementSE` (ignored for `"hist"`).

ci_level	Confidence level used when show_ci = TRUE; default 0.95. The returned plot-data object gains CI_Lower / CI_Upper / CI_Level columns on the table element for downstream reuse.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.
...	Additional arguments passed to displacement_table() when x is mfrm_fit.

Details

Displacement quantifies how much a single element's calibration would shift the overall model if it were allowed to move freely. It is computed as:

$$\text{Displacement}_j = \frac{\sum_i (X_{ij} - E_{ij})}{\sum_i \text{Var}_{ij}}$$

where the sums run over all observations involving element j. The standard error is $1/\sqrt{\sum_i \text{Var}_{ij}}$, and a t-statistic $t = \text{Displacement}/\text{SE}$ flags elements whose observed residual pattern is inconsistent with the current anchor structure.

Displacement is most informative after anchoring: large values suggest that anchored values may be drifting from the current sample. For non-anchored analyses, displacement reflects residual calibration tension.

Value

A plotting-data object of class mfrm_plot_data.

Plot types

"lollipop" **(default)** Dot-and-line chart of displacement values. X-axis: displacement (logits). Y-axis: element labels. Points colored red when flagged (default: $|\text{Disp.}| > 0.5$ logits). Dashed lines at $\pm$ threshold. Ordered by absolute displacement.

"hist" Histogram of displacement values with Freedman-Diaconis breaks. Dashed reference lines at $\pm$ threshold. Use for inspecting the overall distribution shape.

Interpreting output

Lollipop: top absolute displacement levels; flagged points indicate larger movement from anchor expectations.

Histogram: overall displacement distribution and threshold lines. A symmetric distribution centred near zero indicates good anchor stability; heavy tails or skew suggest systematic drift.

Use anchored_only = TRUE when your main question is anchor robustness.

Typical workflow

1. Run with plot_type = "lollipop" and anchored_only = TRUE.
2. Inspect distribution with plot_type = "hist".
3. Drill into flagged rows via displacement_table().

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`displacement_table()`, `plot_unexpected()`, `plot_fair_average()`, `plot_qc_dashboard()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
p <- plot_displacement(fit, anchored_only = FALSE, draw = FALSE)
if (interactive()) {
  plot_displacement(
    fit,
    anchored_only = FALSE,
    plot_type = "lollipop",
    preset = "publication"
  )
}
```

---

plot_facets_chisq *Plot facet variability diagnostics using base R*

Description

Plot facet variability diagnostics using base R

Usage

```
plot_facets_chisq(
  x,
  diagnostics = NULL,
  fixed_p_max = 0.05,
  random_p_max = 0.05,
  plot_type = c("fixed", "random", "variance"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

x	Output from fit_mfrm() or facets_chisq_table().
diagnostics	Optional output from diagnose_mfrm() when x is mfrm_fit.
fixed_p_max	Warning cutoff for fixed-effect chi-square p-values.
random_p_max	Warning cutoff for random-effect chi-square p-values.
plot_type	"fixed", "random", or "variance".
main	Optional custom plot title.
palette	Optional named color overrides (fixed_ok, fixed_flag, random_ok, random_flag, variance).
label_angle	X-axis label angle for bar-style plots.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.

Details

Facet chi-square tests assess whether the elements within each facet differ significantly.

Fixed-effect chi-square tests the null hypothesis $H_0 : \delta_1 = \delta_2 = \cdots = \delta_J$ (all element measures are equal). A flagged result ($p <$ fixed_p_max) suggests detectable between-element spread under the fitted model, but it should be interpreted alongside design quality, sample size, and other diagnostics.

Random-effect chi-square tests whether element heterogeneity exceeds what would be expected from measurement error alone, treating element measures as random draws. A flagged result is screening evidence that the facet may not be exchangeable under the current model.

Random variance is the estimated between-element variance component after removing measurement error. It quantifies the magnitude of true heterogeneity on the logit scale.

Value

A plotting-data object of class mfrm_plot_data.

Plot types

"fixed" **(default)** Bar chart of fixed-effect chi-square by facet. Bars colored red when the null hypothesis is rejected at fixed_p_max. A flagged (red) bar means the facet shows spread worth reviewing under the fitted model.

"random" Bar chart of random-effect chi-square by facet. Bars colored red when rejected at random_p_max.

"variance" Bar chart of estimated random variance (logit^2) by facet. Reference line at 0. Larger values indicate greater true heterogeneity among elements.

Interpreting output

Colored flags reflect configured p-value thresholds (fixed_p_max, random_p_max). For the fixed test, a flagged (red) result suggests facet spread worth reviewing under the current model. For the random test, a flagged result is screening evidence that the facet may contribute non-trivial heterogeneity beyond measurement error.

Typical workflow

1. Review "fixed" and "random" panels for flagged facets.
2. Check "variance" to contextualize heterogeneity.
3. Cross-check with inter-rater and element-level fit diagnostics.

See Also

facets_chisq_table(), plot_interrater_agreement(), plot_qc_dashboard()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
p <- plot_facets_chisq(fit, draw = FALSE)
if (interactive()) {
  plot_facets_chisq(
    fit,
    draw = TRUE,
    plot_type = "fixed",
    preset = "publication",
    main = "Facet Chi-square (Customized)",
    palette = c(fixed_ok = "#2b8cbe", fixed_flag = "#cb181d"),
    label_angle = 45
  )
}
```

---

plot_facet_equivalence

Plot facet-equivalence results

Description

Plot facet-equivalence results

Usage

```
plot_facet_equivalence(
  x,
  diagnostics = NULL,
  facet = NULL,
  type = c("forest", "rope"),
  draw = TRUE,
  ...
)
```

Arguments

x	Output from analyze_facet_equivalence() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() when x is an mfrm_fit object.
facet	Facet to analyze when x is an mfrm_fit object.
type	Plot type: "forest" (default) or "rope".
draw	If TRUE (default), draw the plot. If FALSE, return the prepared plotting data.
...	Additional graphical arguments passed to base plotting functions.

Details

plot_facet_equivalence() is a visual companion to analyze_facet_equivalence(). It does not recompute the equivalence analysis; it only reshapes and displays the returned results.

Value

Invisibly returns the plotting data. If draw = FALSE, the plotting data are returned without drawing.

Plot types

- "forest" places each level on the logit scale with its confidence interval and shades the practical-equivalence region around the weighted grand mean.
- "rope" shows the percentage of each level's uncertainty mass that falls inside the ROPE.

Interpreting output

In the **forest plot**, the shaded band marks the ROPE (±equivalence_bound around the weighted grand mean). Levels whose entire confidence interval lies inside this band are close to the facet grand mean under this descriptive screen. Levels whose interval extends outside the band are more displaced from the facet average. Overlapping intervals between two elements suggest they are not reliably separable, but overlap alone does not establish formal equivalence—use the TOST results for that.

In the **ROPE bar chart**, each bar shows the proportion of the element's normal-approximation distribution that falls inside the ROPE-style grand-mean proximity. Values > 95\ the element's normal-approximation uncertainty falls near the facet average; 50–95\ meaningfully displaced from that average.

Typical workflow

1. Run analyze_facet_equivalence().
2. Start with type = "forest" to see the facet on the logit scale.
3. Switch to type = "rope" when you want a ranking of levels by grand-mean proximity.

See Also

analyze_facet_equivalence()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
eq <- analyze_facet_equivalence(fit, facet = "Rater")
pdat <- plot_facet_equivalence(eq, type = "forest", draw = FALSE)
c(pdat$facet, pdat$type)
```

---

plot_facet_quality_dashboard

Plot a facet-quality dashboard

Description

Plot a facet-quality dashboard

Usage

```
plot_facet_quality_dashboard(
  x,
  diagnostics = NULL,
  facet = NULL,
  bias_results = NULL,
  severity_warn = 1,
  misfit_warn = 1.5,
  central_tendency_max = 0.25,
  bias_count_warn = 1L,
  bias_abs_t_warn = 2,
  bias_abs_size_warn = 0.5,
  bias_p_max = 0.05,
  plot_type = c("severity", "flags"),
  top_n = 20,
  main = NULL,
  palette = NULL,
  label_angle = 45,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from facet_quality_dashboard() or fit_mfrm().
diagnostics	Optional output from diagnose_mfrm() when x is a fit.
facet	Optional facet name.
bias_results	Optional bias bundle or list of bundles.
severity_warn	Absolute estimate cutoff used to flag severity outliers.

misfit_warn	Mean-square cutoff used to flag misfit.
central_tendency_max	Absolute estimate cutoff used to flag central tendency.
bias_count_warn	Minimum flagged-bias row count required to flag a level.
bias_abs_t_warn	Absolute t cutoff used when deriving bias-row flags from a raw bias bundle.
bias_abs_size_warn	Absolute bias-size cutoff used when deriving bias-row flags from a raw bias bundle.
bias_p_max	Probability cutoff used when deriving bias-row flags from a raw bias bundle.
plot_type	Plot type, "severity" or "flags".
top_n	Number of rows to keep in the plot data.
main	Optional plot title.
palette	Optional named color overrides.
label_angle	Label angle hint for the "flags" plot.
draw	If TRUE, draw with base graphics.
...	Reserved for generic compatibility.

Value

A plotting-data object of class mfrm_plot_data.

See Also

facet_quality_dashboard(), summary.mfrm_facet_dashboard()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
p <- plot_facet_quality_dashboard(fit, diagnostics = diag, draw = FALSE)
p$data$plot
```

---

plot_fair_average *Plot fair-average diagnostics using base R*

Description

Plot fair-average diagnostics using base R

Usage

```
plot_fair_average(
  x,
  diagnostics = NULL,
  facet = NULL,
 metric = c("AdjustedAverage", "StandardizedAdjustedAverage", "FairM", "FairZ"),
  plot_type = c("difference", "scatter"),
  top_n = 40,
  show_ci = FALSE,
  ci_level = 0.95,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  ...
)
```

Arguments

x	Output from fit_mfrm() or fair_average_table().
diagnostics	Optional output from diagnose_mfrm() when x is mfrm_fit.
facet	Optional facet name for level-wise lollipop plots.
metric	Adjusted-score metric. Accepts legacy names ("FairM", "FairZ") and package-native names ("AdjustedAverage", "StandardizedAdjustedAverage").
plot_type	"difference" or "scatter".
top_n	Maximum levels shown for "difference" plot.
show_ci	Logical. When TRUE, draw approximate confidence-interval whiskers on the fair metric using a delta-method propagation from the logit Measure standard error to the observed-score scale. The derivative equals the implied score variance Var(X | Measure), so the fair-scale standard error is Var(X) * ModelSE. CI bounds are clipped to the rating range. Rows where the score variance is effectively zero (levels whose measure sits near the rating boundary, so the delta-method approximation becomes uninformative) are drawn with an open circle and excluded from the whiskers; the excluded count is reported in the subtitle. For bounded GPCM fits, this option requests fair_average_table(fair_se = TRUE) when x is a fit object and uses the structural delta-method fair-average CI columns when they are available. If x is a precomputed fair-average bundle without those columns, the plot records an unavailable-CI note.
ci_level	Confidence level used when show_ci = TRUE; default 0.95. The returned plot-data object gains CI_Lower, CI_Upper, and CI_Level columns for downstream reuse.
draw	If TRUE, draw with base graphics.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
...	Additional arguments passed to fair_average_table() when x is mfrm_fit.

Details

Fair-average plots compare observed scoring tendency against model-based fair metrics.

FairM is the model-predicted mean score for each element, adjusting for the ability distribution of persons actually encountered. It answers: "What average score would this rater/criterion produce if all raters/criteria saw the same mix of persons?"

FairZ standardises FairM to a z-score across elements within each facet, making it easier to compare relative severity across facets with different raw-score scales.

Use FairM when the raw-score metric is meaningful (e.g., reporting average ratings on the original 1–4 scale). Use FairZ when comparing standardised severity ranks across facets.

Value

A plotting-data object of class `mfrm_plot_data`. With `draw = FALSE`, the returned plot data includes `title`, `subtitle`, `legend`, `reference_lines`, and the stacked fair-average data.

Plot types

`"difference"` **(default)** Lollipop chart showing the gap between observed and fair-average score for each element. X-axis: Observed - Fair metric. Y-axis: element labels. Points colored teal (lenient, gap >= 0) or orange (severe, gap < 0). Ordered by absolute gap.

`"scatter"` Scatter plot of fair metric (x) vs observed average (y) with an identity line. Points colored by facet. Useful for checking overall alignment between observed and model-adjusted scores.

Interpreting output

Difference plot: ranked element-level gaps (`Observed - Fair`), useful for triage of potentially lenient/severe levels.

Scatter plot: global agreement pattern relative to the identity line.

Larger absolute gaps suggest stronger divergence between observed and model-adjusted scoring.

Typical workflow

1. Start with `plot_type = "difference"` to find largest discrepancies.
2. Use `plot_type = "scatter"` to check overall alignment pattern.
3. Follow up with facet-level diagnostics for flagged levels.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`fair_average_table()`, `plot_unexpected()`, `plot_displacement()`, `plot_qc_dashboard()`, mfrmr_visual_diagnostics

Examples

```
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
p <- plot_fair_average(fit, metric = "AdjustedAverage", draw = FALSE)
if (interactive()) {
  plot_fair_average(fit, metric = "AdjustedAverage", plot_type = "difference")
}
```

---

plot_guttman_scalogram

Guttman-style scalogram of person x item observed responses

Description

Draws a person x item (or person x facet-level) matrix coloured by observed category, with rows ordered by person measure and columns ordered by location measure. Unexpected responses (those that fall far from the expected category at a given theta) are highlighted with a heavy border so the visual reads as a Rasch-convention Guttman scalogram.

Usage

```
plot_guttman_scalogram(
  fit,
  diagnostics = NULL,
  column_facet = NULL,
  top_n_persons = 40L,
  highlight_unexpected = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`diagnostics`	Optional `diagnose_mfrm()` output; used to pick up unexpected-response flags when available.
`column_facet`	Facet name used for the columns. Default `"Criterion"` when the fit contains it, otherwise the last entry of `fit$config$facet_names`.
`top_n_persons`	Maximum number of persons shown (default `40`). Persons closest to the median measure are retained when the population exceeds this cap.
`highlight_unexpected`	Logical. When `TRUE` (default), draw a heavy border around cells flagged as unexpected by `unexpected_response_table()`.

preset	Visual preset.
draw	If TRUE, draw with base graphics.

Value

An mfrm_plot_data object whose data slot bundles the scalogram matrix and the optional unexpected-response overlay.

See Also

unexpected_response_table() for the case-level review of the cells flagged in the overlay; plot_rater_agreement_heatr for a complementary rater-pair view of the same residual structure; diagnose_mfrm() for the underlying diagnostics bundle.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_guttman_scalogram(fit, draw = FALSE)
dim(p$data$matrix)
# Look for: a clean monotone "staircase" of higher scores in the
#   upper-right triangle and lower scores in the lower-left, once
#   rows are sorted by person ability. Cells circled by the
#   unexpected-response overlay break the staircase and warrant
#   case-level review with `unexpected_response_table()`.
```

---

plot_information *Plot design-weighted precision curves*

Description

Visualize the design-weighted precision curve and optionally per-facet-level contribution curves from compute_information().

Usage

```
plot_information(
  x,
  type = c("tif", "iif", "se", "sem", "csem", "both"),
  facet = NULL,
  draw = TRUE,
  ...
)
```

Arguments

x	Output from compute_information().
type	"tif" for the overall precision curve (default), "iif" for facet-level contribution curves, "se" / "sem" / "csem" for the conditional standard error of measurement implied by that curve, or "both" for precision with conditional SEM on a secondary axis.
facet	For type = "iif", which facet to plot. If NULL, the first facet is used.
draw	If TRUE (default), draw the plot. If FALSE, return reusable mfrm_plot_data invisibly.
...	Additional graphical parameters.

Value

Invisibly, an mfrm_plot_data object.

Plot types

- "tif": overall design-weighted precision across theta.
- "se" / "sem" / "csem": conditional SEM across theta.
- "both": precision and conditional SEM together, useful for presentations.
- "iif": facet-level contribution curves for one selected facet in a supported RSM, PCM, or bounded GPCM fit.

Which type should I use?

- Use "tif" for a quick overall read on precision.
- Use "sem" or "csem" when standard-error language is easier to communicate than precision.
- Use "both" when you want both views in one figure.
- Use "iif" when you want to see which facet levels are shaping the total precision curve.

Interpreting output

- The total curve peaks where the realized design is most precise.
- Conditional SEM is derived as 1 / sqrt(precision); lower is better.
- Facet-level curves show which facet levels contribute most to that realized precision at each theta.
- For bounded GPCM, those contributions include the squared discrimination scaling implied by the fitted slope_facet.
- If the precision peak sits far from the bulk of person measures, the realized design may be poorly targeted.

Returned data when draw = FALSE

`draw = FALSE` returns an `mfrm_plot_data` object. The underlying plotting data are stored in `$data$plot`. For `type = "tif"`, `"se"`, or `"both"`, those rows come from `x$tif`. For `type = "iif"`, the returned rows come from `x$iif` filtered to the requested facet. The plot data also include `plot_long`, `information_long`, `conditional_sem`, `summary`, and `settings` so ggplot2, plotly, Quarto, and table workflows can reuse the information and conditional-SEM series without parsing the drawn figure.

Typical workflow

1. Compute information with `compute_information()`.
2. Plot with `plot_information(info)` for the total precision curve.
3. Use `plot_information(info, type = "iif", facet = "Rater")` for facet-level contributions.
4. Use `draw = FALSE` when you want reusable plot data for custom graphics or reporting helpers.

See Also

`compute_information()`, `fit_mfrm()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
info <- compute_information(fit)
tif_data <- plot_information(info, type = "tif", draw = FALSE)
head(tif_data$data$plot)
iif_data <- plot_information(info, type = "iif", facet = "Rater", draw = FALSE)
head(iif_data$data$plot)
```

---

plot_interrater_agreement

Plot inter-rater agreement diagnostics using base R

Description

Plot inter-rater agreement diagnostics using base R

Usage

```
plot_interrater_agreement(
  x,
  diagnostics = NULL,
  rater_facet = NULL,
  context_facets = NULL,
  exact_warn = 0.5,
  corr_warn = 0.3,
```

```
  plot_type = c("exact", "corr", "difference"),
  top_n = 20,
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

x	Output from fit_mfrm() or interrater_agreement_table().
diagnostics	Optional output from diagnose_mfrm() when x is mfrm_fit.
rater_facet	Name of the rater facet when x is mfrm_fit.
context_facets	Optional context facets when x is mfrm_fit.
exact_warn	Warning threshold for exact agreement.
corr_warn	Warning threshold for pairwise correlation.
plot_type	"exact", "corr", or "difference".
top_n	Maximum pairs displayed for bar-style plots.
main	Optional custom plot title.
palette	Optional named color overrides (ok, flag, expected).
label_angle	X-axis label angle for bar-style plots.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.

Details

Inter-rater agreement plots summarize pairwise consistency for a chosen rater facet. Agreement statistics are computed over observations that share the same person and context-facet levels, ensuring that comparisons reflect identical rating targets.

Exact agreement is the proportion of matched observations where both raters assigned the same category score. The **expected agreement** line shows the proportion expected by chance given each rater's marginal category distribution, providing a baseline.

Pairwise correlation is the Pearson correlation between scores assigned by each rater pair on matched observations.

The **difference plot** decomposes disagreement into systematic bias (mean signed difference on x-axis: positive = Rater 1 more severe) and total inconsistency (mean absolute difference on y-axis). Points near the origin indicate both low bias and low inconsistency.

The context_facets parameter specifies which facets define "the same rating target" (e.g., Criterion). When NULL, all non-rater facets are used as context.

Value

A plotting-data object of class mfrm_plot_data.

Plot types

`"exact"` **(default)** Bar chart of exact agreement proportion by rater pair. Expected agreement overlaid as connected circles. Horizontal reference line at `exact_warn`. Bars colored red when observed agreement falls below the warning threshold.

`"corr"` Bar chart of pairwise Pearson correlation by rater pair. Reference line at `corr_warn`. Ordered by correlation (lowest first). Low correlations suggest inconsistent rank ordering of persons between raters.

`"difference"` Scatter plot. X-axis: mean signed score difference (Rater 1 − Rater 2); positive values indicate Rater 1 is more severe. Y-axis: mean absolute difference (overall disagreement magnitude). Points colored red when flagged. Vertical reference at 0.

Interpreting output

Pairs below `exact_warn` and/or `corr_warn` should be prioritized for rater calibration review. On the difference plot, points far from the origin along the x-axis indicate systematic bias; points high on the y-axis indicate large inconsistency regardless of direction.

Typical workflow

1. Select rater facet and run `"exact"` view.
2. Confirm with `"corr"` view.
3. Use `"difference"` to inspect directional disagreement.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`interrater_agreement_table()`, `plot_facets_chisq()`, `plot_qc_dashboard()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
p <- plot_interrater_agreement(fit, rater_facet = "Rater", draw = FALSE)
if (interactive()) {
  plot_interrater_agreement(
    fit,
    rater_facet = "Rater",
    draw = TRUE,
    plot_type = "exact",
    main = "Inter-rater Agreement (Customized)",
    palette = c(ok = "#2b8cbe", flag = "#cb181d"),
    label_angle = 45,
    preset = "publication"
  )
}
```

plot_local_dependence_heatmap *Pairwise standardized-residual heatmap for local-dependence review*

Description

Builds an N x N heatmap of pairwise standardized residuals between facet levels, computed from the diagnostics observation table. Cells with large absolute values flag pairs of facet elements (e.g. two raters, two items) whose residuals co-move more than the main-effects MFRM expects, which is the standard Yen Q3-style indicator of local response dependence.

Usage

```
plot_local_dependence_heatmap(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  min_pairs = 5L,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output. Computed on demand when omitted.
facet	Facet whose levels are placed on both axes (default "Rater").
min_pairs	Minimum number of shared response opportunities required to retain a pair. Pairs below the threshold are shown as NA.
preset	Visual preset.
draw	If TRUE, draw with base graphics.

Details

This helper complements plot_marginal_pairwise(): the marginal version uses posterior-integrated agreement residuals on a top-N pair list, while this view shows every pair on a shared color scale so an analyst can scan for diagonal blocks or hotspots.

Value

An mfrm_plot_data whose data slot bundles the symmetric residual matrix, the long-form pairs table, and the threshold used.

See Also

plot_marginal_pairwise(), plot_qc_dashboard(), mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_local_dependence_heatmap(fit, draw = FALSE)
dim(p$data$matrix)
# Look for: |off-diagonal correlation| < 0.2 is the typical
#   acceptable regime; values >= 0.3 (Yen 1984 / Marais 2013
#   guideline) flag pairs that may share dependence beyond the
#   main-effects MFRM. Inspect those cells in `diag$obs`.
```

---

plot_marginal_fit *Plot strict marginal-fit follow-up cells using base R*

Description

Plot strict marginal-fit follow-up cells using base R

Usage

```
plot_marginal_fit(
  x,
  diagnostics = NULL,
  plot_type = c("std_residual", "prop_diff"),
  top_n = 20,
  facet = NULL,
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

x	Output from fit_mfrm() or diagnose_mfrm().
diagnostics	Optional output from diagnose_mfrm() when x is mfrm_fit.
plot_type	"std_residual" or "prop_diff".
top_n	Maximum cells shown.
facet	Optional facet name used to keep only matching facet-level rows. When NULL, the plot uses the mixed top-cell table returned by the strict marginal screen.
main	Optional custom plot title.
palette	Optional named color overrides. Recognized names: positive, negative, flag.
label_angle	X-axis label angle.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.

Details

This helper visualizes the largest first-order strict marginal-fit cells from `diagnose_mfrm(..., diagnostic_mode = "both")` or `diagnostic_mode = "marginal_fit"`.

The `"std_residual"` view ranks cells by the absolute standardized residual from posterior-integrated expected category counts. The `"prop_diff"` view ranks the same cells by the signed observed-minus-expected proportion gap.

Use this plot after `summary(diagnostics)` indicates strict marginal flags. The display is exploratory: it highlights which facet/category cells deserve follow-up, but it is not a standalone inferential test.

Value

A plotting-data object of class `mfrm_plot_data`.

Interpreting output

- Positive bars mean the observed category usage exceeded the posterior- expected marginal usage for that cell.
- Negative bars mean the observed usage fell below the posterior-expected marginal usage.
- Red bars indicate the current strict marginal warning rule was triggered by `|StdResidual| >= abs_z_warn`.

Typical workflow

1. Fit with `fit_mfrm()` using `method = "MML"` for RSM / PCM.
2. Run `diagnose_mfrm()` with `diagnostic_mode = "both"`.
3. Use `plot_marginal_fit()` to inspect the largest strict marginal cells.
4. Follow up with `rating_scale_table()` or substantive design review.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`diagnose_mfrm()`, `rating_scale_table()`, `plot_marginal_pairwise()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
  "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
```

```
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
p <- plot_marginal_fit(diag, draw = FALSE, preset = "publication")
p$data$preset
if (interactive()) {
  plot_marginal_fit(
    diag,
    plot_type = "prop_diff",
    draw = TRUE,
    preset = "publication"
  )
}
```

`plot_marginal_pairwise` *Plot strict pairwise local-dependence follow-up using base R*

Description

Plot strict pairwise local-dependence follow-up using base R

Usage

```
plot_marginal_pairwise(
  x,
  diagnostics = NULL,
  metric = c("exact", "adjacent"),
  top_n = 20,
  facet = NULL,
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`x`	Output from `fit_mfrm()` or `diagnose_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()` when `x` is `mfrm_fit`.
`metric`	`"exact"` or `"adjacent"`.
`top_n`	Maximum level pairs shown.
`facet`	Optional facet name used to keep only matching pairwise rows.
`main`	Optional custom plot title.
`palette`	Optional named color overrides. Recognized names: `ok`, `flag`.

label_angle	X-axis label angle.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draw with base graphics.

