Package 'EpiNow2'

May 17, 2024

Type Package

Title Estimate Real-Time Case Counts and Time-Varying Epidemiological Parameters

Version 1.5.2

Description Estimates the time-varying reproduction number, rate of spread, and doubling time using a range of open-source tools (Abbott et al. (2020) <doi:10.12688/wellcomeopenres.16006.1>), and current best practices (Gostic et al. (2020) <doi:10.1101/2020.06.18.20134858>). It aims to help users avoid some of the limitations of naive implementations in a framework that is informed by community feedback and is actively supported.

License MIT + file LICENSE

URL https://epiforecasts.io/EpiNow2/, https://epiforecasts.io/EpiNow2/dev/, https://github.com/epiforecasts/EpiNow2

BugReports https://github.com/epiforecasts/EpiNow2/issues

Depends R (>= 3.5.0)

Imports checkmate, data.table, futile.logger (>= 1.4), future, future.apply, ggplot2, lifecycle, lubridate, methods, patchwork, posterior, progressr, purrr, R.utils (>= 2.0.0), Rcpp (>= 0.12.0), rlang (>= 0.4.7), rstan (>= 2.26.0), rstantools (>= 2.2.0), runner, scales, stats, truncnorm, utils

Suggests cmdstanr, covr, here, knitr, precommit, rmarkdown, spelling, testthat, usethis, withr

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

Additional_repositories https://mc-stan.org/r-packages/

Biarch true

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.1

NeedsCompilation yes

SystemRequirements GNU make C++17

VignetteBuilder knitr

Author Sam Abbott [aut] (<https://orcid.org/0000-0001-8057-8037>),
Joel Hellewell [aut] (<https://orcid.org/0000-0003-2683-0849>),
Katharine Sherratt [aut],
Katelyn Gostic [aut],
Joe Hickson [aut],
Hamada S. Badr [aut] (<https://orcid.org/0000-0002-9808-2344>),
Michael DeWitt [aut] (<https://orcid.org/0000-0001-8940-1967>),
James M. Azam [aut] (<https://orcid.org/0000-0001-5782-7330>),
Robin Thompson [ctb],
Sophie Meakin [ctb],
James Munday [ctb],
Nikos Bosse [ctb],
Paul Mee [ctb],
Peter Ellis [ctb],
Pietro Monticone [ctb],
Lloyd Chapman [ctb],
Andrew Johnson [ctb],
EpiForecasts [aut],
Sebastian Funk [aut, cre] (<https://orcid.org/0000-0002-2842-3406>)

Maintainer Sebastian Funk <sebastian.funk@lshtm.ac.uk>

Repository CRAN

Date/Publication 2024-05-16 22:50:06 UTC

R topics documented:

`+.dist_spec` *Creates a delay distribution as the sum of two other delay distributions.*

Description

[Experimental]

Usage

```
## S3 method for class 'dist_spec'
e1 + e2
```

Arguments

`e1`	The first delay distribution (of type `dist_spec()`) to combine.
`e2`	The second delay distribution (of type `dist_spec()`) to combine.

Value

A delay distribution representing the sum of the two delays

Examples

```
# A fixed lognormal distribution with mean 5 and sd 1.
dist1 <- LogNormal(
  meanlog = 1.6, sdlog = 1, max = 20
)
dist1 + dist1

# An uncertain gamma distribution with mean 3 and sd 2
dist2 <- Gamma(
  mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20
)
dist1 + dist2
```

---

apply_tolerance *Applies a threshold to all nonparametric distributions in a <dist_spec>*

Description

[Experimental] This removes any part of the tail of the nonparametric distributions in the <dist_spec> where the probability mass is below the threshold level.

Usage

```
apply_tolerance(x, tolerance)
```

Arguments

x	A <dist_spec>
tolerance	Numeric; the desired tolerance level.

Value

A <dist_spec> where probability masses below the threshold level have been removed

Examples

```
dist <- discretise(Gamma(mean = 5, sd = 1, max = 20))
apply_tolerance(dist, 0.01)
```

backcalc_opts *Back Calculation Options*

Description

[Stable] Defines a list specifying the optional arguments for the back calculation of cases. Only used if `rt = NULL`.

