Package ‘vglmer’

September 12, 2024

Type Package

Title Variational Inference for Hierarchical Generalized Linear Models

Version 1.0.5

Encoding UTF-8

License GPL (>= 2)

Description Estimates hierarchical models using mean-field variational Bayes.
At present, it can estimate logistic, linear, and negative binomial models.
It can accommodate models with an arbitrary number of random effects and
requires no integration to estimate. It also provides the ability to improve
the quality of the approximation using marginal augmentation.
Goplerud (2022) <doi:10.1214/21-BA1266> and Goplerud (2024) <doi:10.1017/S0003055423000035>
provide details on the variational algorithms.

Imports Rcpp (>= 1.0.1), lme4, CholWishart, mvtnorm, Matrix, stats,
graphics, methods, lmtest, splines, mgcv

Depends R (>= 3.0.2)

Suggests SuperLearner, MASS, tictoc, testthat, gKRLS

LinkingTo Rcpp, RcppEigen (>= 0.3.3.4.0)

URL `https://github.com/mgoplerud/vglmer`

BugReports `https://github.com/mgoplerud/vglmer/issues`

RoxygenNote 7.3.2

NeedsCompilation yes

Author Max Goplerud [aut, cre]

Maintainer Max Goplerud <`mgoplerud@austin.utexas.edu`>

Repository CRAN

Date/Publication 2024-09-12 17:40:05 UTC

Contents

`MAVB` *Perform MAVB after fitting vglmer*

Description

Given a model estimated using `vglmer`, this function performs marginally augmented variational Bayes (MAVB) to improve the approximation quality.

Usage

```
MAVB(object, samples, verbose = FALSE, var_px = Inf)
```

Arguments

`object`	Model fit using `vglmer`.
`samples`	Number of samples to draw.
`verbose`	Show progress in drawing samples.
`var_px`	Variance of working prior for marginal augmentation. Default (`Inf`) is a flat, improper, prior.

Details

This function returns the improved estimates of the *parameters*. To use MAVB when generating predictions, one should use predict_MAVB. At present, MAVB is only enabled for binomial models.

Value

This function returns a matrix with `samples` rows and columns for each fixed and random effect.

References

Goplerud, Max. 2022a. "Fast and Accurate Estimation of Non-Nested Binomial Hierarchical Models Using Variational Inference." *Bayesian Analysis*. 17(2): 623-650.

posterior_samples.vglmer — *Draw samples from the variational distribution*

Description

This function draws samples from the estimated variational distributions. If using MAVB to improve the quality of the approximating distribution, please use MAVB or predict_MAVB.

Usage

```
posterior_samples.vglmer(object, samples, verbose = FALSE)
```

Arguments

object	Model fit using vglmer.
samples	Number of samples to draw.
verbose	Show progress in drawing samples.

Value

This function returns a matrix with samples rows and columns for each fixed and random effect.

sl_vglmer — *SuperLearner with (Variational) Hierarchical Models*

Description

These functions integrate vglmer (or glmer) into SuperLearner. Most of the arguments are standard for SuperLearner functions.

Usage

```
SL.vglmer(
  Y,
  X,
  newX,
  formula,
  family,
  id,
  obsWeights,
  control = vglmer_control()
)

## S3 method for class 'SL.vglmer'
```

```
predict(object, newdata, allow_missing_levels = TRUE, ...)

SL.glmer(Y, X, newX, formula, family, id, obsWeights, control = NULL)

## S3 method for class 'SL.glmer'
predict(object, newdata, allow.new.levels = TRUE, ...)

add_formula_SL(learner, env = parent.frame())
```