Details

This helper visualizes the strict pairwise local-dependence follow-up derived from posterior-integrated expected exact and adjacent agreement.

The "exact" view ranks level pairs by the absolute exact-agreement standardized residual. The "adjacent" view uses the adjacent-agreement standardized residual instead. Both are exploratory corroboration screens for strict marginal-fit flags.

Value

A plotting-data object of class mfrm_plot_data.

Interpreting output

- Positive bars mean the observed agreement exceeded the posterior-expected agreement for that level pair.
- Negative bars mean the observed agreement fell below the posterior-expected agreement.
- Red bars indicate the pair exceeded the current strict-warning threshold.

Typical workflow

1. Fit with fit_mfrm() using method = "MML" for RSM / PCM.
2. Run diagnose_mfrm() with diagnostic_mode = "both".
3. Use plot_marginal_pairwise() to inspect level pairs behind pairwise local-dependence flags.
4. Corroborate with legacy diagnostics, design review, and substantive interpretation before making claims.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and vignette("mfrmr-visual-diagnos package = "mfrmr").

See Also

diagnose_mfrm(), plot_marginal_fit(), mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy,
  "Person",
  c("Rater", "Criterion"),
```

```
  "Score",
  method = "MML",
  quad_points = 7,
  maxit = 30
)
diag <- diagnose_mfrm(fit, residual_pca = "none", diagnostic_mode = "both")
p <- plot_marginal_pairwise(diag, draw = FALSE, preset = "publication")
p$data$preset
if (interactive()) {
  plot_marginal_pairwise(
    diag,
    metric = "adjacent",
    draw = TRUE,
    preset = "publication"
  )
}
```

`plot_person_fit` *Plot per-person fit*

Description

Per-person diagnostic bubble plot inspired by FACETS Table 6 / KIDMAP summaries. Each bubble represents one person at the intersection of Infit (x) and Outfit (y), sized by total observations and coloured by the standard 0.5/1.5 fit envelope: green when both Infit and Outfit fall in `[lower, upper]`, amber when one statistic is outside, red when both are outside. Set `fit_index = "loglik"` for a ranked view of the report-ready `lz_star` / `lz` index instead.

Usage

```
plot_person_fit(
  fit,
  diagnostics = NULL,
  lower = 0.5,
  upper = 1.5,
  top_n_label = 12L,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  fit_index = c("meansquare", "loglik")
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`diagnostics`	Optional `diagnose_mfrm()` output. When omitted, `diagnose_mfrm(fit, residual_pca = "none")` is run internally.
`lower`	Lower fit threshold (default `0.5`, Linacre 2002).

upper	Upper fit threshold (default 1.5).
top_n_label	Maximum number of persons whose label is drawn. The default mean-square view uses largest |Infit - 1| + |Outfit - 1|; fit_index = "loglik" uses largest absolute report index. Default 12.
preset	Visual preset, including "monochrome".
draw	If TRUE, draw with base graphics.
fit_index	Plot focus. "meansquare" keeps the Infit/Outfit bubble plot. "loglik" draws the report index selected by compute_person_fit_indices() (lz_star when available, otherwise lz with a caveat).

Value

An mfrm_plot_data object whose reusable plot data include data with one row per person, plot_long for custom R graphics, person_fit_indices from compute_person_fit_indices(), and compact flag/status summaries.

Interpreting output

The default 0.5-1.5 envelope follows Linacre (2002) Rasch Measurement Transactions. Persons in the green centre are fit-acceptable; amber and red corners are candidates for misfit review (overfit / underfit) using unexpected_response_table() for follow-up.

See Also

diagnose_mfrm(), unexpected_response_table(), build_misfit_casebook().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_person_fit(fit, draw = FALSE)
head(p$data$data)
```

---

plot_qc_dashboard *Plot a base-R QC dashboard*

Description

Plot a base-R QC dashboard

Usage

```
plot_qc_dashboard(
  fit,
  diagnostics = NULL,
  threshold_profile = "standard",
  thresholds = NULL,
  abs_z_min = 2,
  prob_max = 0.3,
  rater_facet = NULL,
  interrater_exact_warn = 0.5,
  interrater_corr_warn = 0.3,
  fixed_p_max = 0.05,
  random_p_max = 0.05,
  top_n = 20,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome")
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`threshold_profile`	Threshold profile name (`strict`, `standard`, `lenient`).
`thresholds`	Optional named threshold overrides.
`abs_z_min`	Absolute standardized-residual cutoff for unexpected panel.
`prob_max`	Maximum observed-category probability cutoff for unexpected panel.
`rater_facet`	Optional rater facet used in inter-rater panel.
`interrater_exact_warn`	Warning threshold for inter-rater exact agreement.
`interrater_corr_warn`	Warning threshold for inter-rater correlation.
`fixed_p_max`	Warning cutoff for fixed-effect facet chi-square p-values.
`random_p_max`	Warning cutoff for random-effect facet chi-square p-values.
`top_n`	Maximum elements displayed in displacement panel.
`draw`	If `TRUE`, draw with base graphics.
`preset`	Visual preset (`"standard"`, `"publication"`, `"compact"`, or `"monochrome"`).

Details

The dashboard draws nine QC panels in a 3×3 grid:

Panel	What it shows	Key reference lines
1. Category counts	Observed (bars) vs model-expected counts (line)	–

2. Infit vs Outfit	Scatter of element MnSq values	heuristic 0.5, 1.0, 1.5 bands
3. |ZSTD| histogram	Distribution of absolute standardised residuals	|ZSTD| = 2
4. Unexpected responses	Standardised residual vs $-\log_{10} P_{\text{obs}}$	`abs_z_min`, `prob_max`
5. Fair-average gaps	Boxplots of (Observed - FairM) per facet	zero line
6. Displacement	Top absolute displacement values	± 0.5 logits
7. Inter-rater agreement	Exact agreement with expected overlay per pair	`interrater_exact_warn`
8. Fixed chi-square	Fixed-effect χ^2 per facet	`fixed_p_max`
9. Separation & Reliability	Bar chart of separation index per facet	–

`threshold_profile` controls warning overlays. Three built-in profiles are available: `"strict"`, `"standard"` (default), and `"lenient"`. Use `thresholds` to override any profile value with named entries.

For bounded `GPCM`, the dashboard now reuses the residual-based diagnostics stack and marks the fair-average panel unavailable rather than silently reusing the Rasch-only compatibility calculation.

Value

A plotting-data object of class `mfrm_plot_data`.

Plot types

This function draws a fixed 3×3 panel grid (no `plot_type` argument). For individual panel control, use the dedicated helpers: `plot_unexpected()`, `plot_fair_average()`, `plot_displacement()`, `plot_interrater_agreement()`, `plot_facets_chisq()`.

Interpreting output

Recommended panel order for fast review:

1. **Category counts + Infit/Outfit** (row 1): first-pass model screening. Category bars should roughly track the expected line; Infit/Outfit points are often reviewed against the heuristic 0.5–1.5 band.
2. **Unexpected responses + Displacement** (row 2): element-level outliers. Sparse points and small displacements are desirable.
3. **Inter-rater + Chi-square** (row 3): facet-level comparability. Read these as screening panels: higher agreement suggests stronger scoring consistency, and significant fixed chi-square indicates detectable facet spread under the current model.
4. **Separation/Reliability** (row 3): approximate screening precision. Higher separation indicates more statistically distinct strata under the current SE approximation.

Treat this dashboard as a screening layer; follow up with dedicated helpers (`plot_unexpected()`, `plot_displacement()`, `plot_interrater_agreement()`, `plot_facets_chisq()`) for detailed diagnosis.

Typical workflow

1. Fit and diagnose model.
2. Run `plot_qc_dashboard()` for one-page triage.
3. Drill into flagged panels using dedicated functions.

See Also

`plot_unexpected()`, `plot_fair_average()`, `plot_displacement()`, `plot_interrater_agreement()`, `plot_facets_chisq()`, `build_visual_summaries()`

Examples

```
# Fast smoke run: build the plot data only (no graphics device).
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:3], ]
fit_quick <- suppressWarnings(
  fit_mfrm(toy_small, "Person", c("Rater", "Criterion"), "Score",
           method = "JML", maxit = 3)
)
qc_quick <- plot_qc_dashboard(fit_quick, draw = FALSE)
names(qc_quick$data)


fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
qc <- plot_qc_dashboard(fit, draw = FALSE)
qc$data$panels$Status
# Look for: a row whose `Status` is "OK" for each panel that
#   the run should support. "WARN" / "REVIEW" rows tell you which
#   downstream helper to run next (e.g. `plot_unexpected()`,
#   `plot_residual_pca()`); the dashboard is a triage screen, not
#   a publication figure on its own.
if (interactive()) {
  plot_qc_dashboard(fit, rater_facet = "Rater")
}
```

---

plot_qc_pipeline *Plot QC pipeline results*

Description

Visualizes the output from `run_qc_pipeline()` as either a traffic-light bar chart or a detail panel showing values versus thresholds.

Usage

```
plot_qc_pipeline(x, type = c("traffic_light", "detail"), draw = TRUE, ...)
```

Arguments

x	Output from run_qc_pipeline().
type	Plot type: "traffic_light" (default) or "detail".
draw	If FALSE, return plot data invisibly without drawing.
...	Additional graphical parameters passed to plotting functions.

Details

Two plot types are provided for visual triage of QC results:

- "traffic_light" (default): A horizontal bar chart with one row per QC check. Bars are coloured green (Pass), amber (Warn), or red (Fail). Provides an at-a-glance summary of the current QC review state.
- "detail": A panel showing each check's observed value and its pass/warn/fail thresholds. Useful for understanding how close a borderline result is to the next verdict level.

Value

Invisible verdicts tibble from the QC pipeline.

QC checks performed

The pipeline evaluates up to 10 checks (depending on available diagnostics):

1. **Convergence**: did the optimizer converge?
2. **Overall Infit**: global information-weighted mean-square
3. **Overall Outfit**: global unweighted mean-square
4. **Misfit rate**: proportion of elements with $|\mathrm{ZSTD}| > 2$
5. **Category usage**: minimum observations per score category
6. **Disordered steps**: whether threshold estimates are monotonic
7. **Separation** (per facet): element discrimination adequacy
8. **Residual PCA eigenvalue**: first-component eigenvalue (if computed)
9. **Displacement**: maximum absolute displacement across elements
10. **Inter-rater agreement**: minimum pairwise exact agreement

Interpreting plots

- **Green** (Pass): the check meets the current threshold-profile criteria.
- **Amber** (Warn): borderline—monitor but not necessarily disqualifying. Review the detail panel to see how close the value is to the fail threshold.
- **Red** (Fail): requires investigation before strong operational or interpretive claims are made from the current run. Common remedies include collapsing categories (for disordered steps), removing outlier raters (for misfit), or increasing sample size (for low separation).
- The detail view shows numeric values, making it easy to communicate exact results to stakeholders.

See Also

run_qc_pipeline(), plot_qc_dashboard(), build_visual_summaries(), mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("study1")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
qc <- run_qc_pipeline(fit)
plot_qc_pipeline(qc, draw = FALSE)
```

---

plot_rater_agreement_heatmap

Pairwise rater-agreement heatmap

Description

Summarizes inter-rater agreement as a symmetric rater x rater heatmap. Cells are coloured by the chosen agreement metric: exact agreement proportion by default, or the Pearson-style `Corr` column from `interrater_agreement_table()` when `metric = "correlation"`. The plot is a compact alternative to `plot_interrater_agreement()`'s bar chart when the rater count exceeds ~6 pairs.

Usage

```
plot_rater_agreement_heatmap(
  fit,
  diagnostics = NULL,
  rater_facet = "Rater",
  metric = c("exact", "correlation"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`fit`	An `mfrm_fit`.
`diagnostics`	Optional `diagnose_mfrm()` output; piped through to `interrater_agreement_table()` when supplied.
`rater_facet`	Name of the rater facet (default `"Rater"`).
`metric`	Column to colour by: `"exact"` (default) or `"correlation"`. Quadratic-weighted kappa is not currently computed by `interrater_agreement_table()` and is therefore not offered here.
`preset`	Visual preset.
`draw`	If `TRUE`, draw with base graphics.

Value

An `mfrm_plot_data` object whose `data` slot bundles the rater x rater matrix and the raw pairwise rows.

See Also

`interrater_agreement_table()` for the underlying numeric table; `plot_guttman_scalogram()` for a complementary person-by-element view of residual structure; `diagnose_mfrm()` for the diagnostics bundle the heatmap reads from.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_rater_agreement_heatmap(fit, draw = FALSE)
dim(p$data$matrix)
# Look for (default `metric = "exact"`):
# - Off-diagonal cells close to the corresponding entry of
#   `summary(diag)$interrater$ExactAgreement` indicate consistent
#   pair behaviour; cells well below the average mark a pair
#   that disagrees more than the rest.
# - With `metric = "correlation"` the colour scale switches to
#   `[-1, 1]`; positive cells = pairs agree on relative ordering,
#   negative cells = pairs systematically rank persons in opposite
#   directions and are the highest-priority review cases.
```

`plot_rater_severity_profile` *Plot per-rater severity ranking with confidence interval whiskers*

Description

Ranks the levels of a chosen rater facet by estimated severity and draws each level as a horizontal CI whisker around the point estimate. Optional gentle / strict guidance bands at `+/-0.5` and `+/-1.0` logit relative to the centred mean make rater calibration easy to read for training feedback.

Usage

```
plot_rater_severity_profile(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  ci_level = 0.95,
  show_bands = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output. When omitted, diagnose_mfrm(fit, residual_pca = "none") is run internally.
facet	Facet name to plot (default "Rater"). Any non-Person facet name is accepted.
ci_level	Confidence level used for the whiskers (default 0.95). Bounds use +/- z * ModelSE.
show_bands	Logical. When TRUE (default) draw shaded +/-0.5 (gentle) and +/-1.0 (strict) logit guidance bands.
preset	Visual preset.
draw	If TRUE, draw with base graphics.

Value

An mfrm_plot_data object whose data slot contains columns Level, Estimate, SE, CI_Lower, CI_Upper, Band.

Interpreting output

The vertical reference line at zero is the sum-to-zero centring point. Levels well within +/- 0.5 logit (gentle band) are typically interchangeable in operational scoring; levels outside +/- 1.0 logit (strict band) deserve targeted training or anchoring.

See Also

diagnose_mfrm(), analyze_facet_equivalence(), plot_facet_equivalence().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_rater_severity_profile(fit, draw = FALSE)
head(p$data$data)
```

---

plot_rater_trajectory *Rater-severity trajectory across an ordered wave / occasion variable*

Description

Plots each rater's severity estimate across a user-supplied ordering variable (e.g. Session, Wave, AdminDate), producing one line per rater. When the ordering column is time-like (numeric or date), the x-axis is drawn on that scale; otherwise the values are rendered as discrete ordered categories. Useful for rater training / drift feedback loops.

Usage

```
plot_rater_trajectory(
  fits,
  facet = "Rater",
  ci_level = 0.95,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

fits	A named list of mfrm_fit objects, one per wave. Names become the x-axis labels in their supplied order. Fits are assumed to have been placed on a common scale via anchor-linking or an equivalent post-hoc transformation (see the caveat above).
facet	Facet whose levels are tracked (default "Rater").
ci_level	Confidence level for the per-wave CI ribbons drawn around each trajectory (default 0.95).
preset	Visual preset.
draw	If TRUE, draw with base graphics.

Value

An mfrm_plot_data object whose data slot is a long data.frame with Wave, Level, Estimate, SE, CI_Lower, CI_Upper columns.

Anchor-linking caveat

Each wave is fit independently under its own sum-to-zero identification, so the per-wave severity logits live on separate scales unless you actively link them. Before interpreting movement across waves as rater drift, link the waves by either (i) holding common anchors fixed across fits (see mfrmr_linking_and_dff for the supported linking route), or (ii) harmonizing the scale post-hoc with a Stocking-Lord type transformation and reviewing the result via plot_anchor_drift(). The trajectory plot itself does not perform linking; it only visualizes the supplied fits on their as-fit scales.

See Also

plot_anchor_drift(), mfrmr_linking_and_dff

Examples

```
toy <- load_mfrmr_data("example_core")
fit_a <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                  method = "JML", maxit = 30)
fit_b <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                  method = "JML", maxit = 30)
p <- plot_rater_trajectory(list(T1 = fit_a, T2 = fit_b), draw = FALSE)
```

```
head(p$data$data)
# Look for: stable trajectories (small wave-to-wave shifts within
#   each rater's CI ribbon) once the waves are anchor-linked. A
#   rater whose line drifts >0.5 logits across waves is the typical
#   "calibration drift" signal. Without anchor linking the per-wave
#   logits are on different scales and the picture cannot be read
#   as drift; see the Anchor-linking caveat in the docstring.
```

`plot_reliability_snapshot`

Facet reliability and separation snapshot bar plot

Description

Compact facet-level visual of the Wright & Masters (1982) separation, strata, and reliability indices that `diagnose_mfrm()` computes. Helpful as a single small figure for "are persons / raters / criteria distinguishable?" review. These are Rasch/FACETS-style separation indices on the fitted logit scale, not ICCs; use `compute_facet_icc()` for the complementary observed-score variance-share view.

Usage

```
plot_reliability_snapshot(
  fit,
  diagnostics = NULL,
  metric = c("reliability", "separation", "strata"),
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`diagnostics`	Optional `diagnose_mfrm()` output. Computed on demand when omitted.
`metric`	`"reliability"` (default), `"separation"`, or `"strata"`.
`preset`	Visual preset.
`draw`	If `TRUE`, draw with base graphics.

Value

An `mfrm_plot_data` whose `data` slot bundles a tidy `Facet`, `Metric`, `Value` data frame.

See Also

`diagnose_mfrm()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_reliability_snapshot(fit, draw = FALSE)
p$data$table
# Look for (default `metric = "reliability"`):
# - >= 0.9 strong, 0.7-0.9 adequate, < 0.7 weak (Wright & Masters 1982).
# - The Person row is the operative reliability for ability scores.
# - Non-Person rows (Rater / Criterion) report the same index but
#   should be read as "are facet elements distinguishable?"; values
#   close to 1 mean facet means differ reliably from each other.
```

---

plot_residual_matrix *Person x facet-level standardized-residual matrix*

Description

Visualizes the person x element matrix of standardized residuals from `diagnose_mfrm()` as a heatmap. Complements `plot_guttman_scalogram()` (which shows raw responses) by exposing the residual structure directly: large positive cells show under-prediction, negative cells over-prediction.

Usage

```
plot_residual_matrix(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  top_n_persons = 40L,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`diagnostics`	Optional `diagnose_mfrm()` output. Computed on demand when omitted.
`facet`	Facet whose levels become the column axis (default `"Rater"`).
`top_n_persons`	Cap on the number of rows. Defaults to 40 to keep the figure legible; persons are kept by largest absolute residual mean.
`preset`	Visual preset.
`draw`	If `TRUE`, draw with base graphics.

Value

An `mfrm_plot_data` whose `data` slot bundles the residual `matrix` (rows = Person, columns = facet level) and the long-form `obs` table.

See Also

`plot_guttman_scalogram()`, `plot_unexpected()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_residual_matrix(fit, top_n_persons = 12, draw = FALSE)
dim(p$data$matrix)
# Look for: cell values within ~|2| are routine; |residual| > 2 is
#   misfit at the 5% level and |residual| > 3 at the 1% level
#   (Wright & Linacre 1994). Persons with multiple high-magnitude
#   cells across the same facet level point at scoring drift.
```

---

plot_residual_pca *Visualize residual PCA results*

Description

Visualize residual PCA results

Usage

```
plot_residual_pca(
  x,
  mode = c("overall", "facet"),
  facet = NULL,
  plot_type = c("scree", "parallel_scree", "parallel_excess", "loadings"),
  component = 1L,
  top_n = 20L,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`x`	Output from `analyze_residual_pca()`, `diagnose_mfrm()`, or `fit_mfrm()`.
`mode`	`"overall"` or `"facet"`.
`facet`	Facet name for `mode = "facet"`.
`plot_type`	`"scree"`, `"parallel_scree"`, `"parallel_excess"`, or `"loadings"`.
`component`	Component index for loadings plot.

top_n	Maximum number of variables shown in loadings plot.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
draw	If TRUE, draws the plot using base graphics.

Details

x can be either:

- output of analyze_residual_pca(), or
- a diagnostics object from diagnose_mfrm() (PCA is computed internally), or
- a fitted object from fit_mfrm() (diagnostics and PCA are computed internally).

Plot types:

- "scree": component vs eigenvalue line plot
- "parallel_scree": observed eigenvalues with residual-permutation parallel-analysis mean and upper cutoff
- "parallel_excess": observed eigenvalue minus the parallel-analysis cutoff by component
- "loadings": horizontal bar chart of top absolute loadings

For mode = "facet" and facet = NULL, the first available facet is used.

Value

A named list of plotting data (class mfrm_plot_data) with:

- plot: "scree", "parallel_scree", "parallel_excess", or "loadings"
- mode: "overall" or "facet"
- facet: facet name (or NULL)
- title: plot title text
- data: underlying table used for plotting

Interpreting output

- plot_type = "scree": look for dominant early components relative to later components and the unit-eigenvalue reference line. Treat this as exploratory residual-structure screening, not a standalone unidimensionality test or a DIMTEST/UNIDIM substitute.
- plot_type = "parallel_scree" or "parallel_excess": use only after running analyze_residual_pca() with parallel = TRUE. Components above the residual-permutation cutoff are candidates for follow-up, not proof of multidimensionality.
- plot_type = "loadings": identifies variables/elements driving each component; inspect both sign and absolute magnitude.

Facet mode (mode = "facet") helps localize residual structure to a specific facet after global PCA review.

Typical workflow

1. Run diagnose_mfrm() with residual_pca = "overall" or "both".
2. Build PCA object via analyze_residual_pca() (or pass diagnostics directly).
3. Use scree plot first, then loadings plot for targeted interpretation.

See Also

analyze_residual_pca(), diagnose_mfrm()

Examples

```
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:24]
toy <- toy_full[match(toy_full$Person, toy_people, nomatch = 0L) > 0L, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
diag <- diagnose_mfrm(fit, residual_pca = "overall")
pca <- analyze_residual_pca(diag, mode = "overall")
plt <- plot_residual_pca(pca, mode = "overall", plot_type = "scree", draw = FALSE)
head(plt$data)

pca_pa <- analyze_residual_pca(diag, mode = "overall", parallel = TRUE, parallel_reps = 10)
pa <- plot_residual_pca(pca_pa, mode = "overall", plot_type = "parallel_scree", draw = FALSE)
head(pa$data)

plt_load <- plot_residual_pca(
  pca, mode = "overall", plot_type = "loadings", component = 1, draw = FALSE
)
head(plt_load$data)
if (interactive()) {
  plot_residual_pca(pca, mode = "overall", plot_type = "scree", preset = "publication")
}
```

---

plot_residual_qq *Normal quantile-quantile plot of person standardized residuals*

Description

Produces a Q-Q plot of per-person standardized residuals. Under the fitted Rasch-family model the residuals are approximately N(0, 1), so deviations from the reference line diagnose distributional misfit that mean-square summaries may miss.

Usage

```
plot_residual_qq(
  fit,
  diagnostics = NULL,
```

```
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

fit	An mfrm_fit.
diagnostics	Optional diagnose_mfrm() output; required entries are generated internally when absent.
preset	Visual preset.
draw	If TRUE, draw with base graphics.