Usage

```
backcalc_opts(
  prior = c("reports", "none", "infections"),
  prior_window = 14,
  rt_window = 1
)
```

Arguments

`prior`	A character string defaulting to "reports". Defines the prior to use when deconvolving. Currently implemented options are to use smoothed mean delay shifted reported cases ("reports"), to use the estimated infections from the previous time step seeded for the first time step using mean shifted reported cases ("infections"), or no prior ("none"). Using no prior will result in poor real time performance. No prior and using infections are only supported when a Gaussian process is present . If observed data is not reliable then it a sensible first step is to explore increasing the `prior_window` wit a sensible second step being to no longer use reported cases as a prior (i.e set `prior = "none"`).
`prior_window`	Integer, defaults to 14 days. The mean centred smoothing window to apply to mean shifted reports (used as a prior during back calculation). 7 days is minimum recommended settings as this smooths day of the week effects but depending on the quality of the data and the amount of information users wish to use as a prior (higher values equalling a less informative prior).
`rt_window`	Integer, defaults to 1. The size of the centred rolling average to use when estimating Rt. This must be odd so that the central estimate is included.

Value

A <backcalc_opts> object of back calculation settings.

Examples

```
# default settings
backcalc_opts()
```

`bootstrapped_dist_fit` *Fit a Subsampled Bootstrap to Integer Values and Summarise Distribution Parameters*

Description

[Stable] Fits an integer adjusted distribution to a subsampled bootstrap of data and then integrates the posterior samples into a single set of summary statistics. Can be used to generate a robust reporting delay that accounts for the fact the underlying delay likely varies over time or that the size of the available reporting delay sample may not be representative of the current case load.

Usage

```
bootstrapped_dist_fit(
  values,
  dist = "lognormal",
  samples = 2000,
  bootstraps = 10,
  bootstrap_samples = 250,
  max_value,
  verbose = FALSE
)
```

Arguments

`values`	Integer vector of values.
`dist`	Character string, which distribution to fit. Defaults to lognormal (`"lognormal"`) but gamma (`"gamma"`) is also supported.
`samples`	Numeric, number of samples to take overall from the bootstrapped posteriors.
`bootstraps`	Numeric, defaults to 1. The number of bootstrap samples (with replacement) of the delay distribution to take.
`bootstrap_samples`	Numeric, defaults to 100. The number of samples to take in each bootstrap. When the sample size of the supplied delay distribution is less than 100 this is used instead.
`max_value`	Numeric, defaults to the maximum value in the observed data. Maximum delay to allow (added to output but does impact fitting).
`verbose`	Logical, defaults to `FALSE`. Should progress messages be printed.

Value

A `<dist_spec>` object summarising the bootstrapped distribution

Examples

```
# lognormal
delays <- rlnorm(500, log(5), 1)
out <- bootstrapped_dist_fit(delays,
  samples = 1000, bootstraps = 10,
  dist = "lognormal"
)
out
```

`c.dist_spec` *Combines multiple delay distributions for further processing*

Description

[Experimental] This combines the parameters so that they can be fed as multiple delay distributions to `epinow()` or `estimate_infections()`.

Usage

```
## S3 method for class 'dist_spec'
c(...)
```

Arguments

`...`	The delay distributions (from calls to `dist_spec()`) to combine

Value

Combined delay distributions (with class `<dist_spec>`)

Examples

```
# A fixed lognormal distribution with mean 5 and sd 1.
dist1 <- LogNormal(
  meanlog = 1.6, sdlog = 1, max = 20
)
dist1 + dist1

# An uncertain gamma distribution with mean 3 and sd 2
dist2 <- Gamma(
  mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20
)
c(dist1, dist2)
```

calc_CrI *Calculate Credible Interval*

Description

[Stable] Adds symmetric a credible interval based on quantiles.

Usage

```
calc_CrI(samples, summarise_by = NULL, CrI = 0.9)
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.
CrI	Numeric between 0 and 1. The credible interval for which to return values. Defaults to 0.9.

Value

A data.table containing the upper and lower bounds for the specified credible interval.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# add 90% credible interval
calc_CrI(samples)
# add 90% credible interval grouped by type
calc_CrI(samples, summarise_by = "type")
```

calc_CrIs *Calculate Credible Intervals*

Description

[Stable] Adds symmetric credible intervals based on quantiles.

Usage

```
calc_CrIs(samples, summarise_by = NULL, CrIs = c(0.2, 0.5, 0.9))
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.
CrIs	Numeric vector of credible intervals to calculate.

Value

A data.table containing the summarise_by variables and the specified lower and upper credible intervals.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# add credible intervals
calc_CrIs(samples)
# add 90% credible interval grouped by type
calc_CrIs(samples, summarise_by = "type")
```

---

calc_summary_measures *Calculate All Summary Measures*

Description

[Stable] Calculate summary statistics and credible intervals from a <data.frame> by group.