Arguments

- `Y` From `SuperLearner`: The outcome in the training data set.
- `X` From `SuperLearner`: The predictor variables in the training data.
- `newX` From `SuperLearner`: The predictor variables in validation data.
- `formula` The formula used for estimation.
- `family` From `SuperLearner`: Currently allows `gaussian` or `binomial`.
- `id` From `SuperLearner`: Optional cluster identification variable. See `SuperLearner` for more details.
- `obsWeights` From `SuperLearner`: Weights for each observation. Not permitted for `SL.vglmer`.
- `control` Control object for estimating `vglmer` (e.g., vglmer_control) or `[g]lmer`.
- `object` Used in `predict` for `SL.glmer` and `SL.vglmer`. A model estimated using either `SL.vglmer` or `SL.glmer`.
- `newdata` Dataset to use for predictions.
- `allow_missing_levels` Default (`TRUE`) allows prediction for levels not observed in the estimation data; the value of `0` (with no uncertainty) is used for the corresponding random effect. **Note:** This default differs from `predict.vglmer`.
- `...` Not used; included to maintain compatibility with existing methods.
- `allow.new.levels` From `lme4`: Allow levels in prediction that are not in the training data. Default is `TRUE` for `SuperLearner`.
- `learner` Character name of model from `SuperLearner`. See "Details" for how this is used.
- `env` Environment to assign model. See "Details" for how this is used.

Details

This documentation describes two types of function.

Estimating Hierarchical Models in SuperLearner: Two methods for estimating hierarchical models are provided one for variational methods (`SL.vglmer`) and one for non-variational methods (`SL.glmer`). The accompanying prediction functions are also provided.

Formula with SuperLearner: The `vglmer` package provides a way to estimate models that require or use a formula with `SuperLearner`. This allows for a design to be passed that contains variables that are *not* used in estimation. This can be used as follows (see "Examples").

One calls the function `add_formula_SL` around the quoted name of a `SuperLearner` model, e.g. `add_formula_SL(learner = "SL.knn")`. This creates a new model and predict function with the suffix `"_f"`. This **requires** a formula to be provided for estimation.

With this in hand, `"SL.knn_f"` can be passed to `SuperLearner` with the accompanying formula argument and thus one can compare models with different formula or design on the same ensemble. The `env` argument may need to be manually specified to ensure the created functions can be called by `SuperLearner`.

Value

The functions here return different types of output. `SL.vglmer` and `SL.glmer` return fitted models with the in-sample predictions as standard for `SuperLearner`. The `predict` methods return vectors of predicted values. `add_formula_SL` creates two objects in the environment (one for estimation `model_f` and one for prediction `predict.model_f`) used for `SuperLearner`.

Examples

```
set.seed(456)

if (requireNamespace('SuperLearner', quietly = TRUE)){
require(SuperLearner)
sim_data <- data.frame(
  x = rnorm(100),
  g = sample(letters, 100, replace = TRUE)
)
sim_data$y <- rbinom(nrow(sim_data),
  1, plogis(runif(26)[match(sim_data$g, letters)]))
sim_data$g <- factor(sim_data$g)
sl_vglmer <- function(...){SL.vglmer(..., formula = y ~ x + (1 | g))}
SL.glm <- SuperLearner::SL.glm
add_formula_SL('SL.glm')
sl_glm_form <- function(...){SL.glm_f(..., formula = ~ x)}

   SuperLearner::SuperLearner(
     Y = sim_data$y, family = 'binomial',
     X = sim_data[, c('x', 'g')],
     cvControl = list(V = 2),
     SL.library = c('sl_vglmer', 'sl_glm_form')
   )

}
```

`vglmer` *Variational Inference for Hierarchical Generalized Linear Models*

Description

This function estimates hierarchical models using mean-field variational inference. `vglmer` accepts standard syntax used for `lme4`, e.g., `y ~ x + (x | g)`. Options are described below. Goplerud (2022a; 2022b) provides details on the variational algorithms.

Usage

```
vglmer(formula, data, family, control = vglmer_control())
```

Arguments

formula	lme4 style-formula for random effects. Typically, (1 + z | g) indicates a random effect for each level of variable "g" with a differing slope for the effect of variable "z" and an intercept (1); see "Details" for further discussion and how to incorporate splines.
data	data.frame containing the outcome and predictors.
family	Options are "binomial", "linear", or "negbin" (experimental). If "binomial", outcome must be either binary ($\{0, 1\}$) or cbind(success, failure) as per standard glm(er) syntax. Non-integer values are permitted for binomial if force_whole is set to FALSE in vglmer_control.
control	Adjust internal options for estimation. Must use an object created by vglmer_control.