Value

An mfrm_plot_data object with a data slot containing Person, Theoretical, Sample columns.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_residual_qq(fit, draw = FALSE)
head(p$data$data)
# Look for: points hugging the y = x reference line. Heavy upper-
#   right tails indicate persons whose residual aggregates exceed
#   the standard normal expectation; pair with `plot_unexpected()`
#   for case-level follow-up. This is an exploratory screen; do
#   not treat tail behaviour as a definitive normality test.
```

---

plot_response_time_review

Plot response-time review summaries

Description

Draw or return reusable plot data for a response_time_review() object. Plot types are descriptive screening views and do not represent a joint response-time model.

Usage

```
plot_response_time_review(
  x,
  type = c("distribution", "person", "facet", "score"),
  facet = NULL,
  top_n = 25L,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE,
  ...
)
```

Arguments

x	A mfrm_response_time_review object.
type	Plot type: "distribution", "person", "facet", or "score".
facet	Optional facet name when type = "facet".
top_n	Maximum number of person or facet rows to plot.
preset	Visual preset.
draw	If TRUE, draw with base graphics. If FALSE, return only an mfrm_plot_data object.
...	Unused.

Value

Invisibly, an mfrm_plot_data object containing the plot table, thresholds, overview, and interpretation notes.

See Also

response_time_review(), plot_data_components(), mfrmr_output_guide()

Examples

```
toy <- load_mfrmr_data("example_core")
toy$ResponseTime <- 10 + seq_len(nrow(toy)) %% 6 + as.numeric(toy$Score)
rt <- response_time_review(
  toy, person = "Person", facets = c("Rater", "Criterion"),
  score = "Score", time = "ResponseTime"
)
plot_response_time_review(rt, type = "distribution", draw = FALSE)
plot_response_time_review(rt, type = "person", draw = FALSE)
```

---

plot_shrinkage_funnel *Empirical-Bayes shrinkage funnel / caterpillar*

Description

Visualizes empirical-Bayes shrinkage by drawing one row per facet level with the raw (pre-shrinkage) and shrunken estimates plus the shrinkage factor. Rows are ordered by absolute shrinkage so the levels that move most under the prior appear at the top.

Usage

```
plot_shrinkage_funnel(
  fit,
  facet = NULL,
  top_n = 30L,
  preset = c("standard", "publication", "compact", "monochrome"),
  show_ci = FALSE,
  ci_level = 0.95,
  draw = TRUE
)
```

Arguments

fit	An mfrm_fit augmented with empirical-Bayes shrinkage.
facet	Facet to draw (default: first non-person facet with shrinkage columns present).
top_n	Maximum number of rows to draw (default 30).
preset	Visual preset.
show_ci	Logical. When TRUE, draw approximate confidence-interval whiskers for raw and shrunken estimates when SE / ShrunkSE evidence is available.
ci_level	Confidence level used when show_ci = TRUE; default 0.95.
draw	If TRUE, draw with base graphics.

Details

Requires a fit produced via apply_empirical_bayes_shrinkage() or a fit_mfrm(..., facet_shrinkage = "empirical_bayes") run, so that fit$facets$others carries Estimate, ShrunkEstimate, and ShrinkageFactor columns.

Value

An mfrm_plot_data whose data slot bundles the long Level, RawEstimate, ShrunkEstimate, ShrinkageFactor table. When show_ci = TRUE, the table also includes RawCI_Lower, RawCI_Upper, ShrunkCI_Lower, ShrunkCI_Upper, and CI_Level.

See Also

apply_empirical_bayes_shrinkage(), mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
fit_eb <- apply_empirical_bayes_shrinkage(fit)
p <- plot_shrinkage_funnel(fit_eb, draw = FALSE)
head(p$data$table)
# Look for: short segments (Raw and Shrunken close together) =
#   little pooling. Long segments fanning toward the centre = the
```

```
#   prior pulled the estimate strongly; this is most pronounced for
#   small-N levels. ShrinkageFactor near 1 means most of the
#   movement was driven by the prior rather than the data.
```

`plot_threshold_ladder` *Plot RSM/PCM threshold ladders with disorder highlighting*

Description

Renders the Rasch-Andrich threshold structure as a vertical ladder per step-facet level. Each tick is a `tau_k`; lines connecting adjacent thresholds are coloured to make disordered crossings (`tau_{k+1} < tau_k`) visually obvious. For RSM there is one ladder; for PCM (and bounded GPCM) there is one ladder per `step_facet` level.

Usage

```
plot_threshold_ladder(
  fit,
  highlight_disorder = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`fit`	An `mfrm_fit` from `fit_mfrm()`.
`highlight_disorder`	Logical. When `TRUE` (default), draw disordered segments with the preset's `fail` colour and add a subtitle counting the disordered groups.
`preset`	Visual preset (`"standard"`, `"publication"`, `"compact"`, or `"monochrome"`).
`draw`	If `TRUE`, draw with base graphics.

Value

An `mfrm_plot_data` object with a `data` slot containing columns `Group`, `Step`, `Threshold`, `Disordered` for each ladder row.

Interpreting output

Within each ladder, thresholds should ascend monotonically. A disordered crossing (highlighted in the fail colour) suggests that the corresponding category is rarely the most likely response over any logit interval, and is a common trigger for category-collapsing decisions.

See Also

`category_structure_report()`, `category_curves_report()`, `plot.mfrm_fit()` (`type = "ccc"`).

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
p <- plot_threshold_ladder(fit, draw = FALSE)
head(p$data$data)
```

`plot_unexpected` *Plot unexpected responses using base R*

Description

Plot unexpected responses using base R

Usage

```
plot_unexpected(
  x,
  diagnostics = NULL,
  abs_z_min = 2,
  prob_max = 0.3,
  top_n = 100,
  rule = c("either", "both"),
  plot_type = c("scatter", "severity"),
  main = NULL,
  palette = NULL,
  label_angle = 45,
  preset = c("standard", "publication", "compact", "monochrome"),
  draw = TRUE
)
```

Arguments

`x`	Output from `fit_mfrm()` or `unexpected_response_table()`.
`diagnostics`	Optional output from `diagnose_mfrm()` when `x` is `mfrm_fit`.
`abs_z_min`	Absolute standardized-residual cutoff.
`prob_max`	Maximum observed-category probability cutoff.
`top_n`	Maximum rows used from the unexpected table.
`rule`	Flagging rule (`"either"` or `"both"`).
`plot_type`	`"scatter"` or `"severity"`.
`main`	Optional custom plot title.
`palette`	Optional named color overrides (`higher`, `lower`, `bar`).
`label_angle`	X-axis label angle for `"severity"` bar plot.
`preset`	Visual preset (`"standard"`, `"publication"`, `"compact"`, or `"monochrome"`).
`draw`	If `TRUE`, draw with base graphics.

Details

This helper visualizes flagged observations from `unexpected_response_table()`. An observation is "unexpected" when its standardised residual and/or observed-category probability exceed user-specified cutoffs.

The **severity index** is a composite ranking metric that combines the absolute standardised residual $|Z|$ and the negative log probability $-\log_{10} P_{\text{obs}}$. Higher severity indicates responses that are more surprising under the fitted model.

The `rule` parameter controls flagging logic:

- `"either"`: flag if $|Z| \geq$ `abs_z_min` **or** $P_{\text{obs}} \leq$ `prob_max`.
- `"both"`: flag only if **both** conditions hold simultaneously.

Under common thresholds, many well-behaved runs will produce relatively few flagged observations, but the flagged proportion is design- and model-dependent. Treat the output as a screening display rather than a calibrated goodness-of-fit test.

Value

A plotting-data object of class `mfrm_plot_data`.

Plot types

`"scatter"` **(default)** X-axis: standardized residual Z. Y-axis: $-\log_{10}(P_{\text{obs}})$ (negative log of observed-category probability; higher = more surprising). Points colored orange when the observed score is *higher* than expected, teal when *lower*. Dashed lines mark `abs_z_min` and `prob_max` thresholds. Clusters of points in the upper corners indicate systematic misfit patterns worth investigating.

`"severity"` Ranked bar chart of the composite severity index for the `top_n` most unexpected responses. Bar length reflects the combined unexpectedness; labels identify the specific person-facet combination. Use for QC triage and case-level prioritization.

Interpreting output

Scatter plot: farther from zero on x-axis = larger residual mismatch; higher y-axis = lower observed-category probability. A uniform scatter with few points beyond the threshold lines indicates fewer locally surprising responses under the current thresholds.

Severity plot: focuses on the most extreme observations for targeted case review. Look for recurring persons or facet levels among the top entries—repeated appearances may signal rater misuse, scoring errors, or model misspecification.

Typical workflow

1. Fit model and run `diagnose_mfrm()`.
2. Start with `"scatter"` to assess global unexpected pattern.
3. Switch to `"severity"` for case prioritization.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`unexpected_response_table()`, `plot_fair_average()`, `plot_displacement()`, `plot_qc_dashboard()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
p <- plot_unexpected(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 10, draw = FALSE)
if (interactive()) {
  plot_unexpected(
    fit,
    abs_z_min = 1.5,
    prob_max = 0.4,
    top_n = 10,
    plot_type = "severity",
    preset = "publication",
    main = "Unexpected Response Severity (Customized)",
    palette = c(higher = "#d95f02", lower = "#1b9e77", bar = "#2b8cbe"),
    label_angle = 45
  )
}
```

---

plot_wright_unified *Plot a unified Wright map with all facets on a shared logit scale*

Description

Produces a shared-logit variable map showing person ability distribution alongside measure estimates for every facet in side-by-side columns on the same scale.

Usage

```
plot_wright_unified(
  fit,
  diagnostics = NULL,
  bins = 20L,
  show_thresholds = TRUE,
  top_n = 30L,
  show_ci = FALSE,
  ci_level = 0.95,
  draw = TRUE,
  preset = c("standard", "publication", "compact", "monochrome"),
```

```
  palette = NULL,
  label_angle = 45,
  ...
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
bins	Integer number of bins for the person histogram. Default 20.
show_thresholds	Logical; if TRUE, display threshold/step positions on the map. Default TRUE.
top_n	Maximum number of facet/step points retained for labeling.
show_ci	Logical; if TRUE, draw approximate confidence intervals when standard errors are available.
ci_level	Confidence level used when show_ci = TRUE.
draw	If TRUE (default), draw the plot. If FALSE, return plot data invisibly.
preset	Visual preset ("standard", "publication", "compact", or "monochrome").
palette	Optional named color overrides passed to the shared Wright-map drawer.
label_angle	Rotation angle for group labels on the facet panel.
...	Additional graphical parameters.

Details

This unified map arranges:

- Column 1: Person measure distribution (horizontal histogram)
- Shared facet/step panel: facet levels and optional threshold positions on the same vertical logit axis
- Range and interquartile overlays for each facet group to show spread

This is the package's most compact targeting view when you want one display that shows where persons, facet levels, and category thresholds sit relative to the same latent scale.

The logit scale on the y-axis is shared, allowing direct visual comparison of all facets and persons.

Value

Invisibly, a list with persons, facets, and thresholds data used for the plot.

Interpreting output

- Facet levels at the same height on the map are at similar difficulty.
- The person histogram shows where examinees cluster relative to the facet scale.
- Thresholds (if shown) indicate category boundary positions.
- Large gaps between the person distribution and facet locations can signal targeting problems.

Typical workflow

1. Fit a model with `fit_mfrm()`.
2. Plot with `plot_wright_unified(fit)`.
3. Compare person distribution with facet level locations.
4. Use `show_thresholds = TRUE` when you want the category structure in the same view.

When to use this instead of plot_information

Use `plot_wright_unified()` when your main question is targeting or coverage on the shared logit scale. Use `plot_information()` when your main question is measurement precision across theta.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

See Also

`fit_mfrm()`, `plot.mfrm_fit()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]
fit <- fit_mfrm(toy_small, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", model = "RSM", maxit = 30)
map_data <- plot_wright_unified(fit, draw = FALSE)
names(map_data)
```

---

precision_review_report

Build a precision review report

Description

Build a precision review report

Usage

```
precision_review_report(fit, diagnostics = NULL)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.

Details

This helper summarizes how mfrmr derived SE, CI, and reliability values for the current run. It also includes a source-grounded fit/separation basis table so users can keep mean-square fit, ZSTD standardization, Rasch/FACETS-style separation, and package QC thresholds in separate reporting lanes.

Value

A named list with:

- profile: one-row precision overview
- checks: package-native precision review checks
- fit_separation_basis: source-grounded fit/separation reporting boundary
- approximation_notes: detailed method notes
- settings: resolved model and method labels

What this review means

precision_review_report() is a reporting gatekeeper for precision claims. It tells you how the package derived uncertainty summaries for the current run and how cautiously those summaries should be written up.

What this review does not justify

- It does not, by itself, validate the measurement model or substantive conclusions.
- A favorable precision tier does not override convergence, fit, linking, or design problems elsewhere in the analysis.
- Fit and separation rows in this report are reporting/validation boundaries, not standalone success criteria.

Interpreting output

- profile: one-row overview of the active precision tier and recommended use.
- checks: package-native review checks for SE ordering, reliability ordering, coverage of sample/population summaries, and SE source labels.
- fit_separation_basis: source-grounded boundary table for fit and separation reporting.
- approximation_notes: method notes copied from diagnose_mfrm().

Recommended next step

Use the profile$PrecisionTier and checks table to decide whether SE, CI, and reliability language can be phrased as model-based, should be qualified as hybrid, or should remain exploratory in the final report.

Typical workflow

1. Run `diagnose_mfrm()` for the fitted model.
2. Build `precision_review_report(fit, diagnostics = diag)`.
3. Use `summary()` to see whether the run supports model-based reporting language or should remain in exploratory/screening mode.

See Also

`diagnose_mfrm()`, `facet_statistics_report()`, `reporting_checklist()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
out <- precision_review_report(fit, diagnostics = diag)
summary(out)
```

---

predict_mfrm_population

Forecast population-level MFRM operating characteristics for one future design

---

Description

Forecast population-level MFRM operating characteristics for one future design

Usage

```
predict_mfrm_population(
  fit = NULL,
  sim_spec = NULL,
  n_person = NULL,
  n_rater = NULL,
  n_criterion = NULL,
  raters_per_person = NULL,
  design = NULL,
  reps = 50,
  fit_method = NULL,
  model = NULL,
  maxit = 25,
  quad_points = 7,
  residual_pca = c("none", "overall", "facet", "both"),
  seed = NULL
)
```

Arguments

fit	Optional output from fit_mfrm() used to derive a fit-based simulation specification.
sim_spec	Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec(). Supply exactly one of fit or sim_spec.
n_person	Number of persons/respondents in the future design. Defaults to the value stored in the base simulation specification.
n_rater	Number of rater facet levels in the future design. Defaults to the value stored in the base simulation specification.
n_criterion	Number of criterion/item facet levels in the future design. Defaults to the value stored in the base simulation specification.
raters_per_person	Number of raters assigned to each person in the future design. Defaults to the value stored in the base simulation specification.
design	Optional named design override supplied as a named list, named vector, or one-row data frame. Names may use canonical variables (n_person, n_rater, n_criterion, raters_per_person), current public aliases (for example n_judge, n_task, judge_per_person), or role keywords (person, rater, criterion, assignment). The schema-only future branch input design$facets = c(person = ..., judge = ..., task = ...) is also accepted for the currently exposed facet keys. Do not specify the same variable through both design and the scalar count arguments.
reps	Number of replications used in the forecast simulation.
fit_method	Estimation method used inside the forecast simulation. When fit is supplied, defaults to that fit's estimation method; otherwise defaults to "MML".
model	Measurement model used when refitting the forecasted design. Defaults to the model recorded in the base simulation specification.
maxit	Maximum iterations passed to fit_mfrm() in each replication.
quad_points	Quadrature points for fit_method = "MML".
residual_pca	Residual PCA mode passed to diagnose_mfrm().
seed	Optional seed for reproducible replications.

Details

predict_mfrm_population() is a **scenario-level forecasting helper** built on top of evaluate_mfrm_design(). It is intended for questions such as:

- what separation/reliability would we expect if the next administration had 60 persons, 4 raters, and 2 ratings per person?
- how much Monte Carlo uncertainty remains around those expected summaries?

The function deliberately returns **aggregate operating characteristics** (for example mean separation, reliability, recovery RMSE, convergence rate) rather than future individual true values for one respondent or one rater.

If fit is supplied, the function first constructs a fit-derived parametric starting point with extract_mfrm_sim_spec() and then evaluates the requested future design under that explicit data-generating mechanism. This should be interpreted as a fit-based forecast under modeling assumptions, not as a guaranteed out-of-sample prediction.

When that fit-derived or manually built simulation specification stores an active latent-regression population generator, the helper still operates at the **design / operating-characteristic** level. It repeatedly simulates person-level covariates and responses, refits the MML population-model branch, and summarizes the resulting facet-level behavior. This is distinct from the fitted-model posterior scoring provided by predict_mfrm_units().

Bounded GPCM forecasts are available with caveats through the same repeated simulation/refit design route used by evaluate_mfrm_design(). They summarize design-level operating characteristics under the supplied or fit-derived slope-aware specification; they do not validate operational scoring, diagnostic-screening or signal-detection rules, slope-recovery adequacy, or arbitrary-facet planning.

Value

An object of class mfrm_population_prediction with components:

- design: requested future design
- forecast: facet-level forecast table
- overview: run-level overview
- simulation: underlying evaluate_mfrm_design() result
- sim_spec: simulation specification used for the forecast
- facet_names: public non-person facet names carried by the simulation specification
- design_variable_aliases: public aliases for n_person/n_rater/n_criterion/raters_per_person
- design_descriptor: role-based description of design variables carried from the underlying planning object
- planning_scope: explicit record of the current planning contract, including a facet_manifest and future-planner scaffold marker
- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: combined planner-schema contract carrying the role table, current boundary, mutability map, facet manifest, and a schema-only future facet-count table
- gpcm_boundary: bounded-GPCM caveat row when a GPCM forecast route is used
- settings: forecasting settings
- ademp: simulation-study metadata
- notes: interpretation notes

Interpreting output

- forecast contains facet-level expected summaries for the requested future design.
- Mcse* columns quantify Monte Carlo uncertainty from using a finite number of replications.
- design_variable_aliases and design_descriptor carry the same public naming metadata used by the underlying planning object. They rename the standard two non-person facet roles for presentation, but they do not turn the current planner into a fully arbitrary-facet simulator.

- If `sim_spec$population$active = TRUE`, the forecast summarizes repeated latent-regression MML refits under that stored person-level generator; it is still a scenario forecast rather than direct posterior scoring for one observed sample.
- `simulation` stores the full design-evaluation object in case you want to inspect replicate-level behavior.

What this does not justify

This helper does not produce definitive future person measures or rater severities for one concrete sample. It forecasts design-level behavior under the supplied or derived parametric assumptions.

References

The forecast is implemented as a one-scenario Monte Carlo / operating- characteristic study following the general guidance of Morris, White, and Crowther (2019) and the ADEMP-oriented reporting framework discussed by Siepe et al. (2024). In `mfrmr`, this function is a practical wrapper for future-design planning rather than a direct implementation of a published many-facet forecasting procedure.

- Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods*. Statistics in Medicine, 38(11), 2074-2102.
- Siepe, B. S., Bartos, F., Morris, T. P., Boulesteix, A.-L., Heck, D. W., & Pawel, S. (2024). *Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting*. Psychological Methods.

See Also

`build_mfrm_sim_spec()`, `extract_mfrm_sim_spec()`, `evaluate_mfrm_design()`, summary.mfrm_population_prediction

Examples

```
spec <- build_mfrm_sim_spec(
  n_person = 16,
  n_rater = 3,
  n_criterion = 2,
  raters_per_person = 2,
  assignment = "rotating"
)
pred <- predict_mfrm_population(
  sim_spec = spec,
  design = list(person = 18),
  reps = 1,
  maxit = 30,
  seed = 123
)
s_pred <- summary(pred)
s_pred$forecast[, c("Facet", "MeanSeparation", "McseSeparation")]
```

predict_mfrm_units *Score future or partially observed units under the fitted scoring basis*

Description

Score future or partially observed units under the fitted scoring basis

Usage

```
predict_mfrm_units(
  fit,
  new_data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
  interval_level = 0.95,
  n_draws = 0,
  seed = NULL
)
```

Arguments

fit	Output from fit_mfrm() estimated with method = "MML" or method = "JML". When fit uses the latent-regression MML branch (posterior_basis = "population_model"), score the target persons with the same background-variable contract via person_data.
new_data	Long-format data for the future or partially observed units to be scored.
person	Optional person column in new_data. Defaults to the person column recorded in fit.
facets	Optional facet-column mapping for new_data. Supply either an unnamed character vector in the calibrated facet order or a named vector whose names are the calibrated facet names and whose values are the column names in new_data.
score	Optional score column in new_data. Defaults to the score column recorded in fit.
weight	Optional weight column in new_data. Defaults to the weight column recorded in fit, if any.
person_data	Optional one-row-per-person data.frame with the background variables required by a latent-regression fit. Ignored for ordinary fixed-calibration scoring. For intercept-only latent-regression fits (population_formula = ~ 1), mfrmr reconstructs the minimal one-row-per-person table internally from the scored person IDs. This is the scoring-time table for new_data, not the fit object's replay/export provenance table. For categorical background variables, supply values on the same coding scale used at fit time; the fitted factor levels and contrasts are reused when building the scoring design matrix.

`person_id` Optional person-ID column in `person_data`. Defaults to `person` when that column exists, otherwise `"Person"` for the canonical scoring layout.

`population_policy` How missing background data are handled when `fit` uses the latent-regression branch. `"error"` (default) requires complete person-level covariates for all scored persons; `"omit"` drops scored persons lacking complete covariates and records that omission in `population_review`.

`interval_level` Posterior interval level returned in `Lower`/`Upper`.

`n_draws` Optional number of quadrature-grid posterior draws to return per scored person. Use 0 to skip draws.

`seed` Optional seed for reproducible posterior draws.

Details

`predict_mfrm_units()` is the **individual-unit companion** to `predict_mfrm_population()`. It uses the fitted calibration and, when available, the fitted one-dimensional population model to score new or partially observed persons via Expected A Posteriori (EAP) summaries on a quadrature grid.

When the original fit uses ordinary `method = "MML"`, the posterior summaries are taken under that fitted MML calibration. When the original fit uses the latent-regression MML branch, the scoring prior is the fitted conditional normal population model $\theta \mid x \sim N(x^\top \hat{\beta}, \hat{\sigma}^2)$, so the returned summaries are population-model-aware posterior EAP estimates. When the original fit uses `method = "JML"`, `mfrmr` applies the fitted facet/step parameters with a standard normal reference prior on the quadrature grid, so the returned person scores remain fixed-calibration EAP summaries rather than direct JML estimates from the fitting step.

When the fitted population model is intercept-only (`population_formula = ~ 1`), `predict_mfrm_units()` still uses the fitted population-model basis, but it can reconstruct the minimal scored-person table internally because no background covariates are needed beyond the person IDs in `new_data`.

The current bounded `GPCM` branch is included in this scoring layer, so fitted `GPCM` objects can be used for the same fixed-calibration posterior summaries. This does not imply that every downstream diagnostic or reporting helper has already been generalized to `GPCM`.

This is appropriate for questions such as:

- what posterior location/uncertainty do these partially observed new respondents have under the existing calibration?
- how uncertain are those scores, given the observed response pattern?

All non-person facet levels in `new_data` must already exist in the fitted calibration. The function does **not** recalibrate the model, update facet estimates, or treat overlapping person IDs as the same latent units from the training data. Person IDs in `new_data` are treated as labels for the rows being scored.

When `n_draws > 0`, the returned `draws` component contains discrete quadrature-grid posterior draws that can be used as approximate plausible values under the fitted scoring basis. They should be interpreted as posterior uncertainty summaries, not as deterministic future truth values.

For `JML` fits, this scoring stage is intentionally post hoc: `mfrmr` uses the fitted facet and step parameters from the joint-likelihood fit, then adds a standard normal reference prior only for the scoring layer so that new or partially observed units can be summarized on a quadrature grid. This is a

practical fixed-calibration EAP procedure, not a claim that the original JML fit itself estimated a population model.

Value

An object of class mfrm_unit_prediction with components:

- estimates: posterior summaries by person
- draws: optional quadrature-grid posterior draws
- row_review: row-level preparation review for new_data
- population_review: optional person-level omission review for latent-regression scoring
- input_data: cleaned canonical scoring rows retained from new_data
- person_data: cleaned or supplied person-level background data used for latent-regression scoring; NULL otherwise
- settings: scoring settings
- notes: interpretation notes

Interpreting output

- estimates contains posterior EAP summaries for each person in new_data.
- Lower and Upper are quadrature-grid posterior interval bounds at the requested interval_level.
- SD is posterior uncertainty under the fitted scoring basis used for scoring.
- draws, when requested, contains approximate plausible values on the fitted quadrature grid.
- population_review, when present, records whether scored persons were omitted because their background data were incomplete for a latent-regression fit.

What this does not justify

This helper does not update the original calibration, estimate new non-person facet levels, or produce deterministic future person true values. It scores new response patterns under the fitted calibration and, when applicable, the fitted one-dimensional population model.

References

The posterior summaries follow the usual quadrature-based EAP scoring framework used in item response modeling under calibrated parameters (for example Bock & Aitkin, 1981). When fit uses the latent-regression branch, mfrmr scores under the fitted conditional normal population model in the general plausible-values spirit discussed by Mislevy (1991). Optional posterior draws are exposed as quadrature-grid plausible-value-style summaries for practical many-facet scoring rather than as a claim of full ConQuest numerical equivalence. When the source fit is JML, the same literature supports the quadrature-based scoring layer, but the standard normal prior is a package-level reference prior introduced for post hoc scoring rather than an estimated population distribution.

- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Mislevy, R. J. (1991). *Randomization-based inference about latent variables from complex samples*. Psychometrika, 56(2), 177-196.

- Muraki, E. (1992). *A generalized partial credit model: Application of an EM algorithm.* Applied Psychological Measurement, 16(2), 159-176.

See Also

predict_mfrm_population(), fit_mfrm(), summary.mfrm_unit_prediction

Examples

```
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- suppressWarnings(
  fit_mfrm(
    toy[toy$Person %in% keep_people, , drop = FALSE],
    "Person", c("Rater", "Criterion"), "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
)
raters <- unique(toy$Rater)[1:2]
criteria <- unique(toy$Criterion)[1:2]
new_units <- data.frame(
  Person = c("NEW01", "NEW01", "NEW02", "NEW02"),
  Rater = c(raters[1], raters[2], raters[1], raters[2]),
  Criterion = c(criteria[1], criteria[2], criteria[1], criteria[2]),
  Score = c(2, 3, 2, 4)
)
pred_units <- predict_mfrm_units(toy_fit, new_units, n_draws = 0)
summary(pred_units)$estimates[, c("Person", "Estimate", "Lower", "Upper")]
```

---

print.mfrm_apa_text *Print APA narrative text with preserved line breaks*

Description

Print APA narrative text with preserved line breaks

Usage

```
## S3 method for class 'mfrm_apa_text'
print(x, ...)
```

Arguments

x	Character text object from build_apa_outputs()$report_text.
...	Reserved for generic compatibility.

Details

Prints APA narrative text with preserved paragraph breaks using `cat()`. This is preferred over bare `print()` when you want readable multi-line report output in the console.

Value

The input object (invisibly).

Interpreting output

The printed text is the same content stored in `build_apa_outputs(...)$report_text`, but with explicit paragraph breaks.

Typical workflow

1. Generate `apa <- build_apa_outputs(...)`.
2. Print readable narrative with `apa$report_text`.
3. Use `summary(apa)` to check completeness before manuscript use.

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
apa <- build_apa_outputs(fit, diag)
apa$report_text
```

---

q3_statistic *Yen-style Q3 local-dependence statistic between facet levels*

Description

Computes a Q3-style index inspired by Yen (1984) – the Pearson correlation of **standardized** residuals between every pair of levels of a chosen facet – from a `diagnose_mfrm()` bundle. Under the conditional-independence assumption of the MFRM, |Q3| should be small for every pair; large absolute values flag pairs of facet elements (e.g. two raters or two items) whose residuals co-move more than the main-effects model expects.