Usage

```
calc_summary_measures(
  samples,
  summarise_by = NULL,
  order_by = NULL,
  CrIs = c(0.2, 0.5, 0.9)
)
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.
order_by	A character vector of parameters to order by, defaults to all summarise_by variables.
CrIs	Numeric vector of credible intervals to calculate.

Value

A data.table containing summary statistics by group.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# default
calc_summary_measures(samples)
#  by type
calc_summary_measures(samples, summarise_by = "type")
```

calc_summary_stats *Calculate Summary Statistics*

Description

[Stable] Calculate summary statistics from a <data.frame> by group. Currently supports the mean, median and standard deviation.

Usage

```
calc_summary_stats(samples, summarise_by = NULL)
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.

Value

A data.table containing the upper and lower bounds for the specified credible interval

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# default
calc_summary_stats(samples)
#  by type
calc_summary_stats(samples, summarise_by = "type")
```

clean_nowcasts *Clean Nowcasts for a Supplied Date*

Description

[Stable] This function removes nowcasts in the format produced by EpiNow2 from a target directory for the date supplied.

Usage

```
clean_nowcasts(date = NULL, nowcast_dir = ".")
```

Arguments

date	Date object. Defaults to today's date
nowcast_dir	Character string giving the filepath to the nowcast results directory. Defaults to the current directory.

Value

No return value, called for side effects

`clean_regions` *Clean Regions*

Description

[Stable] Removes regions with insufficient time points, and provides logging information on the input.

Usage

```
clean_regions(data, non_zero_points)
```

Arguments

`data`	A <`data.frame`> of confirmed cases (confirm) by date (date), and region (`region`).
`non_zero_points`	Numeric, the minimum number of time points with non-zero cases in a region required for that region to be evaluated. Defaults to 7.

Value

A dataframe of cleaned regional data

See Also

`regional_epinow()`

`collapse` *Collapse nonparametric distributions in a <dist_spec>*

Description

[Experimental] This convolves any consecutive nonparametric distributions contained in the <dist_spec>.

Usage

```
collapse(x)
```

Arguments

`x`	A <`dist_spec`>

Value

A <dist_spec> where consecutive nonparametric distributions have been convolved

Examples

```
# A fixed gamma distribution with mean 5 and sd 1.
dist1 <- Gamma(mean = 5, sd = 1, max = 20)

# An uncertain lognormal distribution with mean 3 and sd 2
dist2 <- LogNormal(mean = 3, sd = 2, max = 20)

# The maxf the sum of two distributions
collapse(discretise(dist1 + dist2))
```

---

convert_to_logmean *Convert mean and sd to log mean for a log normal distribution*

Description

[Stable] Convert from mean and standard deviation to the log mean of the lognormal distribution. Useful for defining distributions supported by `estimate_infections()`, `epinow()`, and `regional_epinow()`.

Usage

```
convert_to_logmean(mean, sd)
```

Arguments

mean	Numeric, mean of a distribution
sd	Numeric, standard deviation of a distribution

Value

The log mean of a lognormal distribution

Examples

```
convert_to_logmean(2, 1)
```

convert_to_logsd *Convert mean and sd to log standard deviation for a log normal distribution*

Description

[Stable] Convert from mean and standard deviation to the log standard deviation of the lognormal distribution. Useful for defining distributions supported by `estimate_infections()`, `epinow()`, and `regional_epinow()`.

Usage

```
convert_to_logsd(mean, sd)
```

Arguments

mean	Numeric, mean of a distribution
sd	Numeric, standard deviation of a distribution

Value

The log standard deviation of a lognormal distribution

Examples

```
convert_to_logsd(2, 1)
```

convolve_and_scale *Convolve and scale a time series*

Description

This applies a lognormal convolution with given, potentially time-varying parameters representing the parameters of the lognormal distribution used for the convolution and an optional scaling factor. This is akin to the model used in `estimate_secondary()` and `simulate_secondary()`.

Usage

```
convolve_and_scale(
  data,
  type = c("incidence", "prevalence"),
  family = c("none", "poisson", "negbin"),
  delay_max = 30,
  ...
)
```

Arguments

data	A <data.frame> containing the date of report and primary cases as a numeric vector.
type	A character string indicating the type of observation the secondary reports are. Options include: • "incidence": Assumes that secondary reports equal a convolution of previously observed primary reported cases. An example application is deaths from an infectious disease predicted by reported cases of that disease (or estimated infections). • "prevalence": Assumes that secondary reports are cumulative and are defined by currently observed primary reports minus a convolution of secondary reports. An example application is hospital bed usage predicted by hospital admissions.
family	Character string defining the observation model. Options are Negative binomial ("negbin"), the default, Poisson ("poisson"), and "none" meaning the expectation is returned.
delay_max	Integer, defaulting to 30 days. The maximum delay used in the convolution model.
...	Additional parameters to pass to the observation model (i.e rnbinom or rpois).