Details

Estimation Syntax: The formula argument takes syntax designed to be a similar as possible to lme4. That is, one can specify models using y ~ x + (1 | g) where (1 | g) indicates a random intercept. While not tested extensively, terms of (1 | g / f) should work as expected. Terms of (1 + x || g) may work, although will raise a warning about duplicated names of random effects. (1 + x || g) terms may not work with spline estimation. To get around this, one can might copy the column g to g_copy and then write (1 | g) + (0 + x | g_copy).

Splines: Splines can be added using the term v_s(x) for a spline on the variable x. These are transformed into hierarchical terms in a standard fashion (e.g. Ruppert et al. 2003) and then estimated using the variational algorithms. At the present, only truncated linear functions (type = "tpf"; the default) and O'Sullivan splines (Wand and Ormerod 2008) are included. The options are described in more detail at v_s.

It is possible to have the spline vary across some categorical predictor by specifying the "by" argument such as v_s(x, by = g). In effect, this adds additional hierarchical terms for the group-level deviations from the "global" spline. *Note:* In contrast to the typical presentation of these splines interacted with categorical variables (e.g., Ruppert et al. 2003), the default use of "by" includes the lower order interactions that are regularized, i.e. (1 + x | g), versus their unregularized version (e.g., x * g); this can be changed using the by_re argument described in v_s. Further, all group-level deviations from the global spline share the same smoothing parameter (same prior distribution).

Default Settings: By default, the model is estimated using the "strong" (i.e. fully factorized) variational assumption. Setting vglmer_control(factorization_method = "weak") will improve the quality of the variance approximation but may take considerably more time to estimate. See Goplerud (2022a) for discussion.

By default, the prior on each random effect variance (Σ_j) uses a Huang-Wand prior (Huang and Wand 2013) with hyper-parameters $\nu_j = 2$ and $A_{j,k} = 5$. This is designed to be proper but weakly informative. Other options are discussed in vglmer_control under the prior_variance argument.

By default, estimation is accelerated using SQUAREM (Varadhan and Roland 2008) and (one-step-late) parameter expansion for variational Bayes. Under the default "strong" factorization, a "translation" expansion is used; under other factorizations a "mean" expansion is used. These can be adjusted using vglmer_control. See Goplerud (2022b) for more discussion of these methods.

Value

This returns an object of class `vglmer`. The available methods (e.g. `coef`) can be found using `methods(class="vglmer")`.

beta Contains the estimated distribution of the fixed effects (β). It is multivariate normal. `mean` contains the means; `var` contains the variance matrix; `decomp_var` contains a matrix L such that $L^T L$ equals the full variance matrix.

alpha Contains the estimated distribution of the random effects (α). They are all multivariate normal. `mean` contains the means; `dia.var` contains the variance of each random effect. `var` contains the variance matrix of each random effect (j,g). `decomp_var` contains a matrix L such that $L^T L$ equals the full variance of the entire set of random effects.

joint If `factorization_method="weak"`, this is a list with one element (`decomp_var`) that contains a matrix L such that $L^T L$ equals the full variance matrix between the fixed and random effects $q(\beta, \alpha)$. The marginal variances are included in `beta` and `alpha`. If the factorization method is not `"weak"`, this is `NULL`.

sigma Contains the estimated distribution of each random effect covariance Σ_j; all distributions are Inverse-Wishart. `cov` contains a list of the estimated scale matrices. `df` contains a list of the degrees of freedom.

hw If a Huang-Wand prior is used (see Huang and Wand 2013 or Goplerud 2022b for more details), then the estimated distribution. Otherwise, it is `NULL`. All distributions are Inverse-Gamma. `a` contains a list of the scale parameters. `b` contains a list of the shape parameters.

sigmasq If `family="linear"`, this contains a list of the estimated parameters for σ^2; its distribution is Inverse-Gamma. `a` contains the scale parameter; `b` contains the shape parameter.

ln_r If `family="negbin"`, this contains the variational parameters for the log dispersion parameter $\ln(r)$. `mu` contains the mean; `sigma` contains the variance.

family Family of outcome.

ELBO Contains the ELBO at the termination of the algorithm.

ELBO_trajectory `data.frame` tracking the ELBO per iteration.

control Contains the control parameters from `vglmer_control` used in estimation.

internal_parameters Variety of internal parameters used in post-estimation functions.

formula Contains the formula used for estimation; contains the original formula, fixed effects, and random effects parts separately for post-estimation functions. See `formula.vglmer` for more details.