Usage

```
q3_statistic(
  fit,
  diagnostics = NULL,
  facet = "Rater",
  min_pairs = 5L,
  yen_threshold = 0.2,
  marais_threshold = 0.3,
  relative_offset = 0.2
)
```

Arguments

fit	An mfrm_fit from fit_mfrm().
diagnostics	Optional diagnose_mfrm() output. Computed on demand when omitted.
facet	Facet whose levels are paired (default "Rater").
min_pairs	Minimum number of shared response opportunities required to retain a pair. Pairs below the threshold drop out of the table (mirrors plot_local_dependence_heatmap()).
yen_threshold	Community-convention flag threshold (default 0.20). Often attributed to Yen (1984), but the 0.20 cutoff is actually from Chen & Thissen (1997, p. 284); Yen herself did not propose a fixed cutoff. The cutoff was derived for raw-residual Q3 under the 3PL - applying it to standardized-residual Q3 under MFRM is approximate.
marais_threshold	Stricter community-convention threshold (default 0.30). Marais (2013, p. 121) reports this as a value "often considered" in the literature, not as her own recommendation; her actual recommendation is the relative comparison implemented by relative_offset.
relative_offset	Screening offset for the relative-flag rule |Q3 - mean(Q3)| > relative_offset (default 0.20). This is a simplified screening approximation of the relative comparison advocated by Marais (2013) and operationalized by Christensen et al. (2017) as Q3_* = Q3_max - mean(Q3). Christensen et al. (2017) demonstrate empirically that no single critical value is appropriate across designs and recommend a parametric bootstrap; the fixed 0.20 here is a screening default, not a substitute for that bootstrap.

Value

An object of class mfrm_q3 containing:

pairs A data frame with one row per facet-level pair and columns Level1, Level2, Q3, N, AbsQ3, YenFlag, MaraisFlag, RelativeFlag, and a textual Interpretation summarising which thresholds were exceeded.

summary One-row tibble with MeanQ3, MaxAbsQ3, and the three flagged-pair counts.

thresholds The thresholds used, for reproducibility.

facet The facet whose levels were paired.

Statistic definition (departures from Yen 1984)

This implementation differs from Yen's (1984) original definition in two respects that together affect threshold interpretation.

(1) Standardized vs raw residuals. Yen (1984, eqs. 7-8, p. 127) defines Q3 = cor(d_i, d_j) where d_{ik} = u_{ik} - P_hat_{ik} is the **raw** residual. mfrmr uses **standardized** residuals Z = (u - P_hat) / sqrt(Var(u)) because that is what diagnose_mfrm() stores. Standardization down-weights high-variance observations and changes the sampling distribution of the resulting correlation; the published critical values (Chen & Thissen, 1997; Christensen et al., 2017) were derived for raw-residual Q3.

(2) Mean-aggregation. When the facet being paired (e.g. `Rater`) has multiple residual rows per (Person, Level) cell because of additional facets in the design (e.g. multiple `Criterion` rows per Person-Rater cell), the standardized residuals are first **mean-aggregated to one value per (Person, Level) cell**, and the Pearson correlation is taken over those mean-aggregated residuals. Yen's original formulation takes the correlation directly over per-(Person, Item) residuals, without aggregation. Mean-aggregation reduces noise but also shrinks the effective sample size and can pull correlations toward the cell mean.

For both reasons, treat the values returned here as a **screening summary** rather than a direct substitute for the published Q3 thresholds. For a formal local-dependence test under raw-residual Q3, use a parametric bootstrap as recommended by Christensen et al. (2017).

References

- Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement, 8*(2), 125-145. doi:10.1177/014662168400800201
- Chen, W.-H., & Thissen, D. (1997). Local dependence indexes for item pairs using item response theory. *Journal of Educational and Behavioral Statistics, 22*(3), 265-289. (Origin of the commonly cited `|Q3| > 0.20` cutoff.)
- Marais, I. (2013). Local dependence. In K. B. Christensen, S. Kreiner, & M. Mesbah (Eds.), *Rasch models in health* (pp. 111-130). London: ISTE / Wiley.
- Christensen, K. B., Makransky, G., & Horton, M. (2017). Critical values for Yen's Q3: Identification of local dependence in the Rasch model using residual correlations. *Applied Psychological Measurement, 41*(3), 178-194. doi:10.1177/0146621616677520

See Also

`plot_local_dependence_heatmap()`, `diagnose_mfrm()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
q3 <- q3_statistic(fit)
q3$summary
# Look for: MaxAbsQ3 < 0.20 (Chen & Thissen 1997 community cutoff) is
#   the comfortable regime; values above 0.30 are commonly considered
#   strict-flag worthy (Marais, 2013, summarising literature). For a
#   formal test, use a parametric bootstrap (Christensen et al., 2017).
#   The summary's flag counts give a quick triage; inspect `q3$pairs`
#   for the offending level pairs and follow up with content review.
head(q3$pairs)
```

`rater_halo_network_analysis` *Analyze rater-by-criterion halo-effect networks*

Description

Analyze rater-by-criterion halo-effect networks

Usage

```
rater_halo_network_analysis(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  criterion_facet = NULL,
  context_facets = NULL,
  method = c("spearman", "pearson", "kendall"),
  min_pair_n = 5,
  alpha = 0.05,
  p_adjust = "bonferroni",
  min_abs_weight = 0,
  halo_weight_review = 0.5,
  halo_contrast_review = 0.1,
  min_retained_halo_edges = 1,
  positive_only = TRUE,
  include_graph = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`.
`rater_facet`	Name of the rater-like facet.
`criterion_facet`	Name of the criterion, rubric, task, or item-like facet used to form rater-by-criterion nodes.
`context_facets`	Facets defining rows in the reshaped wide matrix. Defaults to the person facet plus any facets other than the rater and criterion facets.
`method`	Correlation method used for rater-by-criterion node pairs.
`min_pair_n`	Minimum shared contexts required to estimate a node-pair relationship.
`alpha`	Adjusted p-value threshold for retaining edges. Set to 1 to retain all finite correlations after `min_abs_weight` filtering.
`p_adjust`	Multiple-comparison adjustment passed to `stats::p.adjust()`. The default `"bonferroni"` follows the conservative screening used in Lamprianou's halo-network example.

`min_abs_weight` Minimum absolute correlation retained as a graph edge.

`halo_weight_review`
: Same-rater cross-criterion mean absolute correlation at or above which a rater is marked for review.

`halo_contrast_review`
: Minimum difference between a rater's mean halo edge weight and incident non-halo edge weight for a stronger review flag.

`min_retained_halo_edges`
: Minimum retained halo edges required before a strong `"warning"` status is assigned.

`positive_only` If `TRUE`, negative correlations are kept in `pair_metrics` but excluded from the graph edge table.

`include_graph` If `TRUE`, include the underlying `igraph` object.

Details

`rater_halo_network_analysis()` reshapes rating data so that each rater-by-criterion combination is a node. Edges are correlations between those node score vectors across shared contexts. Edges connecting two nodes from the same rater but different criteria are labelled `"halo"`; all other retained edges are labelled `"non_halo"`.

Per-rater `ReviewStatus` combines same-rater cross-criterion mean weight, incident non-halo comparison weight, and the number of retained halo edges. A `"warning"` means these criteria converge strongly enough to prioritize follow-up; `"review"` means at least one screening criterion is elevated. Neither label is a causal halo diagnosis.

The key descriptive comparison is the distribution of halo-edge weights versus non-halo-edge weights. A larger halo-edge distribution is consistent with a halo pattern, but this function deliberately reports it as a screening diagnostic. The included Welch test is descriptive only because edge weights are clustered by rater and node.

Value

A bundle of class `mfrm_halo_network` containing:

`summary` One-row halo-network summary and halo/non-halo contrast.

`node_metrics` Rater-by-criterion node strength and centrality.

`edge_metrics` Retained graph edges.

`pair_metrics` All estimated node-pair correlations before edge filtering.

`halo_summary_by_rater` Per-rater summaries of same-rater criterion-pair edges, including `ReviewStatus` and `ReviewReason`.

`caveats` Interpretation notes.

References

Lai, E. R., Wolfe, E. W., & Vickers, D. (2015). Differentiation of illusory and true halo in writing scores. *Educational and Psychological Measurement, 75*(1), 102-125.

Lamprianou, I. (2025). Network Analysis for the investigation of rater effects in language assessment: A comparison of ChatGPT vs human raters. *Research Methods in Applied Linguistics, 4*, 100205.

See Also

rater_network_analysis(), interrater_agreement_table(), plot.mfrm_bundle()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("igraph", quietly = TRUE)) {
  halo <- rater_halo_network_analysis(fit)
  halo$summary
  head(halo$halo_summary_by_rater)
  plot(halo, type = "edge_distribution", draw = FALSE)
}
```

---

rater_network_analysis

Analyze rater agreement, disagreement, and severity-direction networks

---

Description

Analyze rater agreement, disagreement, and severity-direction networks

Usage

```
rater_network_analysis(
  fit,
  diagnostics = NULL,
  rater_facet = NULL,
  context_facets = NULL,
  mode = c("agreement", "disagreement", "severity_direction"),
  weight_metric = NULL,
  min_pair_n = 1,
  min_weight = 0,
  score_diff_tolerance = 0,
  severity_continuity = 0.5,
  exact_warn = 0.5,
  corr_warn = 0.3,
  include_graph = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().

`rater_facet` Name of the rater-like facet. If omitted, mfrmr uses the same heuristic as `interrater_agreement_table()`.

`context_facets` Facets defining shared scoring contexts. By default, the person facet and all non-rater facets are used.

`mode` Network definition. `"agreement"` builds an undirected network whose edge weights represent observed agreement. `"disagreement"` builds an undirected network whose edge weights represent observed disagreement. `"severity_direction"` builds a directed network: an edge from rater A to rater B means A assigned higher scores than B in shared contexts and is therefore relatively more lenient under the usual higher-score-is-better rating convention.

`weight_metric` Pair-level weight used for `"agreement"` or `"disagreement"` networks. Defaults to `Exact` for agreement and `MAD` for disagreement. Available pair columns include `Exact`, `Adjacent`, `Corr`, `MAD`, `OneMinusExact`, and `AbsMeanDiff`.

`min_pair_n` Minimum number of shared contexts required for a rater pair to contribute an edge.

`min_weight` Minimum edge weight retained in the graph.

`score_diff_tolerance`
Score-difference tolerance for directed severity networks. With the default `0`, any higher score contributes to the outgoing leniency edge. Larger values reproduce thresholded disagreement displays such as "only differences greater than 3 marks".

`severity_continuity`
Continuity constant added to incoming and outgoing strengths before computing the finite severity index `-log((OutStrength + c) / (InStrength + c))`.

`exact_warn, corr_warn`
Passed to `interrater_agreement_table()` to keep pair flags consistent with the tabular agreement view.

`include_graph` If `TRUE`, include the underlying `igraph` object in the returned bundle.

Details

This function implements a package-native rater-effect network view complementary to MFRM output. It follows the pairwise-network logic used in Lamprianou's rater-effect network work: nodes are raters, edges summarize pairwise relationships among raters in shared scoring contexts, and directed disagreement edges can be interpreted as relative leniency/severity indicators. These network summaries are descriptive diagnostics, not Rasch logit estimates and not formal fit statistics.

For `mode = "severity_direction"`, outgoing strength means the rater more often assigned higher scores than comparison raters; incoming strength means comparison raters more often assigned higher scores than this rater. The reported `SeverityIndex` is positive for relatively severe raters and negative for relatively lenient raters, but it is on a network-analysis scale and should not be read as an MFRM severity logit.

Value

A bundle of class `mfrm_rater_network` containing:

`summary` One-row graph summary.

`node_metrics` Rater-level degree, strength, centrality, and severity-direction summaries.

`edge_metrics` Retained rater-pair network edges.

`pair_metrics` All eligible pairwise agreement and directional comparison metrics before edge thresholding.

`caveats` Interpretation notes and sparse-design warnings.

`source_interrater` The underlying `interrater_agreement_table()` output used for agreement statistics.

References

Lamprianou, I. (2018). Investigation of rater effects using Social Network Analysis and Exponential Random Graph Models. *Educational and Psychological Measurement, 78*(3), 430-459.

Lamprianou, I. (2025). Network Analysis for the investigation of rater effects in language assessment: A comparison of ChatGPT vs human raters. *Research Methods in Applied Linguistics, 4*, 100205.

See Also

`interrater_agreement_table()`, `plot_interrater_agreement()`, `mfrm_network_analysis()`, `plot.mfrm_bundle()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
if (requireNamespace("igraph", quietly = TRUE)) {
  rn <- rater_network_analysis(fit, mode = "severity_direction")
  rn$summary
  head(rn$node_metrics)
  plot(rn, type = "severity", draw = FALSE)
}
```

---

rating_scale_table *Build a rating-scale diagnostics report*

Description

Build a rating-scale diagnostics report

Usage

```
rating_scale_table(
  fit,
  diagnostics = NULL,
  whexact = FALSE,
  drop_unused = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
whexact	Use exact ZSTD transformation for category fit.
drop_unused	If TRUE, remove categories with zero count from the displayed category table; summary and caveats still retain the omitted score-support warning.

Details

This helper provides category usage/fit statistics and threshold summaries for reviewing score-category functioning. The category usage portion is a global observed-score screen. In PCM fits with a step_facet, threshold diagnostics should be interpreted within each StepFacet rather than as one pooled whole-scale verdict.

Typical checks:

- sparse category usage (Count, ExpectedCount)
- category fit (Infit, Outfit, ZStd)
- threshold ordering within each StepFacet (threshold_table$Estimate, GapFromPrev)

Value

A named list with:

- category_table: category-level counts, expected counts, fit, and ZSTD
- threshold_table: model step/threshold estimates
- summary: one-row summary (usage and threshold monotonicity)
- caveats: structured score-support warning/review rows
- diagnostic_mode: character scalar carried from diagnostics$diagnostic_mode ("legacy", "both", or "marginal_fit"); used by downstream reporting helpers to pick the correct expected-count basis
- marginal_fit: list bundle from diagnostics$marginal_fit when strict marginal fit was computed, otherwise NULL. Carries the raw OverallRMSD / OverallMaxAbsStdResidual / per-cell tables that feed the MarginalOverallRMSD columns in summary.

Interpreting output

Start with summary:

- UsedCategories close to total Categories suggests that most score categories are represented in the observed data.
- very small MinCategoryCount indicates potential instability.
- ThresholdMonotonic = FALSE indicates disordered thresholds within at least one threshold set. In PCM fits, inspect threshold_table by StepFacet before drawing scale-wide conclusions.

Then inspect:

- category_table for global category-level misfit/sparsity.
- threshold_table for adjacent-step gaps and ordering within each StepFacet.

Typical workflow

1. Fit model: `fit_mfrm()`.
2. Build diagnostics: `diagnose_mfrm()`.
3. Run `rating_scale_table()` and review `summary()`.
4. Use `plot()` to visualize category profile quickly.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and `vignette("mfrmr-visual-diagnos package = "mfrmr")`.

Output columns

The `category_table` data.frame contains:

Category Score category value.

Count, Percent Observed count and percentage of total.

AvgPersonMeasure Mean person measure for respondents in this category.

Infit, Outfit Category-level fit statistics.

InfitZSTD, OutfitZSTD Standardized fit values.

ExpectedCount, DiffCount Expected count and observed-expected difference.

LowCount Logical; `TRUE` if count is below minimum threshold.

InfitFlag, OutfitFlag, ZSTDFlag Fit-based warning flags.

ZeroCount, UnusedCategoryType, WeaklyIdentified, CategoryCaveat Structured score-support caveats for retained zero-count categories.

The `threshold_table` data.frame contains:

Step Step label (e.g., "1-2", "2-3").

Estimate Estimated threshold/step difficulty (logits).

StepFacet Threshold family identifier when the fit uses facet-specific threshold sets.

GapFromPrev Difference from the previous threshold within the same `StepFacet` when thresholds are facet-specific. Gaps below 1.4 logits may indicate category underuse; gaps above 5.0 may indicate wide unused regions (Linacre, 2002).

ThresholdMonotonic Logical flag repeated within each threshold set. For PCM fits, read this within `StepFacet`, not as a pooled item-bank verdict.

LowerCategory, UpperCategory, WeaklyIdentified, ThresholdCaveat Adjacent score-category support metadata. Thresholds adjacent to retained zero-count categories are flagged for cautious interpretation.

References

- Andrich, D. (1978). *A rating formulation for ordered response categories*. Psychometrika, 43(4), 561-573. doi:10.1007/BF02293814
- Masters, G. N. (1982). *A Rasch model for partial credit scoring*. Psychometrika, 47(2), 149-174. doi:10.1007/BF02296272
- Linacre, J. M. (2002). What do Infit and Outfit, mean-square and standardized mean? *Rasch Measurement Transactions, 16*(2), 878. (Source for the 0.5-1.5 mean-square acceptance band and the threshold-gap heuristics used in `summary(t8)$summary`.)
- Wind, S. A. (2023). *Detecting rating scale malfunctioning with the partial credit model and generalized partial credit model*. Educational and Psychological Measurement, 83(5), 953-983. doi:10.1177/00131644221116292 (Recent simulation evidence on PCM- and GPCM-based rating-scale diagnostics; useful for interpreting the `summary(t8)$summary` flags in the bounded `GPCM` route.)

See Also

`diagnose_mfrm()`, `measurable_summary_table()`, `plot.mfrm_fit()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
t8 <- rating_scale_table(fit)
summary(t8)
summary(t8)$summary
p_t8 <- plot(t8, draw = FALSE)
p_t8$data$plot
```

---

read_facets_fit_table *Read a FACETS fit table for fit review*

Description

Read a FACETS fit table for fit review

Usage

```
read_facets_fit_table(
  file,
  facet = NULL,
  facet_map = NULL,
  format = c("auto", "delimited", "scorefile"),
  facet_col = NULL,
  level_col = NULL,
  delimiter = NULL,
```

```
  encoding = "UTF-8"
)

import_facets_fit_table(
  file,
  facet = NULL,
  facet_map = NULL,
  format = c("auto", "delimited", "scorefile"),
  facet_col = NULL,
  level_col = NULL,
  delimiter = NULL,
  encoding = "UTF-8"
)
```

Arguments

`file`	Path to a FACETS-derived fit table. A character vector of files is accepted. A directory containing `score.N.txt` files is also accepted.
`facet`	Optional facet name to assign when the file does not contain a facet column. Use this for one-facet CSV exports.
`facet_map`	Optional character vector mapping FACETS score-file numbers to facet names, for example `c("1" = "Person", "2" = "Rater")`. If unnamed, positions are used as score-file numbers.
`format`	File format. `"auto"` detects FACETS `score.N.txt` files from their name/header; `"delimited"` reads a CSV/TSV/semicolon table; and `"scorefile"` reads a FACETS score-file table.
`facet_col, level_col`	Optional explicit column names for delimited tables when automatic detection is not sufficient. For score files, `level_col` can override the column immediately before `F-Number`.
`delimiter`	Optional delimiter for delimited tables. If omitted, comma, tab, and semicolon are detected from the header line.
`encoding`	File encoding passed to `readLines()`.

Details

This helper does not run FACETS. It reads FACETS output that already exists on disk and normalizes it to columns that `facets_fit_review()` can consume: `Facet`, `Level`, `Estimate`, `SE`, `N`, `Infit`, `Outfit`, `InfitZSTD`, `OutfitZSTD`, `DF_Infit`, and `DF_Outfit`.

Two common workflows are supported:

- a FACETS score file such as `score.2.txt`, where the facet name is supplied by `facet_map` or inferred as `Facet2`. Both comma-delimited score files with field names and fixed-field score files using the FACETS manual column positions are supported;
- a CSV/TSV table already exported from FACETS or a harmonization script, with FACETS-style column names such as `Infit MnSq`, `Outfit ZStd`, and `T.Count`.

After import, pass the table to facets_fit_review(). Inspect review$external_table_quality first when the FACETS export is partial, duplicated, or missing MnSq/df columns. Then inspect review$external_comparison for supplied FACETS-vs-mfrmr differences and review$df_sensitivity / review$df_sensitive for engine-vs-FACETS-style df/ZSTD convention sensitivity. Use plot(review, type = "df_sensitivity") for a quick visual check of the largest ZSTD shifts caused by df convention.

Value

A tibble with standardized fit-table columns suitable for facets_fit_review(fit, facets_fit = read_facets_fit_table(...)).

See Also

facets_fit_review(), diagnose_mfrm()

Examples

```
path <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    Facet = "Rater", Level = "R1", Infit = 1.02, Outfit = 0.98,
    InfitZSTD = 0.3, OutfitZSTD = -0.2, DF_Infit = 12, DF_Outfit = 13
  ),
  path,
  row.names = FALSE
)
read_facets_fit_table(path)
```

---

recode_missing_codes *Recode common missing-value sentinels to* NA

Description

Convenience helper that replaces the standard non-NA missing-code sentinels used in SPSS / SAS / FACETS exports (99, 999, -1, "N", "NA", "n/a", ".", "") with NA across the columns you select. This is the R counterpart of the preprocessing UI in the companion Streamlit app and is useful before calling fit_mfrm() on data exported with those conventions.

Usage

```
recode_missing_codes(
  data,
  columns = NULL,
  codes = c("99", "999", "-1", "N", "NA", "n/a", ".", ""),
  numeric_codes = TRUE,
  verbose = FALSE
)
```

Arguments

data	A data frame.
columns	Character vector of column names to recode. Defaults to NULL, in which case all columns are scanned.
codes	Character vector of code values to convert to NA. Defaults to the FACETS / SPSS / SAS conventions; override when your instrument uses different sentinels.
numeric_codes	Logical; if TRUE (default), numeric columns are also compared against the numeric conversion of codes.
verbose	Logical; if TRUE, emits a message() summary of per-column replacement counts.

Value

The input data with the specified missing sentinels replaced by NA. A mfrm_missing_recoding attribute records the per-column replacement counts for traceability logs.

See Also

describe_mfrm_data(), fit_mfrm().

Examples

```
dat <- data.frame(
  Person = paste0("P", 1:5),
  Rater = c("R1", "R1", "R2", "R2", "R2"),
  Score = c(1, 99, 2, -1, 3)
)
cleaned <- recode_missing_codes(dat, columns = "Score")
cleaned$Score
attr(cleaned, "mfrm_missing_recoding")
```

---

recommend_mfrm_design *Recommend a design condition from simulation results*

Description

Recommend a design condition from simulation results

Usage

```
recommend_mfrm_design(
  x,
  facets = c("Rater", "Criterion"),
  min_separation = 2,
  min_reliability = 0.8,
  max_severity_rmse = 0.5,
  max_misfit_rate = 0.1,
```

```
  min_convergence_rate = 1,
  prefer = c("n_person", "raters_per_person", "n_rater", "n_criterion")
)
```

Arguments

x	Output from evaluate_mfrm_design() or summary.mfrm_design_evaluation().
facets	Facets that must satisfy the planning thresholds.
min_separation	Minimum acceptable mean separation.
min_reliability	Minimum acceptable mean reliability.
max_severity_rmse	Maximum acceptable severity recovery RMSE.
max_misfit_rate	Maximum acceptable mean misfit rate.
min_convergence_rate	Minimum acceptable convergence rate.
prefer	Ranking priority among design variables. Earlier entries are optimized first when multiple designs pass. Custom public aliases from sim_spec are also accepted, as are the role keywords person, rater, criterion, and assignment.

Details

This helper converts a design-study summary into a simple planning table.

A design is marked as recommended when all requested facets satisfy all selected thresholds simultaneously. If multiple designs pass, the helper returns the smallest one according to prefer (by default: fewer persons first, then fewer ratings per person, then fewer raters, then fewer criteria).

Value

A list of class mfrm_design_recommendation with:

- facet_table: facet-level threshold checks, including design-variable alias columns when applicable
- design_table: design-level aggregated checks, including design-variable alias columns when applicable
- recommended: the first passing design after ranking
- thresholds: thresholds used in the recommendation
- design_variable_aliases: accepted public aliases for design variables
- design_descriptor: role-based design-variable metadata
- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: combined planner-schema contract
- caveats: structured warning rows for situations where the recommendation rests on weak evidence (e.g., no design met every threshold; the recommended design is at the boundary of the evaluated grid; only one rep was simulated). Empty tibble() when no caveats apply.

Typical workflow

1. Run evaluate_mfrm_design().
2. Review summary.mfrm_design_evaluation() and plot.mfrm_design_evaluation().
3. Use recommend_mfrm_design(...) to identify the smallest acceptable design.

See Also

evaluate_mfrm_design(), summary.mfrm_design_evaluation, plot.mfrm_design_evaluation

Examples