Details

Up to version 1.4.0 this function was called simulate_secondary().

Value

A <data.frame> containing simulated data in the format required by estimate_secondary().

See Also

estimate_secondary

Examples

```
# load data.table for manipulation
library(data.table)

#### Incidence data example ####

# make some example secondary incidence data
cases <- example_confirmed
cases <- as.data.table(cases)[, primary := confirm]

# Assume that only 40 percent of cases are reported
cases[, scaling := 0.4]

# Parameters of the assumed log normal delay distribution
cases[, meanlog := 1.8][, sdlog := 0.5]
```

```
# Simulate secondary cases
cases <- convolve_and_scale(cases, type = "incidence")
cases
#### Prevalence data example ####

# make some example prevalence data
cases <- example_confirmed
cases <- as.data.table(cases)[, primary := confirm]

# Assume that only 30 percent of cases are reported
cases[, scaling := 0.3]

# Parameters of the assumed log normal delay distribution
cases[, meanlog := 1.6][, sdlog := 0.8]

# Simulate secondary cases
cases <- convolve_and_scale(cases, type = "prevalence")
cases
```

`delay_opts` *Delay Distribution Options*

Description

[Stable] Returns delay distributions formatted for usage by downstream functions.

Usage

```
delay_opts(
  dist = Fixed(0),
  ...,
  fixed = FALSE,
  tolerance = 0.001,
  weight_prior = TRUE
)
```

Arguments

`dist`	A delay distribution or series of delay distributions. Default is a fixed distribution with all mass at 0, i.e. no delay.
`...`	deprecated; use `dist` instead
`fixed`	deprecated; use `dist` instead
`tolerance`	Numeric; the desired tolerance level.
`weight_prior`	Logical; if TRUE (default), any priors given in `dist` will be weighted by the number of observation data points, in doing so approximately placing an independent prior at each time step and usually preventing the posteriors from shifting. If FALSE, no weight will be applied, i.e. any parameters in `dist` will be treated as a single parameters.

Value

A <delay_opts> object summarising the input delay distributions.

See Also

convert_to_logmean() convert_to_logsd() bootstrapped_dist_fit() dist_spec()

Examples

```
# no delays
delay_opts()

# A single delay that has uncertainty
delay <- LogNormal(mean = Normal(1, 0.2), sd = Normal(0.5, 0.1), max = 14)
delay_opts(delay)

# A single delay without uncertainty
delay <- LogNormal(meanlog = 1, sdlog = 0.5, max = 14)
delay_opts(delay)

# Multiple delays (in this case twice the same)
delay_opts(delay + delay)
```

---

discretise *Discretise a <dist_spec>*

Description

[Experimental] By default it will discretise all the distributions it can discretise (i.e. those with finite support and constant parameters).

Usage

```
discretise(x, strict = TRUE)

discretize(x, strict = TRUE)
```

Arguments

x	A <dist_spec>
strict	Logical; If TRUE (default) an error will be thrown if a distribution cannot be discretised (e.g., because no finite maximum has been specified or parameters are uncertain). If FALSE then any distribution that cannot be discretised will be returned as is.

Details

Discretise a <dist_spec>

Value

A <dist_spec> where all distributions with constant parameters are nonparametric.

Examples

```
# A fixed gamma distribution with mean 5 and sd 1.
dist1 <- Gamma(mean = 5, sd = 1, max = 20)

# An uncertain lognormal distribution with mean 3 and sd 2
dist2 <- LogNormal(mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20)

# The maxf the sum of two distributions
discretise(dist1 + dist2, strict = FALSE)
```

Distributions *Probability distributions*

Description

Probability distributions

Generates a nonparametric distribution.

Usage

```
LogNormal(meanlog, sdlog, mean, sd, max = Inf)

Gamma(shape, rate, scale, mean, sd, max = Inf)

Normal(mean, sd, max = Inf)

Fixed(value, max = Inf)

NonParametric(pmf)
```

Arguments

meanlog, sdlog	mean and standard deviation of the distribution on the log scale with default values of 0 and 1 respectively.
mean, sd	mean and standard deviation of the distribution
max	Numeric, maximum value of the distribution. The distribution will be truncated at this value. Default: Inf, i.e. no maximum.
shape, scale	shape and scale parameters. Must be positive, scale strictly.
rate	an alternative way to specify the scale.
value	Value of the fixed (delta) distribution
pmf	Probability mass of the given distribution; this is passed as a zero-indexed numeric vector (i.e. the fist entry represents the probability mass of zero). If not summing to one it will be normalised to sum to one internally.