References

Goplerud, Max. 2022a. "Fast and Accurate Estimation of Non-Nested Binomial Hierarchical Models Using Variational Inference." *Bayesian Analysis*. 17(2): 623-650.

Goplerud, Max. 2022b. "Re-Evaluating Machine Learning for MRP Given the Comparable Performance of (Deep) Hierarchical Models." Working paper.

Huang, Alan, and Matthew P. Wand. 2013. "Simple Marginally Noninformative Prior Distributions for Covariance Matrices." *Bayesian Analysis*. 8(2):439-452.

Ruppert, David, Matt P. Wand, and Raymond J. Carroll. 2003. *Semiparametric Regression*. Cambridge University Press.

Varadhan, Ravi, and Christophe Roland. 2008. "Simple and Globally Convergent Methods for Accelerating the Convergence of any EM Algorithm." *Scandinavian Journal of Statistics*. 35(2): 335-353.

Wand, Matt P. and Ormerod, John T. 2008. "On Semiparametric Regression with O'Sullivan Penalized Splines". *Australian & New Zealand Journal of Statistics*. 50(2): 179-198.

Examples

```
set.seed(234)
sim_data <- data.frame(
  x = rnorm(100),
  y = rbinom(100, 1, 0.5),
  g = sample(letters, 100, replace = TRUE)
)

# Run with defaults
est_vglmer <- vglmer(y ~ x + (x | g), data = sim_data, family = "binomial")

# Simple prediction
predict(est_vglmer, newdata = sim_data)

# Summarize results
summary(est_vglmer)

# Extract parameters
coef(est_vglmer); vcov(est_vglmer)

# Comparability with lme4,
# although ranef is formatted differently.
ranef(est_vglmer); fixef(est_vglmer)


# Run with weaker (i.e. better) approximation
vglmer(y ~ x + (x | g),
  data = sim_data,
  control = vglmer_control(factorization_method = "weak"),
  family = "binomial")



# Use a spline on x with a linear outcome
vglmer(y ~ v_s(x),
  data = sim_data,
  family = "linear")
```

vglmer-class *Generic Functions after Running vglmer*

Description

vglmer uses many standard methods from lm and lme4 with limited changes. These provide summaries of the estimated variational distributions.

Usage

```
## S3 method for class 'vglmer'
fixef(object, ...)

## S3 method for class 'vglmer'
sigma(object, ...)

## S3 method for class 'vglmer'
ranef(object, ...)

## S3 method for class 'vglmer'
coef(object, ...)

## S3 method for class 'vglmer'
vcov(object, ...)

## S3 method for class 'vglmer'
fitted(object, ...)

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

## S3 method for class 'vglmer'
summary(object, display_re = TRUE, ...)

## S3 method for class 'vglmer'
formula(x, form = "original", ...)

format_vglmer(object)

format_glmer(object)

ELBO(object, type = c("final", "trajectory"))
```

Arguments

object	Model fit using vglmer.
...	Not used; included to maintain compatibility with existing methods.

x	Model fit using `vglmer`.
display_re	Default (`TRUE`) prints a summary of the random effects alongside the fixed effects.
form	Describes the type of formula to report: `"original"` returns the user input, `"fe"` returns the fixed effects only, `"re"` returns the random effects only.
type	Default (`"final"`) gives the ELBO at convergence. `"trajectory"` gives the ELBO estimated at each iteration. This is used to assess model convergence.

Details

The accompanying functions are briefly described below.

`coef` and `vcov` return the mean and variance of the fixed effects (β). `fixef` returns the mean of the fixed effects.

`ranef` extracts the random effects (α) in a similar, although slightly different format, to `lme4`. It includes the estimated posterior mean and variance in a list of data.frames with one entry per random effect j.

`fitted` extracts the estimated expected *linear predictor*, i.e. $E_{q(\theta)}[x_i^T\beta + z_i^T\alpha]$ at convergence.

`summary` reports the estimates for all fixed effects as in `lm` as well as some summaries of the random effects (if `display_re=TRUE`).

`format_vglmer` collects the mean and variance of the fixed and random effects into a single data.frame. This is useful for examining all of the posterior estimates simultaneously. `format_glmer` converts an object estimated with `[g]lmer` into a comparable format.