```
sim_eval <- suppressWarnings(evaluate_mfrm_design(
  n_person = c(8, 12),
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 1,
  reps = 1,
  maxit = 30,
  seed = 123
))
rec <- recommend_mfrm_design(sim_eval)
rec$recommended
```

---

reference_case_benchmark

Benchmark packaged reference cases

Description

Benchmark packaged reference cases

Usage

```
reference_case_benchmark(
 cases = c("synthetic_truth", "synthetic_latent_regression", "synthetic_bias_contract",
    "study1_itercal_pair", "study2_itercal_pair", "combined_itercal_pair"),
  method = "MML",
  model = "RSM",
  quad_points = 7,
  maxit = 40,
  reltol = 1e-06,
  mml_engine = c("direct", "em", "hybrid")
)
```

Arguments

cases	Reference cases to run. Defaults to the standard RSM-compatible reference suite. Specialized GPCM and ConQuest-overlap package-side cases can be requested explicitly.
method	Estimation method passed to fit_mfrm(). Defaults to "MML".
model	Model family passed to fit_mfrm(). Defaults to "RSM".
quad_points	Quadrature points for method = "MML".
maxit	Maximum optimizer iterations passed to fit_mfrm().
reltol	Convergence tolerance passed to fit_mfrm().
mml_engine	MML optimization engine passed to fit_mfrm(). Applies only when method = "MML".

Details

This function checks mfrmr against the package's curated reference case families:

- synthetic_truth: checks whether recovered facet measures align with the known generating values from the package's synthetic design.
- synthetic_latent_regression: checks whether the first-version latent-regression MML branch recovers known population coefficients, residual latent variance, criterion ordering, and posterior-shift direction from a synthetic overlap case.
- synthetic_latent_regression_omit: checks whether the population-model complete-case omission policy is reflected in the fitted metadata, response-row review, active person estimates, and replay provenance.
- synthetic_conquest_overlap_dry_run: builds the narrow ConQuest-overlap bundle for the latent-regression synthetic case, round-trips package tables through the normalization/review helpers, and confirms the package-side workflow without claiming that ConQuest itself was executed.
- synthetic_gpcm: checks whether the bounded GPCM branch recovers known criterion-specific slopes, row-centered step parameters, and criterion ordering from a synthetic overlap case. This case currently requires model = "GPCM" and is intended for method = "MML".
- synthetic_bias_contract: checks whether package bias tables and pairwise local comparisons satisfy the identities documented in the bias help workflow.
- *_itercal_pair: compares a baseline packaged dataset with its iterative recalibration counterpart to review fit stability, facet-measure alignment, and linking coverage together.

The resulting object is intended as a reference-case check for package behavior. It does not by itself establish external validity against FACETS, ConQuest, or published calibration studies, and it does not assume any familiarity with external table numbering or printer layouts. When specialized latent-regression omission or ConQuest-overlap package-side cases are requested, summary(bench) prints preview rows from population_policy_checks and conquest_overlap_checks alongside the reference notes so the package-versus-external validation boundary remains visible.

Value

An object of class mfrm_reference_benchmark.

Interpreting output

- `overview`: one-row reference-case summary.
- `case_summary`: pass/warn/fail triage by reference case.
- `fit_runs`: fitted-run metadata (fit, precision tier, convergence, and latent-regression population-model/posterior-basis fields, including categorical-coding details when present).
- `design_checks`: exact design recovery checks for each dataset.
- `recovery_checks`: known-truth recovery metrics for the synthetic cases, including the latent-regression reference case.
- `bias_checks`: source-backed bias/local-measure identity checks.
- `pair_checks`: paired-dataset stability screens for the iterated cases.
- `linking_checks`: common-element reviews for paired calibration datasets.
- `conquest_overlap_checks`: package-side checks for the ConQuest-overlap bundle/normalization/review workflow; this remains a package-side check until actual ConQuest output tables are supplied.
- `population_policy_checks`: complete-case omission checks for population model benchmark fixtures.
- `source_profile`: source-backed rules used by the reference checks.

Examples

```
bench <- reference_case_benchmark(
  cases = "synthetic_truth",
  method = "JML",
  maxit = 30
)
summary(bench)
```

---

reference_case_review *Build a package-native reference review for report completeness*

Description

Build a package-native reference review for report completeness

Usage

```
reference_case_review(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  reference_profile = c("core", "compatibility"),
  include_metrics = TRUE,
  top_n_attention = 15L
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm(). If omitted, diagnostics are computed internally with residual_pca = "none".
bias_results	Optional output from estimate_bias(). If omitted and at least two facets exist, a 2-way interaction screen is computed internally.
reference_profile	Review profile. "core" emphasizes package-native report contracts. "compatibility" exposes the manual-aligned compatibility layer used by facets_output_contract_review(branch = "facets").
include_metrics	If TRUE, run numerical consistency checks in addition to schema coverage checks.
top_n_attention	Number of lowest-coverage components to keep in attention_items.

Details

This function repackages the output-contract review into package-native terminology so users can review output completeness without needing external manual/table numbering. It reports:

- component-level schema coverage
- numerical consistency checks for derived report tables
- the highest-priority attention items for follow-up

It is a package-output completeness review, not an external validation study.

Use reference_profile = "core" for ordinary mfrmr workflows. Use reference_profile = "compatibility" only when you explicitly want to inspect the compatibility layer.

Value

An object of class mfrm_reference_review.

Interpreting output

- overall: one-row review summary with schema coverage and metric pass rate.
- component_summary: per-component coverage summary.
- attention_items: direct list of components needing review.
- metric_summary / metric_checks: numerical consistency status.

See Also

facets_output_contract_review(), diagnose_mfrm(), build_fixed_reports()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
review <- reference_case_review(fit, diagnostics = diag)
summary(review)
```

---

reporting_checklist *Build an auto-filled MFRM reporting checklist*

Description

Build an auto-filled MFRM reporting checklist

Usage

```
reporting_checklist(
  fit,
  diagnostics = NULL,
  bias_results = NULL,
  hierarchical_structure = NULL,
  include_references = TRUE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. When `NULL`, diagnostics are computed with `residual_pca = "none"`.
`bias_results`	Optional output from `estimate_bias()` or a named list of such outputs.
`hierarchical_structure`	Optional output from `analyze_hierarchical_structure()`. When supplied, the "Hierarchical structure review" checklist item is flipped to `DraftReady = TRUE` and its `Detail` column surfaces the number of nested / crossed facet pairs and whether the ICC table is available.
`include_references`	If `TRUE`, include a compact reference table in the returned bundle.

Details

This helper ports the app-level reporting checklist into a package-native bundle. It does not try to judge substantive reporting quality; instead, it checks whether the fitted object and related diagnostics contain the evidence typically reported in MFRM write-ups.

Checklist items are grouped into seven core sections:

- Method section

- Global fit
- Facet-level statistics
- Element-level statistics
- Rating scale diagnostics
- Bias/interaction analysis
- Visual displays

When a fit uses the latent-regression population-model branch, the checklist also adds a `Population Model` section covering coefficient reporting, categorical model-matrix coding, complete-case omissions, posterior-basis wording, and ConQuest scope wording.

The output is designed for manuscript preparation, reproducibility records, and reproducible reporting workflows.

Value

A named list with checklist tables. Class: `mfrm_reporting_checklist`.

What this checklist means

`reporting_checklist()` is a manuscript-preparation guide. It tells you which reporting elements are already present in the current analysis objects and which still need to be generated or documented. The primary draft-status column is `DraftReady`; `ReadyForAPA` is retained as a backward-compatible alias.

What this checklist does not justify

- It is not a single run-level pass/fail decision for publication.
- `DraftReady = TRUE` / `ReadyForAPA = TRUE` does not certify formal inferential adequacy.
- Missing bias rows may simply mean `bias_results` were not supplied.

Interpreting output

- `checklist`: one row per reporting item with `Available = TRUE/FALSE`. `DraftReady = TRUE` means the item can be drafted into a report with the package's documented caveats. `ReadyForAPA` is a backward-compatible alias of the same flag; neither field certifies formal inferential adequacy.
- `section_summary`: available items by section.
- The Global Fit section includes a "Fit/separation reporting boundary" row that points to `precision_review_report()`, `fit_measures_table()`, and `facets_fit_review()` before users phrase fit, ZSTD, separation, or reliability claims.
- `software_scope`: external-software relationship summary for `mfrmr`, FACETS, ConQuest, and SPSS-style tabular handoffs.
- `facets_positioning`: report-ready wording that states `mfrmr` is not a FACETS numerical clone and separates native estimation from FACETS-style handoff or external-table review.
- `visual_scope`: plotting-route summary that separates report-default 2D figures from exploratory surface/3D-ready data handoffs, including a short `InterpretationCheck` for the main user-facing caveat.
- `references`: core background references when requested.

Recommended next step

Review the rows with `Available = FALSE` or `DraftReady = FALSE`, then add the missing diagnostics, bias results, or narrative context before calling `build_apa_outputs()` for draft text generation. For `RSM` / `PCM` reporting runs, the preferred route is an `MML` fit plus `diagnose_mfrm(..., diagnostic_mode = "both")` so the checklist can see the legacy and strict marginal screens together.

How this differs from operational review

`reporting_checklist()` is the manuscript/reporting branch of the package. Use it when the question is "what is still missing from the report?" rather than "which observations or links need follow-up?" For operational review:

- Use `build_misfit_casebook()` after `diagnose_mfrm()` when you need ranked misfit cases and grouping views for local follow-up.
- Use `build_linking_review()` after anchor/drift/chain helpers when you need operational linking triage rather than manuscript-oriented reporting tables.

Typical workflow

1. Fit with `fit_mfrm()`. For `RSM` / `PCM` reporting runs, prefer `method = "MML"`.
2. Compute diagnostics with `diagnose_mfrm()`. For `RSM` / `PCM`, prefer `diagnostic_mode = "both"`.
3. Run `reporting_checklist()` to see which reporting elements are already available from the current analysis objects.
4. If the issue is operational rather than manuscript-facing, branch to `build_misfit_casebook()` or `build_linking_review()` instead of treating `reporting_checklist()` as the single review hub.

See Also

`build_apa_outputs()`, `build_visual_summaries()`, `specifications_report()`, `data_quality_report()`, `build_misfit_casebook()`, `build_linking_review()`

Examples

```
# Fast smoke run: a JML fit + legacy-only diagnostic produces a
# populated checklist in well under a second.
toy <- load_mfrmr_data("example_core")
fit_quick <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                      method = "JML", maxit = 30)
diag_quick <- diagnose_mfrm(fit_quick, residual_pca = "none",
                            diagnostic_mode = "legacy")
chk_quick <- reporting_checklist(fit_quick, diagnostics = diag_quick)
head(chk_quick$checklist[, c("Section", "Item", "DraftReady")])


fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
```

```
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
chk <- reporting_checklist(fit, diagnostics = diag)
summary(chk)
# Look for: a high `Ready` / `Total` ratio in the summary block.
#   Sections with `Ready = 0` need follow-up before submitting
#   (typically diagnostic_mode = "both" or a residual-PCA pass).
apa <- build_apa_outputs(fit, diag)
head(chk$checklist[, c("Section", "Item", "DraftReady", "NextAction")])
# Look for: every row where `DraftReady = "yes"` is ready to paste
#   into the manuscript. `"no"` rows include a concrete `NextAction`
#   step (e.g. "run plot_qc_dashboard()") so the gap can be closed
#   without re-reading the methodology guide.
nchar(apa$report_text)
```

`response_time_review` *Review response-time patterns outside the MFRM likelihood*

Description

Build a descriptive response-time review table from the same long-format rating-event data used by `fit_mfrm()`. This helper does not fit a joint response-time model and does not change MFRM estimates. It summarizes response-time distributions, distributional rapid/slow flags, and person / facet / score-level response-time patterns for screening and reporting context.

Usage

```
response_time_review(
  data,
  person,
  facets = NULL,
  time,
  score = NULL,
  time_unit = "seconds",
  min_time = 0,
  rapid_threshold = NULL,
  slow_threshold = NULL,
  rapid_quantile = 0.05,
  slow_quantile = 0.95,
  rapid_rate_warn = 0.25,
  slow_rate_warn = 0.25,
  min_n_flag = 3L
)
```

Arguments

`data`	A data.frame in long format with one row per observed rating event.
`person`	Column name for the person identifier.

facets	Optional character vector of facet columns to summarize.
time	Column name containing positive response times.
score	Optional ordered-score column. When supplied, score-level response-time summaries are returned.
time_unit	Label for the response-time unit, such as "seconds".
min_time	Minimum valid response time. Values must be strictly greater than this threshold; default 0.
rapid_threshold	Optional numeric response-time cutoff for rapid responses. When NULL, it is estimated from rapid_quantile.
slow_threshold	Optional numeric response-time cutoff for slow responses. When NULL, it is estimated from slow_quantile.
rapid_quantile	Quantile used when rapid_threshold = NULL.
slow_quantile	Quantile used when slow_threshold = NULL.
rapid_rate_warn	Group-level rapid-response rate that creates a descriptive flag; default 0.25.
slow_rate_warn	Group-level slow-response rate that creates a descriptive flag; default 0.25.
min_n_flag	Minimum group size before rapid/slow rates are flagged; default 3.

Value

An object of class mfrm_response_time_review, a list with overview, thresholds, observations, person_summary, facet_summary, score_summary, flags, and notes.

See Also

plot_response_time_review(), fit_mfrm(), mfrmr_visual_diagnostics, mfrmr_output_guide()

Examples

```
toy <- load_mfrmr_data("example_core")
toy$ResponseTime <- 12 + as.numeric(factor(toy$Person)) * 0.4 +
  as.numeric(toy$Score)
rt <- response_time_review(
  toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  time = "ResponseTime"
)
summary(rt)
plot_response_time_review(rt, draw = FALSE)
```

review_accessors *Extract canonical review components*

Description

Extract canonical review components

Usage

```
anchor_review(x, required = TRUE)

precision_review(x, required = TRUE)
```

Arguments

x	A fitted mfrm_fit, diagnostics object, summary object, or compatible list containing the requested review component.
required	Logical. If TRUE, error when no compatible review component is found. If FALSE, return NULL instead.

Details

anchor_review() returns the fitted object's config$anchor_review component. precision_review() returns the diagnostics precision_review table. These helpers intentionally do not search older field names.

Value

anchor_review() returns an mfrm_anchor_review-like object when available. precision_review() returns the precision-review table when available. If required = FALSE and no component is available, both helpers return NULL.

Examples

```
fit_like <- list(config = list(
  anchor_review = structure(list(issue_counts = data.frame()),
                            class = "mfrm_anchor_review")
))
anchor_review(fit_like)
diag_like <- list(
  precision_review = data.frame(Check = "SE", Status = "ok", Detail = "Model-based")
)
precision_review(diag_like)
```

`review_conquest_overlap` *Review an exact-overlap ConQuest comparison against an* `mfrmr` *overlap bundle*

Description

Review an exact-overlap ConQuest comparison against an `mfrmr` overlap bundle

Usage

```
review_conquest_overlap(
  bundle,
  conquest_population = NULL,
  conquest_item_estimates = NULL,
  conquest_case_eap = NULL,
  conquest_population_term = "auto",
  conquest_population_estimate = "auto",
  conquest_item_id = "auto",
  conquest_item_estimate = "auto",
  item_id_source = c("auto", "response_var", "level"),
  conquest_case_person = "auto",
  conquest_case_estimate = "auto"
)
```

Arguments

`bundle`	Output from `build_conquest_overlap_bundle()`.
`conquest_population`	Normalized ConQuest population-parameter table as a data.frame, or output from `normalize_conquest_overlap_tables()`.
`conquest_item_estimates`	Normalized ConQuest item-estimate table as a data.frame. Leave `NULL` when `conquest_population` is an object from `normalize_conquest_overlap_tables()`.
`conquest_case_eap`	Normalized ConQuest case-level EAP table as a data.frame. Leave `NULL` when `conquest_population` is an object from `normalize_conquest_overlap_tables()`.
`conquest_population_term`	Column in `conquest_population` that stores parameter names. `"auto"` tries conservative aliases such as `Parameter` and `Term`.
`conquest_population_estimate`	Column in `conquest_population` that stores parameter estimates. `"auto"` tries aliases such as `Estimate` and `Est`.
`conquest_item_id`	Column in `conquest_item_estimates` that stores the item identifier. This may be the exported response variable (for example `I001`) or the original item/facet level. `"auto"` tries aliases such as `ResponseVar`, `ItemID`, `Item`, and `Label`.

conquest_item_estimate
: Column in conquest_item_estimates that stores the item estimate. "auto" tries aliases such as Estimate, Est, and Facility.

item_id_source
: How conquest_item_id should be matched. "auto" chooses the larger overlap between exported response variables and original item levels, with ties resolved toward exported response variables.

conquest_case_person
: Column in conquest_case_eap that stores person IDs. "auto" tries conservative aliases such as Person, PID, and Sequence ID.

conquest_case_estimate
: Column in conquest_case_eap that stores case EAP estimates. "auto" tries conservative aliases such as Estimate, EAP_1, and EAP.

Details

This helper compares normalized ConQuest output tables against the exact- overlap bundle produced by build_conquest_overlap_bundle(). It is intentionally conservative:

- it does **not** parse raw ConQuest text output automatically;
- it expects already normalized data frames or output from normalize_conquest_overlap_tables();
- and it reports numerical differences and missing elements without claiming that any fixed tolerance implies software equivalence.

This is the package's external-table review path. It is distinct from reference_case_benchmark(cases = "synthetic_conquest_overlap_dry_run"), which only round-trips package-native tables through the same normalization and review contract without executing ConQuest.

The intended workflow is:

1. export an exact-overlap bundle with build_conquest_overlap_bundle();
2. run the narrow matching case in ConQuest;
3. normalize the resulting ConQuest outputs into data frames;
4. pass those tables here to inspect direct differences, centered item agreement, and case-level EAP agreement.

Value

A named list with class mfrm_conquest_overlap_review.

Output

The returned object has class mfrm_conquest_overlap_review and includes:

- overall: one-row comparison summary with missing/duplicate/non-numeric attention-item counts and worst-row labels
- population_comparison: parameter-by-parameter comparison table
- item_comparison: centered item-estimate comparison table
- case_comparison: case-level EAP comparison table

- attention_items: missing, malformed, or unmatched elements
- settings: review settings
- notes: interpretation notes

Interpretation

- Read summary(review)$review_scope first to confirm that the result is a supplied-table review, not raw ConQuest text parsing or a software- equivalence claim.
- Population slopes and sigma2 are intended for direct comparison.
- Item estimates should be interpreted after centering.
- Case estimates should be interpreted as posterior EAP summaries under the fitted population model.
- The overall table reports both mean and maximum absolute differences for compared population, centered item, and case rows. The PopulationMaxAbsParameter, ItemCenteredMaxAbsItem, and CaseMaxAbsPerson columns identify the row where each maximum absolute difference occurs.
- Missing or non-numeric rows in attention_items indicate that the external tables do not yet align cleanly with the exported overlap bundle.

See Also

build_conquest_overlap_bundle(), normalize_conquest_overlap_files(), normalize_conquest_overlap_tables
reference_case_benchmark()

Examples

```
bundle <- build_conquest_overlap_bundle()
raw_pop <- data.frame(
  Term = bundle$mfrmr_population$Parameter,
  Est = bundle$mfrmr_population$Estimate
)
raw_item <- data.frame(
  Item = bundle$mfrmr_item_estimates$ResponseVar,
  Est = bundle$mfrmr_item_estimates$Estimate
)
raw_case <- data.frame(
  PID = bundle$mfrmr_case_eap$Person,
  EAP = bundle$mfrmr_case_eap$Estimate
)
normalized <- normalize_conquest_overlap_tables(
  conquest_population = raw_pop,
  conquest_item_estimates = raw_item,
  conquest_case_eap = raw_case,
  conquest_population_term = "Term",
  conquest_population_estimate = "Est",
  conquest_item_id = "Item",
  conquest_item_estimate = "Est",
  conquest_case_person = "PID",
  conquest_case_estimate = "EAP"
```

```
)
review <- review_conquest_overlap(bundle, normalized)
summary(review)$summary
```

`review_mfrm_anchors` *Review and normalize anchor/group-anchor tables*

Description

Review and normalize anchor/group-anchor tables

Usage

```
review_mfrm_anchors(
  data,
  person,
  facets,
  score,
  anchors = NULL,
  group_anchors = NULL,
  weight = NULL,
  rating_min = NULL,
  rating_max = NULL,
  keep_original = FALSE,
  missing_codes = NULL,
  min_common_anchors = 5L,
  min_obs_per_element = 30,
  min_obs_per_category = 10,
  noncenter_facet = "Person",
  dummy_facets = NULL
)
```

Arguments

`data`	A data.frame in long format (one row per rating event).
`person`	Column name for person IDs.
`facets`	Character vector of facet column names.
`score`	Column name for observed score.
`anchors`	Optional anchor table (Facet, Level, Anchor).
`group_anchors`	Optional group-anchor table (Facet, Level, Group, GroupValue).
`weight`	Optional weight/frequency column name.
`rating_min`	Optional minimum category value.
`rating_max`	Optional maximum category value.
`keep_original`	Keep original category values.

`missing_codes` Optional. `NULL` (default) is a no-op; `TRUE` or `"default"` converts the FACETS / SPSS / SAS sentinel set to `NA` on the person, facets, and score columns before review. Supply a character vector for a custom code set.

`min_common_anchors`
Minimum anchored levels per linking facet used in recommendations (default `5`).

`min_obs_per_element`
Minimum weighted observations per facet level used in recommendations (default `30`).

`min_obs_per_category`
Minimum weighted observations per score category used in recommendations (default `10`).

`noncenter_facet`
One facet to leave non-centered.

`dummy_facets` Facets to fix at zero.

Details

Anchoring (also called "fixing" or scale linking) constrains selected parameter estimates to pre-specified values, placing the current analysis on a previously established scale. This is essential when comparing results across administrations, linking test forms, or monitoring rater drift over time.

This function applies the same preprocessing and key-resolution rules as `fit_mfrm()`, but returns a review object so constraints can be checked *before* estimation. Running the review first helps avoid estimation failures caused by misspecified or data-incompatible anchors.

Anchor types:

- *Direct anchors* fix individual element measures to specific logit values (e.g., Rater R1 anchored at 0.35 logits).
- *Group anchors* constrain the mean of a set of elements to a target value, allowing individual elements to vary freely around that mean.
- When both types overlap for the same element, the direct anchor takes precedence.

Design checks verify that each anchored element has at least `min_obs_per_element` weighted observations (default 30) and each score category has at least `min_obs_per_category` (default 10). These thresholds follow standard Rasch sample-size recommendations (Linacre, 1994).

Value

A list of class `mfrm_anchor_review` with:

- `anchors`: cleaned anchor table used by estimation
- `group_anchors`: cleaned group-anchor table used by estimation
- `facet_summary`: counts of levels, constrained levels, and free levels
- `design_checks`: observation-count checks by level/category
- `thresholds`: active threshold settings used for recommendations

- `issue_counts`: issue-type counts
- `issues`: list of issue tables
- `recommendations`: package-native anchor guidance strings

Interpreting output

- `issue_counts`/`issues`: concrete data or specification problems.
- `facet_summary`: constraint coverage by facet.
- `design_checks`: whether anchor targets have enough observations.
- `recommendations`: action items before estimation.

Typical workflow

1. Build candidate anchors (e.g., with `make_anchor_table()`).
2. Run `review_mfrm_anchors(...)`.
3. Resolve issues, then fit with `fit_mfrm()`.

See Also

`fit_mfrm()`, `describe_mfrm_data()`, `make_anchor_table()`

Examples

```
toy <- load_mfrmr_data("example_core")

anchors <- data.frame(
  Facet = c("Rater", "Rater"),
  Level = c("R1", "R1"),
  Anchor = c(0, 0.1),
  stringsAsFactors = FALSE
)
review <- review_mfrm_anchors(
  data = toy,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  anchors = anchors
)
review$issue_counts
summary(review)
p_review <- plot(review, draw = FALSE)
p_review$data$plot
```

run_mfrm_facets *Run a legacy-compatible estimation workflow wrapper*

Description

This helper mirrors `mfrmRFacets.R` behavior as a package API and keeps legacy-compatible defaults (`model = "RSM"`, `method = "JML"`), while allowing users to choose compatible estimation options.

Usage

```
run_mfrm_facets(
  data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  keep_original = FALSE,
  model = c("RSM", "PCM"),
  method = c("JML", "JMLE", "MML"),
  step_facet = NULL,
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
  positive_facets = NULL,
  quad_points = 15,
  maxit = 400,
  reltol = 1e-06,
  mml_engine = c("direct", "em", "hybrid"),
  top_n_interactions = 20L
)

mfrmRFacets(
  data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  keep_original = FALSE,
  model = c("RSM", "PCM"),
  method = c("JML", "JMLE", "MML"),
  step_facet = NULL,
  anchors = NULL,
  group_anchors = NULL,
  noncenter_facet = "Person",
  dummy_facets = NULL,
```

```
  positive_facets = NULL,
  quad_points = 15,
  maxit = 400,
  reltol = 1e-06,
  mml_engine = c("direct", "em", "hybrid"),
  top_n_interactions = 20L
)
```

Arguments

data	A data.frame in long format.
person	Optional person column name. If NULL, guessed from names.
facets	Optional facet column names. If NULL, inferred from remaining columns after person/score/weight mapping.
score	Optional score column name. If NULL, guessed from names.
weight	Optional weight column name.
keep_original	Passed to fit_mfrm().
model	MFRM model ("RSM" default, or "PCM").
method	Estimation method ("JML" default; "JMLE" and "MML" also supported).
step_facet	Step facet for PCM mode; passed to fit_mfrm().
anchors	Optional anchor table (data.frame).
group_anchors	Optional group-anchor table (data.frame).
noncenter_facet	Non-centered facet passed to fit_mfrm().
dummy_facets	Optional dummy facets fixed at zero.
positive_facets	Optional facets with positive orientation.
quad_points	Quadrature points for MML; passed to fit_mfrm().
maxit	Maximum optimizer iterations.
reltol	Optimization tolerance.
mml_engine	MML optimization engine passed to fit_mfrm(). Applies only when method = "MML".
top_n_interactions	Number of rows for interaction diagnostics.