Details

Probability distributions are ubiquitous in EpiNow2, usually representing epidemiological delays (e.g., the generation time for delays between becoming infecting and infecting others; or reporting delays)

They are generated using functions that have a name corresponding to the probability distribution that is being used. They generated `dist_spec` objects that are then passed to the models underlying EpiNow2. All parameters can be given either as fixed values (a numeric value) or as uncertain values (a `dist_sepc`). If given as uncertain values, currently only normally distributed parameters (generated using `Normal()`) are supported.

Each distribution has a representation in terms of "natural" parameters (the ones used in stan) but can sometimes also be specified using other parameters such as the mean or standard deviation of the distribution. If not given as natural parameters then these will be calculated from the given parameters. If they have uncertainty, this will be done by random sampling from the given uncertainty and converting resulting parameters to their natural representation.

Currently available distributions are lognormal, gamma, normal, fixed (delta) and nonparametric. The nonparametric is a special case where the probability mass function is given directly as a numeric vector.

Value

A `dist_spec` representing a distribution of the given specification.

Examples

```
LogNormal(mean = 4, sd = 1)
LogNormal(mean = 4, sd = 1, max = 10)
LogNormal(mean = Normal(4, 1), sd = 1, max = 10)
Gamma(mean = 4, sd = 1)
Gamma(shape = 16, rate = 4)
Gamma(shape = Normal(16, 2), rate = Normal(4, 1))
Gamma(shape = Normal(16, 2), scale = Normal(1/4, 1))
Normal(mean = 4, sd = 1)
Normal(mean = 4, sd = 1, max = 10)
Fixed(value = 3)
Fixed(value = 3.5)
NonParametric(c(0.1, 0.3, 0.2, 0.4))
NonParametric(c(0.1, 0.3, 0.2, 0.1, 0.1))
```

---

dist_fit *Fit an Integer Adjusted Exponential, Gamma or Lognormal distributions*

Description

[Stable] Fits an integer adjusted exponential, gamma or lognormal distribution using stan.

Usage

```
dist_fit(
  values = NULL,
  samples = 1000,
  cores = 1,
  chains = 2,
  dist = "exp",
  verbose = FALSE,
  backend = "rstan"
)
```

Arguments

values	Numeric vector of values
samples	Numeric, number of samples to take. Must be >= 1000. Defaults to 1000.
cores	Numeric, defaults to 1. Number of CPU cores to use (no effect if greater than the number of chains).
chains	Numeric, defaults to 2. Number of MCMC chains to use. More is better with the minimum being two.
dist	Character string, which distribution to fit. Defaults to exponential ("exp") but gamma ("gamma") and lognormal ("lognormal") are also supported.
verbose	Logical, defaults to FALSE. Should verbose progress messages be printed.
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".

Value

A stan fit of an interval censored distribution

Examples

```
# integer adjusted exponential model
dist_fit(rexp(1:100, 2),
  samples = 1000, dist = "exp",
  cores = ifelse(interactive(), 4, 1), verbose = TRUE
)


# integer adjusted gamma model
dist_fit(rgamma(1:100, 5, 5),
  samples = 1000, dist = "gamma",
  cores = ifelse(interactive(), 4, 1), verbose = TRUE
)

# integer adjusted lognormal model
dist_fit(rlnorm(1:100, log(5), 0.2),
  samples = 1000, dist = "lognormal",
```

```
  cores = ifelse(interactive(), 4, 1), verbose = TRUE
)
```

`dist_skel` *Distribution Skeleton*

Description

[Questioning] This function acts as a skeleton for a truncated distribution defined by model type, maximum value and model parameters. It is designed to be used with the output from `get_dist()`.

Usage

```
dist_skel(
  n,
  dist = FALSE,
  cum = TRUE,
  model,
  discrete = FALSE,
  params,
  max_value = 120
)
```

Arguments

`n`	Numeric vector, number of samples to take (or days for the probability density).
`dist`	Logical, defaults to `FALSE`. Should the probability density be returned rather than a number of samples.
`cum`	Logical, defaults to `TRUE`. If `dist = TRUE` should the returned distribution be cumulative.
`model`	Character string, defining the model to be used. Supported options are exponential ("exp"), gamma ("gamma"), and log normal ("lognormal")
`discrete`	Logical, defaults to `FALSE`. Should the probability distribution be discretised. In this case each entry of the probability mass function corresponds to the 2-length interval ending at the entry except for the first interval that covers (0, 1). That is, the probability mass function is a vector where the first entry corresponds to the integral over the (0,1] interval of the continuous distribution, the second entry corresponds to the (0,2] interval, the third entry corresponds to the (1, 3] interval etc.
`params`	A list of parameters values (by name) required for each model. For the exponential model this is a rate parameter and for the gamma model this is alpha and beta.
`max_value`	Numeric, the maximum value to allow. Defaults to 120. Samples outside of this range are resampled.