`ELBO` extracts the ELBO from the estimated model. `type` can be set equal to `"trajectory"` to get the estimated ELBO at each iteration and assess convergence.

`sigma` extracts the square root of the posterior mode of $q(\sigma^2)$ if a linear model is used.

`formula` extracts the formula associated with the `vglmer` object. By default, it returns the formula provided. The fixed and random effects portions can be extracted separately using the `form` argument.

Value

The functions here return a variety of different objects depending on the specific function. "Details" describes the behavior of each one. Their output is similar to the typical behavior for the corresponding generic functions.

`vglmer_control` *Control for vglmer estimation*

Description

This function controls various estimation options for `vglmer`.

Usage

```
vglmer_control(
  iterations = 1000,
  prior_variance = "hw",
  factorization_method = c("strong", "partial", "weak"),
  parameter_expansion = "translation",
  do_SQUAREM = TRUE,
  tolerance_elbo = 1e-08,
  tolerance_parameters = 1e-05,
  force_whole = TRUE,
  print_prog = NULL,
  do_timing = FALSE,
  verbose_time = FALSE,
  return_data = FALSE,
  linpred_method = "joint",
  vi_r_method = "VEM",
  verify_columns = FALSE,
  debug_param = FALSE,
  debug_ELBO = FALSE,
  debug_px = FALSE,
  quiet = TRUE,
  quiet_rho = TRUE,
  px_method = "dynamic",
  px_numerical_it = 10,
  hw_inner = 10,
  init = "EM_FE"
)
```

Arguments

`iterations` Default of 1000; this sets the maximum number of iterations used in estimation.

`prior_variance` Prior distribution on the random effect variance Σ_j. Options are `hw`, `jeffreys`, `mean_exists`, `uniform`, and `gamma`. The default (`hw`) is the Huang-Wand (2013) prior whose hyper-parameters are $\nu_j = 2$ and $A_{j,k} = 5$. Otherwise, the prior is an Inverse Wishart with the following parameters where d_j is the dimensionality of the random effect j.

- mean_exists: $IW(d_j + 1, I)$
- jeffreys: $IW(0, 0)$
- uniform: $IW(-[d_j + 1], 0)$
- limit: $IW(d_j - 1, 0)$

Estimation may fail if an improper prior (`jeffreys`, `uniform`, `limit`) is used.

`factorization_method`
Factorization assumption for the variational approximation. Default of `"strong"`, i.e. a fully factorized model. Described in detail in Goplerud (2022a). `"strong"`, `"partial"`, and `"weak"` correspond to Schemes I, II, and III respectively in that paper.

parameter_expansion
: Default of "translation" (see Goplerud 2022b). Valid options are "translation", "mean", or "none". "mean" should be employed if "translation" is not enabled or is too computationally expensive. For negative binomial estimation or any estimation where factorization_method != "strong", only "mean" and "none" are available.

do_SQUAREM
: Default (TRUE) accelerates estimation using SQUAREM (Varadhan and Roland 2008).

tolerance_elbo
: Default (1e-8) sets a convergence threshold if the change in the ELBO is below the tolerance.

tolerance_parameters
: Default (1e-5) sets a convergence threshold that is achieved if no parameter changes by more than the tolerance from the prior estimated value.

force_whole
: Default (TRUE) requires integers for observed outcome for binomial or count models. FALSE allows for fractional responses.

print_prog
: Default (NULL) prints a "." to indicate once 5% of the total iterations have elapsed. Set to a positive integer int to print a "." every int iterations.

do_timing
: Default (FALSE) does not estimate timing of each variational update; TRUE requires the package tictoc.

verbose_time
: Default (FALSE) does not print the time elapsed for each parameter update. Set to TRUE, in conjunction with do_timing=TRUE, to see the time taken for each parameter update.

return_data
: Default (FALSE) does not return the original design. Set to TRUE to debug convergence issues.

linpred_method
: Default ("joint") updates the mean parameters for the fixed and random effects simultaneously. This can improve the speed of estimation but may be costly for large datasets; use "cyclical" to update each parameter block separately.

vi_r_method
: Default ("VEM") uses a variational EM algorithm for updating r if family="negbin". This assumes a point mass distribution on r. A number can be provided to fix r. These are the only available options.