Details

run_mfrm_facets() is intended as a one-shot workflow helper: fit -> diagnostics -> key report tables. Returned objects can be inspected with summary() and plot().

Value

A list with components:

- fit: fit_mfrm() result
- diagnostics: diagnose_mfrm() result
- iteration: estimation_iteration_report() result
- fair_average: fair_average_table() result
- rating_scale: rating_scale_table() result
- run_info: run metadata table
- mapping: resolved column mapping

Estimation-method notes

- method = "JML" (default): legacy-compatible joint estimation route; the default preserves the FACETS-style output continuity that existing one-shot scripts expect. For new analysis scripts, prefer fit_mfrm(..., method = "MML") – MML is the package-wide recommended route because person parameters are integrated out under an N(0, 1) prior and per-person posterior SEs are available.
- method = "JMLE": explicit JMLE label; internally equivalent to JML route.
- method = "MML": marginal maximum likelihood route using quad_points. Use mml_engine = "em" or "hybrid" only for RSM / PCM fits when you want the staged MML alternatives.

model = "PCM" is supported; set step_facet when facet-specific step structure is needed.

Visualization

- plot(out, type = "fit") delegates to plot.mfrm_fit() and returns fit-level visual bundles (e.g., Wright/pathway/CCC).
- plot(out, type = "qc") delegates to plot_qc_dashboard() and returns a QC dashboard plot object.

Interpreting output

Start with summary(out):

- check convergence and iteration count in overview.
- confirm resolved columns in mapping.

Then inspect:

- out$rating_scale for category/threshold behavior.
- out$fair_average for observed-vs-model scoring tendencies.
- out$diagnostics for misfit/reliability/interactions.

Typical workflow

1. Run `run_mfrm_facets()` with explicit column mapping.
2. Check `summary(out)` and `summary(out$diagnostics)`.
3. Visualize with `plot(out, type = "fit")` and `plot(out, type = "qc")`.
4. Export selected tables for reporting (`out$rating_scale`, `out$fair_average`).

Preferred route for new analyses

For new scripts, prefer the package-native route: `fit_mfrm()` -> `diagnose_mfrm()` -> `reporting_checklist()` -> `build_apa_outputs()`. Use `run_mfrm_facets()` when you specifically need the legacy-compatible one-shot wrapper.

See Also

`fit_mfrm()`, `diagnose_mfrm()`, `estimation_iteration_report()`, `fair_average_table()`, `rating_scale_table()`, mfrmr_visual_diagnostics, mfrmr_workflow_methods, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:12], , drop = FALSE]

# Legacy-compatible default: RSM + JML
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
out$fit$summary[, c("Model", "Method", "MethodUsed")]
s <- summary(out)
s$overview[, c("Model", "Method", "Converged")]
p_fit <- plot(out, type = "fit", draw = FALSE)
p_fit$wright_map$data$plot

# Optional: MML route
if (interactive()) {
  out_mml <- run_mfrm_facets(
    data = toy_small,
    person = "Person",
    facets = c("Rater", "Criterion"),
    score = "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
  out_mml$fit$summary[, c("Model", "Method", "MethodUsed")]
}
```

`run_qc_pipeline` *Run automated quality control pipeline*

Description

Integrates convergence, model fit, reliability, separation, element misfit, unexpected responses, category structure, connectivity, inter-rater agreement, and DIF/bias into a single pass/warn/fail report.

Usage

```
run_qc_pipeline(
  fit,
  diagnostics = NULL,
  threshold_profile = "standard",
  thresholds = NULL,
  rater_facet = NULL,
  include_bias = TRUE,
  bias_results = NULL
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Output from `diagnose_mfrm()`. Computed automatically if NULL.
`threshold_profile`	Threshold preset: `"strict"`, `"standard"` (default), or `"lenient"`.
`thresholds`	Named list to override individual thresholds.
`rater_facet`	Character name of the rater facet for inter-rater check (auto-detected if NULL).
`include_bias`	If `TRUE` and bias available in diagnostics, check DIF/bias.
`bias_results`	Optional pre-computed bias results from `estimate_bias()`.

Details

The pipeline evaluates 10 quality checks and assigns a verdict (Pass / Warn / Fail) to each. The overall status is the most severe verdict across all checks. Diagnostics are computed automatically via `diagnose_mfrm()` if not supplied.

Reliability and separation are used here as QC signals. In `mfrmr`, `Reliability` / `Separation` are model-based facet indices and `RealReliability` / `RealSeparation` provide more conservative lower bounds. For `MML`, these rely on model-based `ModelSE` values for non-person facets; for `JML`, they remain exploratory approximations.

Three threshold presets are available via `threshold_profile`:

Aspect	strict	standard	lenient
Global fit warn	1.3	1.5	1.7

Global fit fail	1.5	2.0	2.5
Reliability pass	0.90	0.80	0.70
Separation pass	3.0	2.0	1.5
Misfit warn (pct)	3	5	10
Unexpected fail	3	5	10
Min cat count	15	10	5
Agreement pass	60	50	40
Bias fail (pct)	5	10	15

Individual thresholds can be overridden via the `thresholds` argument (a named list keyed by the internal threshold names shown above).

For bounded `GPCM`, this pipeline is available as caveated operational triage over supported diagnostics. Its pass/warn/fail labels remain package QC policy overlays; they are not FACETS score-side equivalence, operational scoring decisions, design-forecasting evidence, or automatic fairness / validity decisions.

Value

Object of class `mfrm_qc_pipeline` with verdicts, overall status, details, and recommendations.

QC checks

The 10 checks are:

1. **Convergence**: Did the model converge?
2. **Global fit**: Infit/Outfit MnSq within the current review band.
3. **Reliability**: Minimum non-person facet model reliability index.
4. **Separation**: Minimum non-person facet model separation index.
5. **Element misfit**: Percentage of elements with Infit/Outfit outside the current review band.
6. **Unexpected responses**: Percentage of observations with large standardized residuals.
7. **Category structure**: Minimum category count and threshold ordering.
8. **Connectivity**: All observations in a single connected subset.
9. **Inter-rater agreement**: Exact agreement percentage for the rater facet (if applicable).
10. **Functioning/Bias screen**: Percentage of interaction cells that cross the screening threshold (if interaction results are available).

Interpreting output

- `$overall`: character string `"Pass"`, `"Warn"`, or `"Fail"`.
- `$verdicts`: tibble with columns `Check`, `Verdict`, `Value`, and `Threshold` for each of the 10 checks.
- `$details`: character vector of human-readable detail strings.
- `$raw_details`: named list of per-check numeric details for programmatic access.
- `$recommendations`: character vector of actionable suggestions for checks that did not pass.
- `$config`: records the threshold profile and effective thresholds.

Typical workflow

1. Fit a model: `fit <- fit_mfrm(...)`.
2. Optionally compute diagnostics and bias: `diag <- diagnose_mfrm(fit)`; `bias <- estimate_bias(fit, diag, ...)`.
3. Run the pipeline: `qc <- run_qc_pipeline(fit, diag, bias_results = bias)`.
4. Check `qc$overall` for the headline verdict.
5. Review `qc$verdicts` for per-check details.
6. Follow `qc$recommendations` for remediation.
7. Visualize with `plot_qc_pipeline()`.

See Also

`diagnose_mfrm()`, `estimate_bias()`, `mfrm_threshold_profiles()`, `plot_qc_pipeline()`, `plot_qc_dashboard()`, `build_visual_summaries()`

Examples

```
toy <- load_mfrmr_data("study1")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
qc <- run_qc_pipeline(fit)
qc
summary(qc)
qc$verdicts
```

---

sample_mfrm_plausible_values

Sample approximate plausible values under fitted posterior scoring

---

Description

Sample approximate plausible values under fitted posterior scoring

Usage

```
sample_mfrm_plausible_values(
  fit,
  new_data,
  person = NULL,
  facets = NULL,
  score = NULL,
  weight = NULL,
  person_data = NULL,
  person_id = NULL,
  population_policy = c("error", "omit"),
```

```
  n_draws = 5,
  interval_level = 0.95,
  seed = NULL
)
```

Arguments

fit	Output from fit_mfrm() estimated with method = "MML" or method = "JML".
new_data	Long-format data for the future or partially observed units to be scored.
person	Optional person column in new_data. Defaults to the person column recorded in fit.
facets	Optional facet-column mapping for new_data. Supply either an unnamed character vector in the calibrated facet order or a named vector whose names are the calibrated facet names and whose values are the column names in new_data.
score	Optional score column in new_data. Defaults to the score column recorded in fit.
weight	Optional weight column in new_data. Defaults to the weight column recorded in fit, if any.
person_data	Optional one-row-per-person data.frame with the background variables required by a latent-regression fit. Ignored for ordinary fixed-calibration scoring. Intercept-only latent-regression fits can reconstruct the minimal scored-person table internally. This is the scoring-time table for new_data, not the fit object's replay/export provenance table. For categorical background variables, supply values on the same coding scale used at fit time; the fitted factor levels and contrasts are reused when building the scoring design matrix.
person_id	Optional person-ID column in person_data.
population_policy	How missing background data are handled when fit uses the latent-regression branch. "error" (default) requires complete person-level covariates; "omit" drops scored persons lacking complete covariates and records that omission in population_review.
n_draws	Number of posterior draws per person. Must be a positive integer.
interval_level	Posterior interval level passed to predict_mfrm_units() for the accompanying EAP summary table.
seed	Optional seed for reproducible posterior draws.

Details

sample_mfrm_plausible_values() is a thin public wrapper around predict_mfrm_units() that exposes the fixed-calibration posterior draws as a standalone object. It is useful when downstream workflows want repeated latent-value imputations rather than just one posterior EAP summary.

In the current mfrmr implementation these are **approximate plausible values** drawn from the fitted quadrature-grid posterior under the scoring basis implied by fit. For ordinary MML fits this is the fitted marginal calibration; for latent-regression MML fits it is the fitted conditional normal population model for the scored persons; for JML fits it is the fixed facet/step calibration together with a standard

normal reference prior on the quadrature grid. They should be interpreted as posterior uncertainty summaries for the scored persons, not as deterministic future truth values and not as a claim of full many-facet plausible-values equivalence with population-model software.

In other words, the `JML` path here is a practical scoring approximation layered on top of the fitted joint-likelihood calibration, whereas the latent-regression `MML` path uses the fitted one-dimensional conditional normal population model. Neither path should be described as a full many-facet plausible-values system with all ConQuest-style extensions.

Value

An object of class `mfrm_plausible_values` with components:

- `values`: one row per person per draw
- `estimates`: companion posterior EAP summaries
- `row_review`: row-preparation review
- `population_review`: optional person-level omission review for latent-regression scoring
- `input_data`: cleaned canonical scoring rows retained from `new_data`
- `person_data`: cleaned or supplied person-level background data used for latent-regression scoring; `NULL` otherwise
- `settings`: scoring settings
- `notes`: interpretation notes

Interpreting output

- `values` contains one row per person per draw.
- `estimates` contains the companion posterior EAP summaries from `predict_mfrm_units()`.
- `summary()` reports draw counts and empirical draw summaries by person.

What this does not justify

This helper does not update the calibration, estimate new non-person facet levels, or provide exact future true values. It samples from the fixed-grid posterior implied by the existing fixed calibration.

References

The underlying posterior scoring follows the usual quadrature-based EAP framework of Bock and Aitkin (1981). The interpretation of multiple posterior draws as plausible-value-style summaries follows the general logic discussed by Mislevy (1991), while the current implementation remains a practical fixed-calibration approximation rather than a full published many-facet plausible-values method. For `JML` source fits, the quadrature posterior uses a package-level standard normal reference prior for this post hoc scoring layer.

- Bock, R. D., & Aitkin, M. (1981). *Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm*. Psychometrika, 46(4), 443-459.
- Mislevy, R. J. (1991). *Randomization-based inference about latent variables from complex samples*. Psychometrika, 56(2), 177-196.

See Also

predict_mfrm_units(), summary.mfrm_plausible_values

Examples

```
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- suppressWarnings(
  fit_mfrm(
    toy[toy$Person %in% keep_people, , drop = FALSE],
    "Person", c("Rater", "Criterion"), "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
)
new_units <- data.frame(
  Person = c("NEW01", "NEW01"),
  Rater = unique(toy$Rater)[1],
  Criterion = unique(toy$Criterion)[1:2],
  Score = c(2, 3)
)
pv <- sample_mfrm_plausible_values(toy_fit, new_units, n_draws = 3, seed = 1)
summary(pv)$draw_summary
```

---

shrinkage_report *Extract the shrinkage report from a fitted mfrm_fit*

Description

Lightweight accessor that returns the per-facet empirical-Bayes shrinkage table stored on a fit when facet_shrinkage != "none". Returns NULL (with a message) when no shrinkage has been applied so callers can probe without error.

Usage

```
shrinkage_report(fit)
```

Arguments

fit	An mfrm_fit object.

Value

A data.frame with one row per facet (and optionally "Person") or NULL when shrinkage has not been applied.

See Also

apply_empirical_bayes_shrinkage(), fit_mfrm().

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30,
                facet_shrinkage = "empirical_bayes")
shrinkage_report(fit)
```

---

simulate_mfrm_data *Simulate long-format ordered many-facet data for design studies*

Description

Simulate long-format ordered many-facet data for design studies

Usage

```
simulate_mfrm_data(
  n_person = 50,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = n_rater,
  design = NULL,
  score_levels = 4,
  theta_sd = 1,
  rater_sd = 0.35,
  criterion_sd = 0.25,
  noise_sd = 0,
  step_span = 1.4,
  group_levels = NULL,
  dif_effects = NULL,
  interaction_effects = NULL,
  seed = NULL,
  model = c("RSM", "PCM", "GPCM"),
  step_facet = "Criterion",
  slope_facet = NULL,
  thresholds = NULL,
  slopes = NULL,
  assignment = NULL,
  sparse_controls = NULL,
  sim_spec = NULL
)
```

Arguments

n_person	Number of persons/respondents.
n_rater	Number of rater facet levels.

`n_criterion` Number of criterion/item facet levels.

`raters_per_person`
Number of raters assigned to each person.

`design` Optional named design override supplied as a named list, named vector, or one-row data frame. When `sim_spec = NULL`, names may use canonical variables (`n_person`, `n_rater`, `n_criterion`, `raters_per_person`) or role keywords (`person`, `rater`, `criterion`, `assignment`). For the currently exposed facet keys, the schema-only future branch input `design$facets = c(person = ..., rater = ..., criterion = ...)` is also accepted. Do not specify the same variable through both `design` and the scalar count arguments.

`score_levels` Number of ordered score categories.

`theta_sd` Standard deviation of simulated person measures.

`rater_sd` Standard deviation of simulated rater severities.

`criterion_sd` Standard deviation of simulated criterion difficulties.

`noise_sd` Optional observation-level noise added to the linear predictor.

`step_span` Spread of step thresholds on the logit scale.

`group_levels` Optional character vector of group labels. When supplied, a balanced `Group` column is added to the simulated data.

`dif_effects` Optional data.frame describing true group-linked DIF effects. Must include `Group`, at least one design column such as `Criterion`, and numeric `Effect`.

`interaction_effects`
Optional data.frame describing true non-group interaction effects. Must include at least one design column such as `Rater` or `Criterion`, plus numeric `Effect`.

`seed` Optional random seed.

`model` Measurement model recorded in the simulation setup. The current public generator supports `RSM`, `PCM`, and bounded `GPCM`.

`step_facet` Step facet used when `model = "PCM"` and threshold values vary across levels. Currently `"Criterion"` and `"Rater"` are supported.

`slope_facet` Slope facet used when `model = "GPCM"`. The current bounded `GPCM` branch requires `slope_facet == step_facet`.

`thresholds` Optional threshold specification. Use a numeric vector of common thresholds; a named list such as `list(C01 = c(-1, 0, 1))`; a numeric matrix with one row per `StepFacet` and one column per step; or a long data frame with columns `StepFacet`, `Step`/`StepIndex`, and `Estimate`.

`slopes` Optional slope specification used when `model = "GPCM"`. Use either a numeric vector aligned to the generated slope-facet levels or a data frame with columns `SlopeFacet` and `Estimate`. Supplied slopes are treated as relative discriminations and normalized to the package's geometric-mean-one identification convention on the log scale. When omitted, slopes default to 1 for every slope-facet level, giving an exact `PCM` reduction.

`assignment` Assignment design. `"crossed"` means every person sees every rater; `"rotating"` uses a balanced rotating subset; `"resampled"` reuses person-level rater-assignment profiles stored in `sim_spec`; `"sparse_linked"` uses an incomplete rating design

with optional linking persons; "skeleton" reuses an observed response skeleton stored in sim_spec, including optional Group/Weight columns when available. When omitted, the function chooses "crossed" if raters_per_person == n_rater, otherwise "rotating".

sparse_controls
: Optional named list used when assignment = "sparse_linked". Supported entries are link_fraction, link_persons, link_raters_per_person, assignment_mode, and min_common_persons_per_rater_pair. See build_mfrm_sim_spec() for the same contract.

sim_spec
: Optional output from build_mfrm_sim_spec() or extract_mfrm_sim_spec(). When supplied, it defines the generator setup; direct scalar arguments are treated as legacy inputs and should generally be left at their defaults except for seed. Any custom public two-facet names recorded in sim_spec$facet_names are also carried into the simulated output and downstream planning helpers. If sim_spec stores an active latent-regression population generator, the returned object also carries the generated one-row-per-person background-data table needed to refit that population model later.

Details

This function generates synthetic ordered many-facet data under RSM, PCM, or the package's bounded GPCM branch. The data-generating process is:

1. Draw person abilities: $\theta_n \sim N(0, \texttt{theta\_sd}^2)$
2. Draw rater severities: $\delta_j \sim N(0, \texttt{rater\_sd}^2)$
3. Draw criterion difficulties: $\beta_i \sim N(0, \texttt{criterion\_sd}^2)$
4. Generate evenly-spaced step thresholds spanning $\pm$step_span/2
5. For each observation, compute the linear predictor $\eta = \theta_n - \delta_j - \beta_i + \epsilon$ where $\epsilon \sim N(0, \texttt{noise\_sd}^2)$ (optional)
6. Compute category probabilities under the recorded measurement model (RSM, PCM, or bounded GPCM) and sample the response

Latent-value generation is explicit:

- latent_distribution = "normal" draws centered normal person/rater/ criterion values using the supplied standard deviations
- latent_distribution = "empirical" resamples centered support values recorded in sim_spec$empirical_support
- if sim_spec$population$active = TRUE, person measures are generated from the stored latent-regression population model and template person covariates rather than from theta_sd

When dif_effects is supplied, the specified logit shift is added to η for the focal group on the target facet level, creating a known DIF signal. Similarly, interaction_effects injects a known bias into specific facet-level combinations.

The generator targets the common two-facet rating design (persons $\times$ raters $\times$ criteria). raters_per_person controls the incomplete-block structure: when less than n_rater, each person is assigned a rotating subset of raters to keep coverage balanced and reproducible.

Threshold handling is intentionally explicit:

- if thresholds = NULL, common equally spaced thresholds are generated from step_span
- if thresholds is a numeric vector, it is used as one common threshold set
- if thresholds is a named list, numeric matrix, or data frame, threshold values may vary by StepFacet (currently Criterion or Rater)

For bounded GPCM, the generator now requires an explicit slope contract in parallel with the threshold table. The current public branch keeps slope_facet == step_facet, normalizes supplied slopes to the same geometric-mean-one log-slope identification used by fit_mfrm(), and uses the internal category_prob_gpcm() helper for response sampling. Broader arbitrary-facet planning remains restricted until that slope-aware contract is generalized beyond the current role-based design, population-forecasting, diagnostic-screening, and signal-detection helpers.

Assignment handling is also explicit:

- "crossed" uses the full person x rater x criterion design
- "rotating" assigns a deterministic rotating subset of raters per person
- "sparse_linked" assigns most persons to an incomplete rater subset and assigns a configurable set of linking persons to a larger rater set
- "resampled" reuses empirical person-level rater profiles stored in sim_spec$assignment_profiles, optionally carrying over person-level Group
- "skeleton" reuses an observed person-by-rater-by-criterion response skeleton stored in sim_spec$design_skeleton, optionally carrying over Group and Weight

Sparse linked simulation is intended for planned-missing rating designs in which connectivity is maintained through common linking persons. The returned mfrm_sparse_design attribute summarizes design density, planned missingness, rater coverage, and rater-pair common-person counts. These summaries are design diagnostics, not model-fit statistics or universal adequacy thresholds. This branch follows sparse rater-mediated assessment design work by Wind, Jones, and Grajeda (2023, doi:10.1177/01466216231182148), Wind and Jones (2018, doi:10.1177/0013164417703733), and DeMars, Shapovalov, and Hathcoat (2023).

For more controlled workflows, build a reusable simulation specification first via build_mfrm_sim_spec() or derive one from an observed fit with extract_mfrm_sim_spec(), then pass it through sim_spec.

Returned data include attributes:

- mfrm_truth: simulated true parameters (for parameter-recovery checks)
- mfrm_truth$signals: injected DIF and interaction signal tables
- mfrm_truth$slope_table: simulated discrimination table for bounded GPCM
- mfrm_population_data: generated one-row-per-person background data when the simulation specification stores an active latent-regression generator, including model-matrix xlevel and contrast provenance for categorical covariates
- mfrm_simulation_spec: generation settings (for reproducibility)
- mfrm_sparse_design: sparse-design diagnostics when assignment = "sparse_linked", including design density, planned missing rate, rater coverage, and rater-pair common-person counts

Value

A long-format `data.frame` with core columns `Study`, `Person`, two simulated non-person facet columns, and `Score`. By default those facet columns are `Rater` and `Criterion`; when `sim_spec` records custom public names, those names are used instead. If group labels are simulated or reused from an observed response skeleton, a `Group` column is included. If a weighted response skeleton is reused, a `Weight` column is also included.

Interpreting output

- Higher `theta` values in `mfrm_truth$person` indicate higher person measures.
- Higher values in `mfrm_truth$facets$Rater` indicate more severe raters.
- Higher values in `mfrm_truth$facets$Criterion` indicate more difficult criteria.
- `mfrm_truth$signals$dif_effects` and `mfrm_truth$signals$interaction_effects` record any injected detection targets.

Typical workflow

1. Generate one design with `simulate_mfrm_data()`.
2. Fit with `fit_mfrm()` and diagnose with `diagnose_mfrm()`.
3. For repeated design studies, use `evaluate_mfrm_design()`.

See Also

`evaluate_mfrm_design()`, `fit_mfrm()`, `diagnose_mfrm()`

Examples

```
sim <- simulate_mfrm_data(
  n_person = 40,
  n_rater = 4,
  n_criterion = 4,
  raters_per_person = 2,
  seed = 123
)
head(sim)
names(attr(sim, "mfrm_truth"))
```

---

specifications_report *Build a specification summary report (preferred alias)*

Description

Build a specification summary report (preferred alias)

Usage

```
specifications_report(
  fit,
  title = NULL,
  data_file = NULL,
  output_file = NULL,
  include_fixed = FALSE
)
```

Arguments

fit	Output from fit_mfrm().
title	Optional analysis title.
data_file	Optional data-file label (for reporting only).
output_file	Optional output-file label (for reporting only).
include_fixed	If TRUE, include a legacy-compatible fixed-width text block.

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_specifications (type = "facet_elements", "anchor_constraints", "convergence").

Value

A named list with specification-report components. Class: mfrm_specifications.

Interpreting output

- header / data_spec: run identity and model settings.
- facet_labels: facet sizes and labels.
- convergence_control: optimizer configuration and status.

Typical workflow

1. Generate specifications_report(fit).
2. Verify model settings and convergence metadata.
3. Use the output as methods and run-documentation support in reports.

See Also

fit_mfrm(), data_quality_report(), estimation_iteration_report(), mfrmr_reports_and_tables, mfrmr_compatibility_layer

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- specifications_report(fit, title = "Toy run")
summary(out)
p_spec <- plot(out, draw = FALSE)
p_spec$data$plot
```

---

subset_connectivity_report

Build a subset connectivity report (preferred alias)

Description

Build a subset connectivity report (preferred alias)

Usage

```
subset_connectivity_report(
  fit,
  diagnostics = NULL,
  top_n_subsets = NULL,
  min_observations = 0
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
top_n_subsets	Optional maximum number of subset rows to keep.
min_observations	Minimum observations required to keep a subset row.

Details

summary(out) is supported through summary(). plot(out) is dispatched through plot() for class mfrm_subset_connectivity (type = "subset_observations", "facet_levels", or "linking_matrix" / "coverage_matrix" / "design_matrix" / "network"). The network route returns reusable node and edge tables with draw = FALSE; drawing uses igraph when available.

Value

A named list with subset-connectivity components. Class: mfrm_subset_connectivity.

Interpreting output

- summary: number and size of connected subsets.
- subset table: whether data are fragmented into disconnected components.
- facet-level columns: where connectivity bottlenecks occur.

Typical workflow

1. Run subset_connectivity_report(fit).
2. Confirm near-single-subset structure when possible.
3. Use results to justify linking/anchoring strategy.

See Also

diagnose_mfrm(), mfrm_network_analysis(), measurable_summary_table(), data_quality_report(), mfrmr_linking_and_dff, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
out <- subset_connectivity_report(fit)
summary(out)
p_sub <- plot(out, draw = FALSE)
p_design <- plot(out, type = "design_matrix", draw = FALSE)
p_net <- plot(out, type = "network", draw = FALSE)
p_sub$data$plot
p_design$data$plot
p_net$data$edges
out$summary[, c("Subset", "Observations", "ObservationPercent")]
```

---

summary.apa_table *Summarize an APA/FACETS table object*

Description

Summarize an APA/FACETS table object

Usage

```
## S3 method for class 'apa_table'
summary(object, digits = 3, top_n = 8, ...)
```

Arguments

object	Output from apa_table().
digits	Number of digits used for numeric summaries.
top_n	Maximum numeric columns shown in numeric_profile.
...	Reserved for generic compatibility.