Value

A vector of samples or a probability distribution.

Examples

```
## Exponential model
# sample
dist_skel(10, model = "exp", params = list(rate = 1))

# cumulative prob density
dist_skel(1:10, model = "exp", dist = TRUE, params = list(rate = 1))

# probability density
dist_skel(1:10,
  model = "exp", dist = TRUE,
  cum = FALSE, params = list(rate = 1)
)

## Gamma model
# sample
dist_skel(10, model = "gamma", params = list(shape = 1, rate = 0.5))

# cumulative prob density
dist_skel(0:10,
  model = "gamma", dist = TRUE,
  params = list(shape = 1, rate = 0.5)
)

# probability density
dist_skel(0:10,
  model = "gamma", dist = TRUE,
  cum = FALSE, params = list(shape = 2, rate = 0.5)
)

## Log normal model
# sample
dist_skel(10,
  model = "lognormal", params = list(meanlog = log(5), sdlog = log(2))
)

# cumulative prob density
dist_skel(0:10,
  model = "lognormal", dist = TRUE,
  params = list(meanlog = log(5), sdlog = log(2))
)

# probability density
dist_skel(0:10,
  model = "lognormal", dist = TRUE, cum = FALSE,
  params = list(meanlog = log(5), sdlog = log(2))
)
```

epinow *Real-time Rt Estimation, Forecasting and Reporting*

Description

[Stable] This function wraps the functionality of `estimate_infections()` in order to estimate Rt and cases by date of infection and forecast these infections into the future. In addition to the functionality of `estimate_infections()` it produces additional summary output useful for reporting results and interpreting them as well as error catching and reporting, making it particularly useful for production use e.g. running at set intervals on a dedicated server.

Usage

```
epinow(
  data,
  generation_time = generation_time_opts(),
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  filter_leading_zeros = TRUE,
  zero_threshold = Inf,
  return_output = FALSE,
  output = c("samples", "plots", "latest", "fit", "timing"),
  plot_args = list(),
  target_folder = NULL,
  target_date,
  logs = tempdir(),
  id = "epinow",
  verbose = interactive(),
  reported_cases
)
```

Arguments

`data`	A <data.frame> of confirmed cases (confirm) by date (date). `confirm` must be numeric and `date` must be in date format.
`generation_time`	A call to `generation_time_opts()` defining the generation time distribution used. For backwards compatibility a list of summary parameters can also be passed.

delays	A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.
truncation	A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.
rt	A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().
gp	A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts(). Set to NULL to disable the Gaussian process.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().
stan	A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
filter_leading_zeros	Logical, defaults to TRUE. Should zeros at the start of the time series be filtered out.
zero_threshold	**[Experimental]** Numeric defaults to Inf. Indicates if detected zero cases are meaningful by using a threshold number of cases based on the 7-day average. If the average is above this threshold then the zero is replaced using fill.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
output	A character vector of optional output to return. Supported options are samples ("samples"), plots ("plots"), the run time ("timing"), copying the dated folder into a latest folder (if target_folder is not null, set using "latest"), and the stan fit ("fit"). The default is to return all options.
plot_args	A list of optional arguments passed to plot.epinow().
target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.
logs	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if logs is set to NULL. If specifying a custom logging setup then the code for setup_default_logging() and the setup_logging() function are a sensible place to start.
id	A character string used to assign logging information on error. Used by regional_epinow() to assign errors to regions. Alter the default to run with error catching.

verbose	Logical, defaults to TRUE when used interactively and otherwise FALSE. Should verbose debug progress messages be printed. Corresponds to the "DEBUG" level from futile.logger. See setup_logging for more detailed logging options.
reported_cases	Deprecated; use data instead.

Value

A list of output from estimate_infections with additional elements summarising results and reporting errors if they have occurred.