verify_columns
: Default (FALSE) **does not** verify that all columns are drawn from the data.frame itself versus the environment. Set to TRUE to debug potential issues.

debug_param
: Default (FALSE) does not store parameters before the final iteration. Set to TRUE to debug convergence issues.

debug_ELBO
: Default (FALSE) does not store the ELBO after each parameter update. Set to TRUE to debug convergence issues.

debug_px
: Default (FALSE) does not store information about whether parameter expansion worked. Set to TRUE to convergence issues.

quiet
: Default (FALSE) does not print intermediate output about convergence. Set to TRUE to debug.

quiet_rho
: Default (FALSE) does not print information about parameter expansions. Set to TRUE to debug convergence issues.

px_method When code `parameter_expansion="translation"`, default (`"dynamic"`) tries a one-step late update and, if this fails, a numerical improvement by L-BFGS-B. For an Inverse-Wishart prior on Σ_j, this is set to `"osl"` that only attempts a one-step-late update.

px_numerical_it Default of 10; if L-BFGS_B is needed for a parameter expansion, this sets the number of steps used.

hw_inner If `prior_variance="hw"`, this sets the number of repeated iterations between estimating Σ_j and $a_{j,k}$ variational distributions at each iteration. A larger number approximates jointly updating both parameters. Default (10) typically performs well.

init Default (`"EM_FE"`) initializes the mean variational parameters for $q(\beta, \alpha)$ by setting the random effects to zero and estimating the fixed effects using a short-running EM algorithm. `"EM"` initializes the model with a ridge regression with a guess as to the random effect variance. `"random"` initializes the means randomly. `"zero"` initializes them at zero.

Value

This function returns a named list with class `vglmer_control`. It is passed to `vglmer` in the argument `control`. This argument only accepts objects created using `vglmer_control`.

References

Goplerud, Max. 2022a. "Fast and Accurate Estimation of Non-Nested Binomial Hierarchical Models Using Variational Inference." *Bayesian Analysis*. 17(2): 623-650.

Goplerud, Max. 2022b. "Re-Evaluating Machine Learning for MRP Given the Comparable Performance of (Deep) Hierarchical Models." Working Paper.

Huang, Alan, and Matthew P. Wand. 2013. "Simple Marginally Noninformative Prior Distributions for Covariance Matrices." *Bayesian Analysis*. 8(2):439-452.

Varadhan, Ravi, and Christophe Roland. 2008. "Simple and Globally Convergent Methods for Accelerating the Convergence of any EM Algorithm." *Scandinavian Journal of Statistics*. 35(2): 335-353.

---

vglmer_predict *Predict after vglmer*

Description

These functions calculate the estimated linear predictor using the variational distributions. `predict.vglmer` draws predictions using the estimated variational distributions; `predict_MAVB` does so using the MAVB procedure described in Goplerud (2022a).

Usage

```
## S3 method for class 'vglmer'
predict(
  object,
  newdata,
  type = "link",
  samples = 0,
  samples_only = FALSE,
  summary = TRUE,
  allow_missing_levels = FALSE,
  ...
)

predict_MAVB(
  object,
  newdata,
  samples = 0,
  samples_only = FALSE,
  var_px = Inf,
  summary = TRUE,
  allow_missing_levels = FALSE
)
```

Arguments

`object` Model fit using `vglmer`.

`newdata` Dataset to use for predictions. It cannot be missing.

`type` Default (`"link"`) returns the linear predictor; `"terms"` returns the predicted value for each random effect (or spline) separately as well as one that collects all fixed effects. At the moment, other options are not enabled.

`samples` Number of samples to draw. Using `0` (default) gives the expectation of the linear predictor. A positive integer draws `samples` samples from the variational distributions and calculates the linear predictor.

`samples_only` Default (`FALSE`) returns the samples from the variational distributions, **not** the prediction. Each row is a sample and each column is a parameter.

`summary` Default (`TRUE`) returns the mean and variance of the samples for each observation. `FALSE` returns a matrix of the sampled linear predictor for each observation. Each row is a sample and each column is an observation.