Details

Compact summary helper for QA of table data before manuscript export.

Value

An object of class `summary.apa_table`.

Interpreting output

- `overview`: table size/composition and missingness.
- `numeric_profile`: quick distribution summary of numeric columns.
- `caption/note`: text metadata readiness.

Typical workflow

1. Build table with `apa_table()`.
2. Run `summary(tbl)` and inspect `overview`.
3. Use `plot.apa_table()` for quick numeric checks if needed.

See Also

`apa_table()`, `plot()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
tbl <- apa_table(fit, which = "summary")
summary(tbl)
```

---

summary.mfrm_anchor_review

Summarize an anchor-review object

Description

Summarize an anchor-review object

Usage

```
## S3 method for class 'mfrm_anchor_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from review_mfrm_anchors().
digits	Number of digits for numeric rounding.
top_n	Maximum rows shown in issue previews.
...	Reserved for generic compatibility.

Details

This summary provides a compact pre-estimation review of anchor and group-anchor specifications.

Value

An object of class summary.mfrm_anchor_review.

Interpreting output

Recommended order:

- issue_counts: primary triage table (non-zero issues first).
- facet_summary: anchored/grouped/free-level balance by facet.
- level_observation_summary and category_counts: sparse-cell diagnostics.
- recommendations: concrete remediation suggestions.

If issue_counts is non-empty, treat anchor constraints as provisional and resolve issues before final estimation.

Typical workflow

1. Run review_mfrm_anchors() with intended anchors/group anchors.
2. Review summary(review) and recommendations.
3. Revise anchor tables, then call fit_mfrm().

See Also

review_mfrm_anchors(), fit_mfrm()

Examples

```
toy <- load_mfrmr_data("example_core")
review <- review_mfrm_anchors(toy, "Person", c("Rater", "Criterion"), "Score")
summary(review)
```

summary.mfrm_apa_outputs *Summarize APA report-output bundles*

Description

Summarize APA report-output bundles

Usage

```
## S3 method for class 'mfrm_apa_outputs'
summary(object, top_n = 3, preview_chars = 160, ...)
```

Arguments

object	Output from build_apa_outputs().
top_n	Maximum non-empty lines shown in each component preview.
preview_chars	Maximum characters shown in each preview cell.
...	Reserved for generic compatibility.

Details

This summary is a diagnostics layer for APA text products, not a replacement for the full narrative.

It reports component completeness, line/character volume, and a compact preview for quick QA before manuscript insertion.

Value

An object of class summary.mfrm_apa_outputs.

Interpreting output

- overview: total coverage across standard text components.
- components: per-component density and mention checks (including residual-PCA mentions).
- sections: package-native section coverage table.
- content_checks: contract-based alignment checks for APA drafting readiness.
- overview$DraftContractPass: the primary contract-completeness flag for draft text components.
- overview$ReadyForAPA: a backward-compatible alias of that contract flag, not a certification of inferential adequacy.
- preview: first non-empty lines for fast visual review.

Typical workflow

1. Build outputs via `build_apa_outputs()`.
2. Run `summary(apa)` to screen for empty/short components.
3. Use `apa$report_text`, `apa$table_figure_notes`, and `apa$table_figure_captions` as draft components for final-text review.

See Also

`build_apa_outputs()`, `summary()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
apa <- build_apa_outputs(fit, diag)
summary(apa)
```

---

summary.mfrm_bias *Summarize an* mfrm_bias *object in a user-friendly format*

Description

Summarize an `mfrm_bias` object in a user-friendly format

Usage

```
## S3 method for class 'mfrm_bias'
summary(object, digits = 3, top_n = 10, p_cut = 0.05, ...)
```

Arguments

`object`	Output from `estimate_bias()`.
`digits`	Number of digits for printed numeric values.
`top_n`	Number of strongest bias rows to keep.
`p_cut`	Significance cutoff used for counting flagged rows.
`...`	Reserved for generic compatibility.

Details

This method returns a compact interaction-bias summary:

- interaction facets/order and analyzed cell counts
- effect-size profile (`|bias|` mean/max, significant cell count)
- fixed-effect chi-square block
- iteration-end convergence indicators
- top rows ranked by absolute t

Value

An object of class `summary.mfrm_bias` with:

- `overview`: interaction facets/order, cell counts, and effect-size profile
- `chi_sq`: fixed-effect chi-square block
- `final_iteration`: end-of-iteration status row
- `top_rows`: highest-`|t|` interaction rows
- `notes`: short interpretation notes

Interpreting output

- `overview`: interaction order, analyzed cells, and effect-size profile.
- `chi_sq`: fixed-effect test block.
- `final_iteration`: end-of-loop status from the bias routine.
- `top_rows`: strongest bias contrasts by `|t|`; bounded `GPCM` summaries also retain the profile-likelihood review columns when present.

Typical workflow

1. Estimate interactions with `estimate_bias()`.
2. Check `summary(bias)` for screen-positive and unstable cells.
3. Use `bias_interaction_report()` or `plot_bias_interaction()` for details.

See Also

`estimate_bias()`, `bias_interaction_report()`

Examples

```
toy <- load_mfrmr_data("example_bias")
toy <- toy[toy$Person %in% unique(toy$Person)[1:8], ]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 1)
summary(bias)
```

summary.mfrm_bundle *Summarize report/table bundles in a user-friendly format*

Description

Summarize report/table bundles in a user-friendly format

Usage

```
## S3 method for class 'mfrm_bundle'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Any report bundle produced by mfrmr table/report helpers.
digits	Number of digits for printed numeric values.
top_n	Number of preview rows shown from the main table component.
...	Reserved for generic compatibility.

Details

This method provides a compact summary for bundle-like outputs (for example: unexpected-response, fair-average, chi-square, and category report objects). It extracts:

- object class and available components
- one-row summary table when available
- preview rows from the main data component
- resolved settings/options

Branch-aware summaries are provided for:

- mfrm_bias_count (branch = "original" / "facets")
- mfrm_fixed_reports (branch = "original" / "facets")
- mfrm_visual_summaries (branch = "original" / "facets")

Additional class-aware summaries are provided for:

- mfrm_unexpected, mfrm_fair_average, mfrm_displacement
- mfrm_interrater, mfrm_facets_chisq, mfrm_bias_interaction
- mfrm_rating_scale, mfrm_category_structure, mfrm_category_curves
- mfrm_measurable, mfrm_unexpected_after_bias, mfrm_output_bundle
- mfrm_residual_pca, mfrm_specifications, mfrm_data_quality, mfrm_fit_measures
- mfrm_iteration_report, mfrm_subset_connectivity, mfrm_facet_statistics
- mfrm_facets_contract_review, mfrm_facets_fit_review, mfrm_facets_fit_df_guide, mfrm_reference_benchmark

Value

An object of class `summary.mfrm_bundle`.

Interpreting output

- `overview`: class, component count, and selected preview component.
- `summary`: one-row aggregate block when supplied by the bundle.
- `preview`: first `top_n` rows from the main table-like component.
- `settings`: resolved option values if available.
- `validation_scope`: internal-versus-external validation scope when summarizing `mfrm_reference_benchmark`.
- `conquest_command_scope`: ConQuest command-template scope when summarizing `mfrm_conquest_overlap_bundl`
- `conquest_output_contract`: requested ConQuest outputs and review handoff when summarizing `mfrm_conquest_overlap_bundle`.
- `normalization_scope`: extracted-table normalization scope when summarizing `mfrm_conquest_overlap_tables`.
- `review_scope`: supplied-table review scope when summarizing `mfrm_conquest_overlap_review`.
- `conquest_overlap_checks` / `population_policy_checks`: specialized benchmark check previews when summarizing `mfrm_reference_benchmark`.

Typical workflow

1. Generate a bundle table/report helper output.
2. Run `summary(bundle)` for compact QA.
3. Drill into specific components via `$` and visualize with `plot(bundle, ...)`.

See Also

`unexpected_response_table()`, `fair_average_table()`, `plot()`

Examples

```
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
t4 <- unexpected_response_table(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 5)
summary(t4)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t11 <- bias_count_table(bias, branch = "facets")
summary(t11)
```

summary.mfrm_data_description
Summarize a data-description object

Description

Summarize a data-description object

Usage

```
## S3 method for class 'mfrm_data_description'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from describe_mfrm_data().
digits	Number of digits for numeric rounding.
top_n	Maximum rows shown in preview blocks.
...	Reserved for generic compatibility.

Details

This summary is intended as a compact pre-fit quality snapshot for manuscripts and analysis logs.

Value

An object of class summary.mfrm_data_description.

- overview: design/sample counts
- missing: top columns by missingness
- score_distribution: compact score-usage table, including zero-count categories retained by the prepared score support
- facet_overview: facet-level coverage summary
- agreement: inter-rater agreement summary when available
- row_retention: row counts before and after preparation filters
- preparation_notes: structured preparation notes retained from describe_mfrm_data()
- reporting_map: manuscript-oriented guide to what is covered here versus which companion outputs should be consulted
- caveats: structured warning/review rows for score-support issues; print(summary(ds)) shows a compact Caveats block when rows are present

Interpreting output

Recommended read order:

- `overview`: sample size, persons/facets/categories.
- `missing`: missingness hotspots by selected input columns.
- `score_distribution`: category usage balance.
- `notes` / printed `Caveats`: retained zero-count score categories and related score-support caveats; intermediate unused categories should be treated as threshold-functioning warnings before model fitting.
- `facet_overview`: coverage per facet (minimum/maximum weighted counts).
- `agreement`: observed-score inter-rater agreement (when available).

Very low `MinWeightedN` in `facet_overview` is a practical warning for unstable downstream facet estimates.

Typical workflow

1. Run `describe_mfrm_data()` on raw long-format data.
2. Inspect `summary(ds)` before model fitting.
3. Resolve sparse/missing issues, then run `fit_mfrm()`.

See Also

`describe_mfrm_data()`, `summary.mfrm_fit()`

Examples

```
toy <- load_mfrmr_data("example_core")
ds <- describe_mfrm_data(toy, "Person", c("Rater", "Criterion"), "Score")
summary(ds)
```

---

summary.mfrm_design_evaluation

Summarize a design-simulation study

Description

Summarize a design-simulation study

Usage

```
## S3 method for class 'mfrm_design_evaluation'
summary(object, digits = 3, ...)
```

Arguments

`object`	Output from `evaluate_mfrm_design()`.
`digits`	Number of digits used in the returned numeric summaries.
`...`	Reserved for generic compatibility.

Details

The summary emphasizes condition-level averages that are useful for practical design planning, especially:

- convergence rate
- separation and reliability by facet
- severity recovery RMSE
- mean misfit rate

Value

An object of class `summary.mfrm_design_evaluation` with components:

- `overview`: run-level overview
- `design_summary`: aggregated design-by-facet metrics, with design-variable alias columns when applicable
- `sparse_review`: compact planned-missingness and rater-link review counts when sparse linked designs are active
- `ademp`: simulation-study metadata carried forward from the original object
- `facet_names`: public facet labels carried from the simulation specification
- `design_variable_aliases`: accepted public aliases for design variables
- `design_descriptor`: role-based design-variable metadata
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of mutable/locked design variables
- `planning_schema`: combined planner-schema contract
- `future_branch_active_summary`: compact deterministic summary of the schema-only future arbitrary-facet planning branch embedded in the current planning schema
- `notes`: short interpretation notes

See Also

`evaluate_mfrm_design()`, plot.mfrm_design_evaluation

Examples

```
sim_eval <- suppressWarnings(evaluate_mfrm_design(
  n_person = c(8, 12),
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 1,
  reps = 1,
  maxit = 30,
  seed = 123
))
s <- summary(sim_eval)
s$overview
head(s$design_summary)
```

---

summary.mfrm_diagnostics

Summarize an `mfrm_diagnostics` *object in a user-friendly format*

Description

Summarize an `mfrm_diagnostics` object in a user-friendly format

Usage

```
## S3 method for class 'mfrm_diagnostics'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`object`	Output from `diagnose_mfrm()`.
`digits`	Number of digits for printed numeric values.
`top_n`	Number of highest-absolute-Z fit rows to keep.
`...`	Reserved for generic compatibility.

Details

This method returns a compact diagnostics summary designed for quick review:

- design overview (observations, persons, facets, categories, subsets)
- diagnostic-basis guide for legacy versus strict fit paths
- global fit statistics
- approximate reliability/separation by facet
- top facet/person fit rows by absolute ZSTD
- counts of flagged diagnostics (unexpected, displacement, interactions)

Value

An object of class `summary.mfrm_diagnostics` with:

- `overview`: design-level counts and residual-PCA mode
- `status`: concise front-door status block for quick review
- `key_warnings`: highest-priority warnings to review first
- `next_actions`: recommended follow-up helpers
- `diagnostic_basis`: guide to legacy versus strict diagnostic targets
- `fit_standardization`: guide to the df convention used for fit ZSTD
- `overall_fit`: global fit block
- `precision_profile`: design-weighted precision summary across the information curve at decile theta points
- `precision_review`: separation / reliability / strata review for the sample- and population-basis modes (paired with `precision_profile`)
- `reliability`: facet-level separation/reliability summary
- `facets_chisq`: facets-style fixed-effect chi-square heterogeneity screen across non-person facets
- `interrater`: inter-rater agreement / pairwise correlation / rater separation overview when a Rater facet is present
- `misfit_flagged`: rows flagged by the Infit / Outfit / ZSTD misfit thresholds active for this fit
- `misfit_thresholds`: named numeric vector with the misfit `lower` / `upper` thresholds used to populate `misfit_flagged`
- `category_usage`: per-category response-frequency summary used to flag empty / collapsed categories
- `top_fit`: top `|ZSTD|` rows
- `marginal_fit`: optional strict marginal-fit overview when requested
- `top_marginal_cells`: largest strict marginal residual cells when requested
- `marginal_pairwise`: optional strict pairwise local-dependence overview
- `top_marginal_pairs`: largest strict pairwise residual summaries
- `marginal_guidance`: interpretation labels for strict marginal diagnostics
- `reporting_map`: manuscript-oriented guide to what is covered here versus which companion outputs should be consulted
- `flags`: compact flag counts for major diagnostics
- `notes`: short interpretation notes
- `digits`: numeric-print precision threaded through to `print.summary.mfrm_diagnostics()`

Interpreting output

- `overview`: analysis scale, subset count, and residual-PCA mode.
- `diagnostic_basis`: plain-language map of which fit path was computed and what each path means statistically.
- `overall_fit`: global fit indices.
- `reliability`: facet separation/reliability block, including model and real bounds when available.
- `top_fit`: highest `|ZSTD|` elements for immediate inspection.
- `flags`: compact counts for key warning domains.

Typical workflow

1. Run diagnostics with `diagnose_mfrm()`, using `diagnostic_mode = "both"` for `RSM` / `PCM` when you want legacy continuity plus strict marginal screening.
2. Review `summary(diag)` for major warnings and inspect `diagnostic_basis` before comparing legacy and strict outputs.
3. Follow up with dedicated tables/plots for flagged domains.

See Also

`diagnose_mfrm()`, `summary.mfrm_fit()`

Examples

```
toy <- load_mfrmr_data("example_core")
toy <- toy[toy$Person %in% unique(toy$Person)[1:4], ]
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
s <- summary(diag, top_n = 3)
s$key_warnings
# Look for: lines beginning with "MnSq misfit:" name the worst
#   element + Infit / Outfit values; "Unexpected responses flagged"
#   counts how many cell-level surprises the screen returned.
s$top_fit
# Look for: rows with |InfitZSTD| or |OutfitZSTD| > 2 are misfitting
#   at the 5% level; > 3 is misfitting at the 1% level. Investigate
#   in order of the AbsZ column.
s$facets_chisq
# Look for: FixedProb < 0.05 in each non-Person facet means the
#   facet contributes meaningful spread; FixedProb >= 0.05 means
#   that facet is statistically indistinguishable.
```

summary.mfrm_diagnostic_screening *Summarize a diagnostic-screening validation study*

Description

Summarizes output from evaluate_mfrm_diagnostic_screening() for reporting, appendix export, and draw-free visualization handoff. The summary keeps simulation operating characteristics separate from validation gates: fit, marginal, pairwise, and report-review signals are screening readouts rather than pass/fail evidence.

Usage

```
## S3 method for class 'mfrm_diagnostic_screening'
summary(object, digits = 3, ...)
```

Arguments

object	Output from evaluate_mfrm_diagnostic_screening().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_diagnostic_screening with:

- overview: run-level design, replication, convergence, and report-review metadata
- reading_order: recommended order for reading the summary tables
- next_actions: action-oriented triage for interpreting and exporting the summary
- reporting_notes: report-facing boundaries and recommended wording safeguards
- figure_recipes: recommended figure/display recipes for the draw-free plot-data tables
- scenario_summary: aggregated scenario-by-design screening summaries
- performance_summary: operating-characteristic rates and runtime summaries
- report_signal_summary: optional mfrm_report() readiness/review signals
- scenario_contrast: misspecification-minus-well-specified contrasts
- plot_*: long-form draw-free plot tables for overview, report, contrast, and runtime views
- planning metadata, settings, ADEMP metadata, and interpretation notes

See Also

evaluate_mfrm_diagnostic_screening(), plot.mfrm_diagnostic_screening, plot_data()

Examples

```
diag_eval <- evaluate_mfrm_diagnostic_screening(
  design = list(person = 10, rater = 2, criterion = 2, assignment = 2),
  reps = 1,
  maxit = 30,
  seed = 123
)
summary(diag_eval)
```

---

summary.mfrm_facets_run

Summarize a legacy-compatible workflow run

Description

Summarize a legacy-compatible workflow run

Usage

```
## S3 method for class 'mfrm_facets_run'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from run_mfrm_facets().
digits	Number of digits for numeric rounding in summaries.
top_n	Maximum rows shown in nested preview tables.
...	Passed through to nested summary methods.

Details

This method returns a compact cross-object summary that combines:

- model overview (objectfitsummary)
- resolved column mapping
- run settings (run_info)
- nested summaries of fit and diagnostics

Value

An object of class summary.mfrm_facets_run.

Interpreting output

- `overview`: convergence, information criteria, and scale size.
- `mapping`: sanity check for auto/explicit column mapping.
- `fit` / `diagnostics`: drill-down summaries for reporting decisions.

Typical workflow

1. Run `run_mfrm_facets()` to execute a one-shot pipeline.
2. Inspect with `summary(out)` for mapping and convergence checks.
3. Review nested objects (`out$fit`, `out$diagnostics`) as needed.

See Also

`run_mfrm_facets()`, `summary.mfrm_fit()`, mfrmr_workflow_methods, `summary()`

Examples

```
toy <- load_mfrmr_data("example_core")
toy_small <- toy[toy$Person %in% unique(toy$Person)[1:8], , drop = FALSE]
out <- run_mfrm_facets(
  data = toy_small,
  person = "Person",
  facets = c("Rater", "Criterion"),
  score = "Score",
  maxit = 30
)
s <- summary(out)
s$overview[, c("Model", "Method", "Converged")]
s$mapping
```

---

summary.mfrm_facet_dashboard

Summarize a facet-quality dashboard

Description

Summarize a facet-quality dashboard

Usage

```
## S3 method for class 'mfrm_facet_dashboard'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from facet_quality_dashboard().
digits	Number of digits for printed numeric values.
top_n	Number of flagged levels to preview.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_facet_dashboard.

See Also

facet_quality_dashboard(), plot_facet_quality_dashboard()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
summary(facet_quality_dashboard(fit, diagnostics = diag))
```

---

summary.mfrm_fit *Summarize an* mfrm_fit *object in a user-friendly format*

Description

Summarize an mfrm_fit object in a user-friendly format

Usage

```
## S3 method for class 'mfrm_fit'
summary(object, digits = 3, top_n = 5, ...)
```

Arguments

object	Output from fit_mfrm().
digits	Number of digits for printed numeric values.
top_n	Number of extreme facet/person rows shown in summaries.
...	Reserved for generic compatibility.

Details

This method provides a compact, human-readable summary oriented to reporting. It returns a structured object and prints:

- model fit overview (N, LogLik, AIC/BIC, convergence)
- estimation settings that affect identification/scoring interpretation
- facet-level estimate distribution (mean/SD/range)
- person measure distribution
- step/threshold checks
- a reporting map showing which companion summaries/tables should be used for manuscript-oriented data description, diagnostics, category checks, and draft reporting
- high/low person measures and extreme facet levels

Value

An object of class `summary.mfrm_fit` with:

- `overview`: global model/fit indicators
- `status`: concise front-door status block for quick review
- `key_warnings`: highest-priority warnings to review first
- `next_actions`: recommended follow-up helpers
- `population_overview`: current population-model basis, residual variance, and omission review
- `population_coefficients`: fitted latent-regression coefficients when a population model is active
- `population_design`: latent-regression design-matrix column check when a population model is active
- `population_coding`: categorical covariate levels and contrast provenance when a population model uses model-matrix coding
- `facet_overview`: per-facet estimate distribution summary
- `person_overview`: person-measure distribution summary
- `targeting`: person-versus-non-person facet targeting overview (Wright-map-style mean/SD comparison)
- `step_overview`: threshold/step diagnostics
- `slope_overview`: discrimination summary for `GPCM` fits
- `interaction_overview`: model-estimated facet-interaction summary when the fit was specified with `facet_interactions`
- `settings_overview`: estimation-settings overview that pins the configuration that affects identification/scoring
- `attached_diagnostics`: logical flag indicating whether the `mfrm_fit` was returned with diagnostics already attached

- attached_diagnostics_cols: character vector of diagnostic columns attached to fit$facets$person when attached_diagnostics = TRUE
- row_retention: row counts before and after preparation filters
- preparation_notes: structured preparation notes retained from fit$prep
- reporting_map: routing map showing which companion summaries and tables should be used for the four manuscript-oriented reporting sections (data description, diagnostics, category checks, draft reporting)
- person_high / person_low: highest and lowest person measures
- facet_extremes: extreme facet-level estimates
- caveats: structured warning/review rows for score-support and latent-regression population-model issues
- notes: short interpretation notes
- digits: numeric-print precision threaded through to print.summary.mfrm_fit()

Interpreting output

- overview: convergence and information criteria.
- facet_overview: per-facet spread and range of estimates.
- person_overview: distribution of person measures.
- step_overview: threshold spread and monotonicity checks.
- settings_overview: estimation settings that affect interpretation.
- population_coding: fitted categorical levels and contrasts that must be reused when scoring new persons under the population-model posterior.
- key_warnings / notes: short triage subset of retained zero-count score categories and latent-regression population-model caveats such as complete-case omissions, zero-variance design columns, missing coefficients, or unstable residual variance when present. Incomplete or non-finite covariates are normally handled before fitting as input errors or complete-case omissions; they appear here only if retained in a population-design check row.
- caveats: structured rows behind those warnings for appendix/export use; print(summary(fit)) shows a compact Caveats block when rows are present.
- reporting_map: where to get companion outputs for manuscript reporting.
- top_person / top_facet: extreme estimates for quick triage.

Typical workflow

1. Fit model with fit_mfrm().
2. Run summary(fit) for first-pass diagnostics.
3. For RSM / PCM, continue with diagnose_mfrm() for element-level fit checks. For bounded GPCM, continue with compute_information() / plot_information() or the fixed-calibration posterior scoring helpers.

See Also

fit_mfrm(), diagnose_mfrm()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(
  toy, "Person", c("Rater", "Criterion"), "Score",
  method = "MML", quad_points = 5
)
s <- summary(fit)
s$overview[, c("Model", "Method", "Converged")]
# Look for: Converged = TRUE. If FALSE the fit is not safe to report;
#   raise `maxit`, relax `reltol`, or rerun with `quad_points = 31`.
s$person_overview
# Look for: Mean ~ 0 (logits) and SD ~ 1 are typical when the sample
#   is centred on the test difficulty. Min < -3 or Max > 3 with
#   `Extreme = "min"/"max"` rows indicates ceiling / floor cases.
s$targeting
# Look for: |Targeting| < ~0.5 logits across non-person facets is
#   comfortable. Larger absolute values mean the test is systematically
#   easier or harder than the person sample. SpreadRatio > 2 means
#   persons dominate facet variability; < 0.5 means facets dominate.
```

---

summary.mfrm_future_branch_active_branch

Summarize a future arbitrary-facet planning active branch

Description

Summarize a future arbitrary-facet planning active branch

Usage

```
## S3 method for class 'mfrm_future_branch_active_branch'
summary(object, digits = 3, top_n = 8, ...)
```

Arguments

`object`	Output from the future-branch active planning scaffold stored in `planning_schema$future_branch_act`
`digits`	Number of digits used in numeric summaries.
`top_n`	Maximum number of recommendation rows to print in the preview.
`...`	Reserved for generic compatibility.