See Also

estimate_infections() forecast_infections() regional_epinow()

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))

# set an example generation time. In practice this should use an estimate
# from the literature or be estimated from data
generation_time <- Gamma(
  shape = Normal(1.3, 0.3),
  rate = Normal(0.37, 0.09),
  max = 14
)
# set an example incubation period. In practice this should use an estimate
# from the literature or be estimated from data
incubation_period <- LogNormal(
   meanlog = Normal(1.6, 0.06),
   sdlog = Normal(0.4, 0.07),
   max = 14
)
# set an example reporting delay. In practice this should use an estimate
# from the literature or be estimated from data
reporting_delay <- LogNormal(mean = 2, sd = 1, max = 10)

# example case data
reported_cases <- example_confirmed[1:40]

# estimate Rt and nowcast/forecast cases by date of infection
out <- epinow(
  reported_cases = reported_cases,
  generation_time = generation_time_opts(generation_time),
  rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
  delays = delay_opts(incubation_period + reporting_delay)
)
# summary of the latest estimates
summary(out)
```

```
# plot estimates
plot(out)

# summary of R estimates
summary(out, type = "parameters", params = "R")

options(old_opts)
```

`estimate_delay` *Estimate a Delay Distribution*

Description

[Maturing] Estimate a log normal delay distribution from a vector of integer delays. Currently this function is a simple wrapper for `bootstrapped_dist_fit()`.

Usage

```
estimate_delay(delays, ...)
```

Arguments

`delays`	Integer vector of delays
`...`	Arguments to pass to internal methods.

Value

A `<dist_spec>` summarising the bootstrapped distribution

See Also

`bootstrapped_dist_fit()`

Examples

```
delays <- rlnorm(500, log(5), 1)
estimate_delay(delays, samples = 1000, bootstraps = 10)
```

`estimate_infections` *Estimate Infections, the Time-Varying Reproduction Number and the Rate of Growth*

Description

[Maturing] Uses a non-parametric approach to reconstruct cases by date of infection from reported cases. It uses either a generative Rt model or non-parametric back calculation to estimate underlying latent infections and then maps these infections to observed cases via uncertain reporting delays and a flexible observation model. See the examples and function arguments for the details of all options. The default settings may not be sufficient for your use case so the number of warmup samples (`stan_args = list(warmup)`) may need to be increased as may the overall number of samples. Follow the links provided by any warnings messages to diagnose issues with the MCMC fit. It is recommended to explore several of the Rt estimation approaches supported as not all of them may be suited to users own use cases. See here for an example of using `estimate_infections` within the `epinow` wrapper to estimate Rt for Covid-19 in a country from the ECDC data source.

Usage

```
estimate_infections(
  data,
  generation_time = generation_time_opts(),
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  filter_leading_zeros = TRUE,
  zero_threshold = Inf,
  weigh_delay_priors = TRUE,
  id = "estimate_infections",
  verbose = interactive(),
  reported_cases
)
```

Arguments

`data`	A `<data.frame>` of confirmed cases (confirm) by date (date). `confirm` must be numeric and `date` must be in date format.
`generation_time`	A call to `generation_time_opts()` defining the generation time distribution used. For backwards compatibility a list of summary parameters can also be passed.

delays — A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.

truncation — A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.

rt — A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.

backcalc — A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().

gp — A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts(). Set to NULL to disable the Gaussian process.

obs — A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().

stan — A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.

horizon — Numeric, defaults to 7. Number of days into the future to forecast.

CrIs — Numeric vector of credible intervals to calculate.

filter_leading_zeros — Logical, defaults to TRUE. Should zeros at the start of the time series be filtered out.

zero_threshold — **[Experimental]** Numeric defaults to Inf. Indicates if detected zero cases are meaningful by using a threshold number of cases based on the 7-day average. If the average is above this threshold then the zero is replaced using fill.

weigh_delay_priors — Logical. If TRUE (default), all delay distribution priors will be weighted by the number of observation data points, in doing so approximately placing an independent prior at each time step and usually preventing the posteriors from shifting. If FALSE, no weight will be applied, i.e. delay distributions will be treated as a single parameters.

id — A character string used to assign logging information on error. Used by regional_epinow() to assign errors to regions. Alter the default to run with error catching.

verbose — Logical, defaults to TRUE when used interactively and otherwise FALSE. Should verbose debug progress messages be printed. Corresponds to the "DEBUG" level from futile.logger. See setup_logging for more detailed logging options.

reported_cases — Deprecated; use data instead.

Value

A list of output including: posterior samples, summarised posterior samples, data used to fit the model, and the fit object itself.

See Also

epinow() regional_epinow() forecast_infections() estimate_truncation()

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))

# get example case counts
reported_cases <- example_confirmed[1:60]

# set an example generation time. In practice this should use an estimate
# from the literature or be estimated from data
generation_time <- Gamma(
  shape = Normal(1.3, 0.3),
  rate = Normal(0.37, 0.09),
  max = 14
)
# set an example incubation period. In practice this should use an estimate
# from the literature or be estimated from data
incubation_period <- LogNormal(
   meanlog = Normal(1.6, 0.06),
   sdlog = Normal(0.4, 0.07),
   max = 14
)
# set an example reporting delay. In practice this should use an estimate
# from the literature or be estimated from data
reporting_delay <- LogNormal(mean = 2, sd = 1, max = 10)

# for more examples, see the "estimate_infections examples" vignette
def <- estimate_infections(reported_cases,
  generation_time = generation_time_opts(generation_time),
  delays = delay_opts(incubation_period + reporting_delay),
  rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
  stan = stan_opts(control = list(adapt_delta = 0.95))
)
# real time estimates
summary(def)
# summary plot
plot(def)
options(old_opts)
```