`allow_missing_levels` Default (`FALSE`) does not allow prediction for levels not observed in the original data. `TRUE` allows for prediction on unseen levels; the value of `0` (with no uncertainty) is used for the corresponding random effect.

`...` Not used; included to maintain compatibility with existing methods.

`var_px` Variance of working prior for marginal augmentation. Default (`Inf`) is a flat, improper, prior.

Value

This function returns an estimate of the linear predictor. The default returns the expected mean, i.e. $E_{q(\alpha,\beta)}[x_i^T \beta + z_i^T \alpha]$. If `samples > 0`, these functions return a summary of the prediction for each observation, i.e. the estimated mean and variance. If `summary = FALSE`, the sampled values of the linear predictor are returned as a matrix. `predict_MAVB` performs MAVB as described in Goplerud (2022a) before returning the linear predictor.

If `allow_missing_levels = TRUE`, then observations with a new (unseen) level for the random effect are given a value of zero for that term of the prediction.

Examples

```
set.seed(123)
sim_data <- data.frame(
  x = rnorm(100),
  y = rbinom(100, 1, 0.5),
  g = sample(letters, 100, replace = TRUE)
)

# Run with defaults
est_vglmer <- vglmer(y ~ x + (x | g), data = sim_data, family = "binomial")

# Simple prediction
predict(est_vglmer, newdata = sim_data)
# Return 10 posterior draws of the linear predictor for each observation.
predict_MAVB(est_vglmer, newdata = sim_data, summary = FALSE, samples = 10)
# Predict with a new level; note this would fail if
# allow_missing_levels = FALSE (the default)
predict(est_vglmer,
  newdata = data.frame(g = "AB", x = 0),
  allow_missing_levels = TRUE
)
```

---

v_s *Create splines for use in vglmer*

Description

This function estimates splines in `vglmer`, similar to `s(...)` in `mgcv` albeit with many fewer options than `mgcv`. It allows for truncated (linear) splines (`type="tpf"`), O'Sullivan splines (`type="o"`), or kernel ridge regression (`type="gKRLS"`). Please see vglmer for more discussion and examples. For information on kernel ridge regression, please consult gKRLS.

Usage

```
v_s(
  ...,
  type = "tpf",
  knots = NULL,
```

```
  by = NA,
  xt = NULL,
  by_re = TRUE,
  force_vector = FALSE,
  outer_okay = FALSE
)
```

Arguments

`...`	Variable name, e.g. `v_s(x)`
`type`	Default (`"tpf"`) uses truncated linear splines for the basis. `"o"` uses O'Sullivan splines (Wand and Ormerod 2008). Smoothing across multiple covariates, e.g. `v_s(x,x2,type="gKRLS")`, can be done using kernel ridge regression. Chang and Goplerud (2024) provide a detailed discussion. Note that `"gKRLS"` by default uses random sketching to create the relevant bases and thus a seed would need to be set to ensure exact replicability.
`knots`	Default (`NULL`) uses $K = min(N/4, 35)$ knots evenly spaced at quantiles of the covariate `x`. A single number specifies a specific number of knots; a vector can set custom locations for knots.
`by`	A categorical or factor covariate to interact the spline with; for example, `v_s(x, by = g)`.
`xt`	Arguments passed to `xt` from `mgcv`; at the moment, this is only used for `type="gKRLS"` to pass the function `gKRLS()`. Please see the documentation of `gKRLS` for more details.
`by_re`	Default (`TRUE`) regularizes the interactions between the categorical factor and the covariate. See "Details" in vglmer for more discussion.
`force_vector`	Force that argument to `knots` is treated as vector. This is usually not needed unless `knots` is a single integer that should be treated as a single knot (vs. the number of knots).
`outer_okay`	Default (`FALSE`) does not permit values in `x` to exceed the outer knots.

Value

This function returns a list of class of `vglmer_spline` that is passed to unexported functions. It contains the arguments noted above where `...` is parsed into an argument called `term`.

References

Chang, Qing, and Max Goplerud. 2024. "Generalized Kernel Regularized Least Squares." *Political Analysis* 32(2):157-171.

Wand, Matt P. and Ormerod, John T. 2008. "On Semiparametric Regression with O'Sullivan Penalized Splines". *Australian & New Zealand Journal of Statistics*. 50(2): 179-198.

Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC.

Index