Details

This summary is intentionally conservative. It aggregates only deterministic branch-side quantities already validated in the schema-first arbitrary-facet planning scaffold: observation bookkeeping, load/balance, coverage, guardrails, structural readiness, and conservative recommendation ranking. It also exposes the same manuscript-facing table/appendix metadata used by `build_summary_table_bundle()` so the future branch can be reviewed directly without first routing through planning summaries.

In addition to bundle-level appendix presets and section counts, it includes export-like appendix selection summaries by preset, reporting role, manuscript section, bundle-aware handoff summaries, preset-specific table surface, and a table-level handoff crosswalk, plus direct `role_summary` / `table_profile` surfaces for table-shape review. It does not report psychometric recovery or Monte Carlo performance.

Value

An object of class `summary.mfrm_future_branch_active_branch`.

See Also

`summary.mfrm_design_evaluation()`, `plot.mfrm_future_branch_active_branch()`

---

summary.mfrm_linking_review

Summarize a linking-review object

Description

Summarize a linking-review object

Usage

```
## S3 method for class 'mfrm_linking_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`object`	Output from `build_linking_review()`.
`digits`	Number of digits for printed numeric values.
`top_n`	Number of top linking-risk rows to keep in the compact summary.
`...`	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_linking_review`.

See Also

`build_linking_review()`

summary.mfrm_misfit_casebook

Summarize a misfit-casebook object

Description

Summarize a misfit-casebook object

Usage

```
## S3 method for class 'mfrm_misfit_casebook'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from build_misfit_casebook().
digits	Number of digits for printed numeric values.
top_n	Number of top case rows to keep in the compact summary.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_misfit_casebook.

See Also

build_misfit_casebook()

summary.mfrm_model_choice_review

Summarize a model-choice review

Description

Summarize a model-choice review

Usage

```
## S3 method for class 'mfrm_model_choice_review'
summary(object, digits = 3, ...)
```

Arguments

object	Output from build_model_choice_review().
digits	Number of digits for printed numeric values.
...	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_model_choice_review`.

See Also

`build_model_choice_review()`

`summary.mfrm_network_review`

Summarize an MFRM network review

Description

Summarize an MFRM network review

Usage

```
## S3 method for class 'mfrm_network_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

`object`	Output from `build_mfrm_network_review()`.
`digits`	Number of digits for printed numeric values.
`top_n`	Number of central/cut/bridge rows to keep in the compact summary.
`...`	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_network_review`.

See Also

`build_mfrm_network_review()`

summary.mfrm_peer_review_design_review

Summarize a peer-review design review

Description

Summarize a peer-review design review

Usage

```
## S3 method for class 'mfrm_peer_review_design_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from build_peer_review_design_review().
digits	Number of digits for printed numeric values.
top_n	Number of rows to keep in compact follow-up tables.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_peer_review_design_review.

See Also

build_peer_review_design_review()

summary.mfrm_person_fit_indices

Summarize person-fit indices

Description

summary() for compute_person_fit_indices() output gives a compact, report-ready reading order: first the number of persons and flags, then status counts, then the highest-priority review rows. The summary keeps lz_star availability visible so users do not silently treat uncorrected lz as Snijders-corrected output.

Usage

```
## S3 method for class 'mfrm_person_fit_indices'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from compute_person_fit_indices().
digits	Number of digits used when printing numeric columns.
top_n	Number of review rows retained in top_review.
...	Unused.

Value

An object of class summary.mfrm_person_fit_indices with:

- overview
- status_summary
- report_index_summary
- lz_star_status_summary
- top_review
- caveats
- thresholds
- reporting_map
- notes

---

summary.mfrm_plausible_values

Summarize approximate plausible values from posterior scoring

---

Description

Summarize approximate plausible values from posterior scoring

Usage

```
## S3 method for class 'mfrm_plausible_values'
summary(object, digits = 3, ...)
```

Arguments

object	Output from sample_mfrm_plausible_values().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_plausible_values` with:

- `draw_summary`: empirical summaries of the sampled values by person
- `estimates`: companion posterior EAP summaries
- `row_review`: row-preparation review
- `population_review`: optional person-level omission review for latent-regression scoring
- `settings`: scoring settings
- `notes`: interpretation notes

See Also

`sample_mfrm_plausible_values()`

Examples

```
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- suppressWarnings(
  fit_mfrm(
    toy[toy$Person %in% keep_people, , drop = FALSE],
    "Person", c("Rater", "Criterion"), "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
)
new_units <- data.frame(
  Person = c("NEW01", "NEW01"),
  Rater = unique(toy$Rater)[1],
  Criterion = unique(toy$Criterion)[1:2],
  Score = c(2, 3)
)
pv <- sample_mfrm_plausible_values(toy_fit, new_units, n_draws = 3, seed = 1)
summary(pv)
```

---

summary.mfrm_population_prediction

Summarize a population-level design forecast

Description

Summarize a population-level design forecast

Usage

```
## S3 method for class 'mfrm_population_prediction'
summary(object, digits = 3, ...)
```

Arguments

object	Output from predict_mfrm_population().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_population_prediction with:

- design: requested future design
- overview: run-level overview
- forecast: facet-level forecast table
- facet_names: public non-person facet names used in the forecast
- design_variable_aliases: public aliases for design variables
- design_descriptor: role-based description of design variables
- planning_scope: explicit record of the current planning contract
- planning_constraints: explicit record of mutable/locked design variables
- planning_schema: combined planner-schema contract
- gpcm_boundary: bounded-GPCM caveat row when present
- future_branch_active_summary: compact deterministic summary of the schema-only future arbitrary-facet planning branch embedded in the current planning schema
- ademp: simulation-study metadata
- notes: interpretation notes

See Also

predict_mfrm_population()

Examples

```
spec <- build_mfrm_sim_spec(
  n_person = 16,
  n_rater = 3,
  n_criterion = 2,
  raters_per_person = 2,
  assignment = "rotating"
)
pred <- predict_mfrm_population(
  sim_spec = spec,
  design = list(person = 18),
  reps = 1,
  maxit = 30,
  seed = 123
)
s <- summary(pred)
s$overview
s$forecast[, c("Facet", "MeanSeparation", "McseSeparation")]
```

summary.mfrm_reporting_checklist *Summarize a reporting-checklist bundle for manuscript work*

Description

Summarize a reporting-checklist bundle for manuscript work

Usage

```
## S3 method for class 'mfrm_reporting_checklist'
summary(object, top_n = 10, ...)
```

Arguments

object	Output from reporting_checklist().
top_n	Maximum number of draft-action rows shown in the compact action table.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_reporting_checklist with:

- overview: run-level counts of available and draft-ready items
- section_summary: section-level checklist coverage
- software_scope: external-software relationship summary
- facets_positioning: report-ready FACETS relationship wording
- visual_scope: plotting-route and 3D-ready data-handoff summary, including the main InterpretationCheck caveat for each visual family
- priority_summary: counts by priority/severity
- action_items: highest-priority rows that still need draft work
- settings: checklist settings rendered as a compact table
- notes: interpretation notes

See Also

reporting_checklist(), summary.mfrm_apa_outputs

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "MML", quad_points = 7, maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "both", diagnostic_mode = "both")
chk <- reporting_checklist(fit, diagnostics = diag)
summary(chk)
```

summary.mfrm_response_time_review

Summarize a response-time review

Description

Summarize a response-time review

Usage

```
## S3 method for class 'mfrm_response_time_review'
summary(object, top_n = 10L, ...)
```

Arguments

object	An object returned by response_time_review().
top_n	Number of top groups to retain in summary previews.
...	Unused.

Value

A list of class summary.mfrm_response_time_review.

summary.mfrm_signal_detection

Summarize a DIF/bias screening simulation

Description

Summarize a DIF/bias screening simulation

Usage

```
## S3 method for class 'mfrm_signal_detection'
summary(object, digits = 3, ...)
```

Arguments

object	Output from evaluate_mfrm_signal_detection().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class `summary.mfrm_signal_detection` with:

- `overview`: run-level overview
- `detection_summary`: aggregated detection rates by design, with design-variable alias columns when applicable
- `ademp`: simulation-study metadata carried forward from the original object
- `facet_names`: public facet labels carried from the simulation specification
- `design_variable_aliases`: accepted public aliases for design variables
- `design_descriptor`: role-based design-variable metadata
- `planning_scope`: explicit record of the current planning contract
- `planning_constraints`: explicit record of mutable/locked design variables
- `planning_schema`: combined planner-schema contract
- `future_branch_active_summary`: compact deterministic summary of the schema-only future arbitrary-facet planning branch embedded in the current planning schema
- `gpcm_boundary`: bounded-`GPCM` caveat row when present
- `notes`: short interpretation notes, including the bias-side screening caveat

See Also

`evaluate_mfrm_signal_detection()`, plot.mfrm_signal_detection

Examples

```
sig_eval <- suppressWarnings(evaluate_mfrm_signal_detection(
  n_person = 8,
  n_rater = 2,
  n_criterion = 2,
  raters_per_person = 1,
  reps = 1,
  maxit = 30,
  bias_max_iter = 1,
  seed = 123
))
summary(sig_eval)
```

---

summary.mfrm_summary_table_bundle

Summarize a summary-table bundle for manuscript QC

Description

Summarize a summary-table bundle for manuscript QC

Usage

```
## S3 method for class 'mfrm_summary_table_bundle'
summary(object, digits = 3, top_n = 8, ...)
```

Arguments

object	Output from build_summary_table_bundle().
digits	Number of digits used for numeric summaries.
top_n	Maximum number of table-profile rows to keep.
...	Reserved for generic compatibility.

Details

This summary is designed to answer a manuscript-facing question: which reporting tables are available, how large are they, which roles do they serve, and which of them contain numeric content suitable for quick plotting or appendix export.

Value

An object of class summary.mfrm_summary_table_bundle.

Interpreting output

- overview: source class, returned-table count, note count, and whether a numeric table is available for plotting.
- role_summary: counts and total size by reporting role.
- table_catalog: complete returned-table registry with plot/export bridges.
- table_profile: table-level dimensions, numeric-column counts, and missing values for the largest returned tables.
- plot_index: which returned tables are plot-ready and which bundle-level numeric QC routes they support.
- appendix_presets: conservative all / recommended / compact plus section-aware methods / results / diagnostics / reporting appendix-export presets derived from table roles.
- appendix_role_summary: counts of returned tables by reporting role under the same conservative appendix routing used by the bundle catalog.
- appendix_section_summary: counts of returned tables by manuscript-facing appendix section.
- selection_handoff_table_summary: workflow-only table-level appendix handoff crosswalk when present in the bundle.
- selection_handoff_preset_summary: workflow-only appendix handoff overview aggregated at the preset level when present in the bundle.
- selection_handoff_bundle_summary: workflow-only appendix handoff overview aggregated at the bundle-by-section level when present in the bundle.
- selection_handoff_role_summary: workflow-only appendix handoff overview aggregated at the reporting-role level when present in the bundle.

- `selection_handoff_role_section_summary`: workflow-only appendix handoff overview aggregated at the reporting-role by appendix-section level when present in the bundle.
- `selection_summary`, `selection_table_summary`, `selection_table_preset_summary`, `selection_role_summary` `selection_section_summary`, and `selection_catalog`: preset-filtered appendix selection surfaces when workflow-only handoff tables are embedded in the bundle.
- `reporting_map`: where to go next for plotting, APA formatting, and export.
- `notes`: carried forward source-level caveats from the originating summary.

Typical workflow

1. Build `bundle <- build_summary_table_bundle(summary(...))`.
2. Run `summary(bundle)` to see reporting coverage.
3. Use `plot(bundle, type = "table_rows")` or `plot(bundle, type = "numeric_profile", which = ...)` for quick QC.

See Also

`build_summary_table_bundle()`, `apa_table()`, `plot()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
bundle <- build_summary_table_bundle(fit)
summary(bundle)
```

---

summary.mfrm_threshold_profiles

Summarize threshold-profile presets for visual warning logic

Description

Summarize threshold-profile presets for visual warning logic

Usage

```
## S3 method for class 'mfrm_threshold_profiles'
summary(object, digits = 3, ...)
```

Arguments

`object`	Output from `mfrm_threshold_profiles()`.
`digits`	Number of digits used for numeric summaries.
`...`	Reserved for generic compatibility.

Details

Summarizes available warning presets and their PCA reference bands used by `build_visual_summaries()`.

Value

An object of class `summary.mfrm_threshold_profiles`.

Interpreting output

- `thresholds`: raw preset values by profile (`strict`, `standard`, `lenient`).
- `threshold_ranges`: per-threshold span across profiles (sensitivity to profile choice).
- `pca_reference`: literature bands used for PCA narrative labeling.

Larger `Span` in `threshold_ranges` indicates settings that most change warning behavior between strict and lenient modes.

Typical workflow

1. Inspect `summary(mfrm_threshold_profiles())`.
2. Choose profile (`strict` / `standard` / `lenient`) for project policy.
3. Override selected thresholds in `build_visual_summaries()` only when justified.

See Also

`mfrm_threshold_profiles()`, `build_visual_summaries()`

Examples

```
profiles <- mfrm_threshold_profiles()
summary(profiles)
```

---

summary.mfrm_unit_prediction

Summarize posterior unit scoring output

Description

Summarize posterior unit scoring output

Usage

```
## S3 method for class 'mfrm_unit_prediction'
summary(object, digits = 3, ...)
```

Arguments

object	Output from predict_mfrm_units().
digits	Number of digits used in numeric summaries.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_unit_prediction with:

- estimates: posterior summaries by person
- row_review: row-preparation review
- population_review: optional person-level omission review for latent-regression scoring
- settings: scoring settings
- notes: interpretation notes

See Also

predict_mfrm_units()

Examples

```
toy <- load_mfrmr_data("example_core")
keep_people <- unique(toy$Person)[1:18]
toy_fit <- suppressWarnings(
  fit_mfrm(
    toy[toy$Person %in% keep_people, , drop = FALSE],
    "Person", c("Rater", "Criterion"), "Score",
    method = "MML",
    quad_points = 5,
    maxit = 30
  )
)
new_units <- data.frame(
  Person = c("NEW01", "NEW01"),
  Rater = unique(toy$Rater)[1],
  Criterion = unique(toy$Criterion)[1:2],
  Score = c(2, 3)
)
pred_units <- predict_mfrm_units(toy_fit, new_units)
summary(pred_units)
```

summary.mfrm_weighting_review

Summarize a weighting-review object

Description

Summarize a weighting-review object

Usage

```
## S3 method for class 'mfrm_weighting_review'
summary(object, digits = 3, top_n = 10, ...)
```

Arguments

object	Output from build_weighting_review().
digits	Number of digits for printed numeric values.
top_n	Number of top rows to retain in compact summary tables.
...	Reserved for generic compatibility.

Value

An object of class summary.mfrm_weighting_review.

See Also

build_weighting_review()

unexpected_after_bias_table

Build an unexpected-after-adjustment screening report

Description

Build an unexpected-after-adjustment screening report

Usage

```
unexpected_after_bias_table(
  fit,
  bias_results,
  diagnostics = NULL,
  abs_z_min = 2,
  prob_max = 0.3,
  top_n = 100,
  rule = c("either", "both")
)
```

Arguments

fit	Output from fit_mfrm().
bias_results	Output from estimate_bias().
diagnostics	Optional output from diagnose_mfrm() for baseline comparison.
abs_z_min	Absolute standardized-residual cutoff.
prob_max	Maximum observed-category probability cutoff.
top_n	Maximum number of rows to return.
rule	Flagging rule: "either" or "both".

Details

This helper recomputes expected values and residuals after interaction adjustments from estimate_bias() have been introduced.

summary(t10) is supported through summary(). plot(t10) is dispatched through plot() for class mfrm_unexpected_after_bias (type = "scatter", "severity", "comparison").

Value

A named list with:

- table: unexpected responses after bias adjustment
- summary: one-row summary (includes baseline-vs-after counts)
- thresholds: applied thresholds
- facets: analyzed bias facet pair

Interpreting output

- summary: before/after unexpected counts and reduction metrics.
- table: residual unexpected responses after bias adjustment.
- thresholds: screening settings used in this comparison.

Large reductions indicate bias terms explain part of prior unexpectedness; persistent unexpected rows indicate remaining model-data mismatch.

Typical workflow

1. Run unexpected_response_table() as baseline.
2. Estimate bias via estimate_bias().
3. Run unexpected_after_bias_table(...) and compare reductions.

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and vignette("mfrmr-visual-diagnos
package = "mfrmr").

Output columns

The `table` data.frame has the same structure as `unexpected_response_table()` output, with an additional `BiasAdjustment` column showing the bias correction applied to each observation's expected value.

The `summary` data.frame contains:

TotalObservations Total observations analyzed.

BaselineUnexpectedN Unexpected count before bias adjustment.

AfterBiasUnexpectedN Unexpected count after adjustment.

ReducedBy, ReducedPercent Reduction in unexpected count.

See Also

`estimate_bias()`, `unexpected_response_table()`, `bias_count_table()`, mfrmr_visual_diagnostics

Examples

```
toy <- load_mfrmr_data("example_bias")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
bias <- estimate_bias(fit, diag, facet_a = "Rater", facet_b = "Criterion", max_iter = 2)
t10 <- unexpected_after_bias_table(fit, bias, diagnostics = diag, top_n = 20)
summary(t10)
p_t10 <- plot(t10, draw = FALSE)
p_t10$data$plot
```

---

unexpected_response_table

Build an unexpected-response screening report

Description

Build an unexpected-response screening report

Usage

```
unexpected_response_table(
  fit,
  diagnostics = NULL,
  abs_z_min = 2,
  prob_max = 0.3,
  top_n = 100,
  rule = c("either", "both")
)
```

Arguments

fit	Output from fit_mfrm().
diagnostics	Optional output from diagnose_mfrm().
abs_z_min	Absolute standardized-residual cutoff.
prob_max	Maximum observed-category probability cutoff.
top_n	Maximum number of rows to return.
rule	Flagging rule: "either" (default) or "both".

Details

A response is flagged as unexpected when:

- rule = "either": |StdResidual| >= abs_z_min OR ObsProb <= prob_max
- rule = "both": both conditions must be met.

The table includes row-level observed/expected values, residuals, observed-category probability, most-likely category, and a composite severity score for sorting.

Value

A named list with:

- table: flagged response rows
- summary: one-row overview
- thresholds: applied thresholds

Interpreting output

- summary: prevalence of unexpected responses under current thresholds.
- table: ranked row-level diagnostics for case review.
- thresholds: active cutoffs and flagging rule.

Compare results across rule = "either" and rule = "both" to assess how conservative your screening should be.

Typical workflow

1. Start with rule = "either" for broad screening.
2. Re-run with rule = "both" for strict subset.
3. Inspect top rows and visualize with plot_unexpected().

Further guidance

For a plot-selection guide and a longer walkthrough, see mfrmr_visual_diagnostics and vignette("mfrmr-visual-diagnos
package = "mfrmr").

Output columns

The `table` data.frame contains:

Row Original row index in the prepared data.

Person Person identifier (plus one column per facet).

Score Observed score category.

Observed, Expected Observed and model-expected score values.

Residual, StdResidual Raw and standardized residuals.

ObsProb Probability of the observed category under the model.

MostLikely, MostLikelyProb Most probable category and its probability.

Severity Composite severity index (higher = more unexpected).

Direction "Higher than expected" or "Lower than expected".

FlagLowProbability, FlagLargeResidual Logical flags for each criterion.

The `summary` data.frame contains:

TotalObservations Total observations analyzed.

UnexpectedN, UnexpectedPercent Count and share of flagged rows.

AbsZThreshold, ProbThreshold Applied cutoff values.

Rule "either" or "both".

See Also

`diagnose_mfrm()`, `displacement_table()`, `fair_average_table()`, mfrmr_visual_diagnostics

Examples

```
toy_full <- load_mfrmr_data("example_core")
toy_people <- unique(toy_full$Person)[1:12]
toy <- toy_full[toy_full$Person %in% toy_people, , drop = FALSE]
fit <- suppressWarnings(
  fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score", method = "JML", maxit = 30)
)
t4 <- unexpected_response_table(fit, abs_z_min = 1.5, prob_max = 0.4, top_n = 5)
summary(t4)
p_t4 <- plot(t4, draw = FALSE)
p_t4$data$plot
```

visual_reporting_template *Figure-reporting template for visual diagnostics*

Description

Return a compact, beginner-oriented template that explains where each visual family normally belongs in a report, which helper to call, what to say, and what not to claim. Use this static table together with the dynamic `reporting_checklist(fit, diagnostics)$visual_scope` table: the template answers "how should I use this figure?", while the checklist answers "is this figure ready for the current run?".

Usage

```
visual_reporting_template(
  scope = c("all", "manuscript", "appendix", "diagnostic", "surface")
)
```

Arguments

`scope`	Which part of the template to return: `"all"` (default), `"manuscript"`, `"appendix"`, `"diagnostic"`, or `"surface"`.

Details

This helper is intentionally conservative. It does not inspect a fitted object and does not certify that a plot is available. Run `reporting_checklist()` for run-specific readiness, then use this table to decide how to describe the resulting figure.

Value

A data.frame with columns:

- `FigureFamily`: short visual family label.
- `Scope`: broad reporting role used for filtering.
- `PrimaryHelper`: public helper or plot route.
- `DefaultPlacement`: recommended location in a report.
- `WhatToReport`: wording focus for results sections or captions.
- `CaptionSkeleton`: caption starter that must be tailored to the study.
- `ResultsWording`: results-sentence starter that must be checked against the fitted object and diagnostics.
- `WhatNotToClaim`: common overclaim to avoid.
- `BeginnerCheck`: first thing a new user should inspect.
- `ThreeDPolicy`: whether 3D is recommended, discouraged, or data-only.

Examples

```
visual_reporting_template()
visual_reporting_template("manuscript")
visual_reporting_template("surface")
mfrmr_interval_guide("visual")[, c("Route", "PrimaryHelper", "DefaultLevel")]
```

`write_mfrm_residual_file` *Write a standalone residual file*

Description

Write a standalone residual file

Usage

```
write_mfrm_residual_file(
  fit,
  diagnostics = NULL,
  path,
  format = c("csv", "tsv"),
  digits = 4,
  overwrite = FALSE,
  include_probabilities = FALSE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. Supplying it avoids recomputing observation diagnostics.
`path`	Output file path.
`format`	File format: `"csv"` or `"tsv"`. If omitted, inferred from `path` when the extension is `.csv` or `.tsv`, otherwise `"csv"`.
`digits`	Rounding digits for numeric columns.
`overwrite`	If `FALSE`, existing files are not overwritten.
`include_probabilities`	If `TRUE`, append model probabilities for all response categories as `PrCategory_*` columns.

Details

The exported table is observation-level and model-native. It includes the observed score, expected score, residual, standardized residual, variance, score information, observed-category probability, and modeled person measure when those quantities are available.

This writer is separate from `facets_output_file_bundle()` because it is a direct analysis handoff rather than a legacy graph/score bundle.

Value

A bundle with `table`, `summary`, `written_files`, and `settings`.

See Also

`diagnose_mfrm()`, `facets_output_file_bundle()`, `write_mfrm_subset_file()`

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
path <- tempfile(fileext = ".csv")
out <- write_mfrm_residual_file(fit, diag, path, overwrite = TRUE)
out$written_files
```

---

write_mfrm_subset_file

Write standalone subset-connectivity files

Description

Write standalone subset-connectivity files

Usage

```
write_mfrm_subset_file(
  fit,
  diagnostics = NULL,
  path,
  node_path = NULL,
  format = c("csv", "tsv"),
  digits = 4,
  overwrite = FALSE,
  include_nodes = TRUE
)
```

Arguments

`fit`	Output from `fit_mfrm()`.
`diagnostics`	Optional output from `diagnose_mfrm()`. Supplying it avoids recomputing subset connectivity.
`path`	Output file path for the subset summary table.
`node_path`	Optional output file path for the node-level subset table. When `NULL` and `include_nodes = TRUE`, a sibling file ending in `_nodes.csv` or `_nodes.tsv` is created.

format	File format: "csv" or "tsv". If omitted, inferred from path when the extension is .csv or .tsv, otherwise "csv".
digits	Rounding digits for numeric columns.
overwrite	If FALSE, existing files are not overwritten.
include_nodes	If TRUE, also write the node-level facet/level to subset membership table.

Details

Subsets are connected components in the observation design graph. The graph links Person and all modeled facet levels that co-occur in an observation. Multiple subsets mean the scale is not fully connected unless external anchoring or a deliberate separate-calibration design justifies it.

Value

A bundle with table, nodes, summary, written_files, and settings.

See Also

diagnose_mfrm(), subset_connectivity_report(), write_mfrm_residual_file()

Examples

```
toy <- load_mfrmr_data("example_core")
fit <- fit_mfrm(toy, "Person", c("Rater", "Criterion"), "Score",
                method = "JML", maxit = 30)
diag <- diagnose_mfrm(fit, residual_pca = "none")
path <- tempfile(fileext = ".csv")
out <- write_mfrm_subset_file(fit, diag, path, overwrite = TRUE)
out$written_files
```

Index