`estimate_secondary` *Estimate a Secondary Observation from a Primary Observation*

Description

[Stable] Estimates the relationship between a primary and secondary observation, for example hospital admissions and deaths or hospital admissions and bed occupancy. See `secondary_opts()` for model structure options. See parameter documentation for model defaults and options. See the examples for case studies using synthetic data and here for an example of forecasting Covid-19 deaths from Covid-19 cases. See here for a prototype function that may be used to estimate and forecast a secondary observation from a primary across multiple regions and here # nolint for an application forecasting Covid-19 deaths in Germany and Poland.

Usage

```
estimate_secondary(
  data,
  secondary = secondary_opts(),
 delays = delay_opts(LogNormal(meanlog = Normal(2.5, 0.5), sdlog = Normal(0.47, 0.25),
    max = 30), weight_prior = FALSE),
  truncation = trunc_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  burn_in = 14,
  CrIs = c(0.2, 0.5, 0.9),
  filter_leading_zeros = FALSE,
  zero_threshold = Inf,
  priors = NULL,
  model = NULL,
  weigh_delay_priors = FALSE,
  verbose = interactive(),
  ...,
  reports
)
```

Arguments

`data`	A <data.frame> containing the `date` of report and both `primary` and `secondary` reports.
`secondary`	A call to `secondary_opts()` or a list containing the following binary variables: cumulative, historic, primary_hist_additive, current, primary_current_additive. These parameters control the structure of the secondary model, see `secondary_opts()` for details.
`delays`	A call to `delay_opts()` defining delay distributions between primary and secondary observations See the documentation of `delay_opts()` for details. By default a diffuse prior is assumed with a mean of 14 days and standard deviation of 7 days (with a standard deviation of 0.5 and 0.25 respectively on the log scale).
`truncation`	A call to `trunc_opts()` defining the truncation of the observed data. Defaults to `trunc_opts()`, i.e. no truncation. See the `estimate_truncation()` help file for an approach to estimating this from data where the `dist` list element returned

by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.

obs — A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().

stan — A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.

burn_in — Integer, defaults to 14 days. The number of data points to use for estimation but not to fit to at the beginning of the time series. This must be less than the number of observations.

CrIs — Numeric vector of credible intervals to calculate.

filter_leading_zeros — Logical, defaults to TRUE. Should zeros at the start of the time series be filtered out.

zero_threshold — **[Experimental]** Numeric defaults to Inf. Indicates if detected zero cases are meaningful by using a threshold number of cases based on the 7-day average. If the average is above this threshold then the zero is replaced using fill.

priors — A <data.frame> of named priors to be used in model fitting rather than the defaults supplied from other arguments. This is typically useful if wanting to inform an estimate from the posterior of another model fit.

model — A compiled stan model to override the default model. May be useful for package developers or those developing extensions.

weigh_delay_priors — Logical. If TRUE, all delay distribution priors will be weighted by the number of observation data points, in doing so approximately placing an independent prior at each time step and usually preventing the posteriors from shifting. If FALSE (default), no weight will be applied, i.e. delay distributions will be treated as a single parameters.

verbose — Logical, should model fitting progress be returned. Defaults to interactive().

... — Additional parameters to pass to stan_opts().

reports — Deprecated; use data instead.

Value

A list containing: predictions (a <data.frame> ordered by date with the primary, and secondary observations, and a summary of the model estimated secondary observations), posterior which contains a summary of the entire model posterior, data (a list of data used to fit the model), and fit (the stanfit object).

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))

# load data.table for manipulation
```

```
library(data.table)

#### Incidence data example ####

# make some example secondary incidence data
cases <- example_confirmed
cases <- as.data.table(cases)[, primary := confirm]
# Assume that only 40 percent of cases are reported
cases[, scaling := 0.4]
# Parameters of the assumed log normal delay distribution
cases[, meanlog := 1.8][, sdlog := 0.5]

# Simulate secondary cases
cases <- convolve_and_scale(cases, type = "incidence")
#
# fit model to example data specifying a weak prior for fraction reported
# with a secondary case
inc <- estimate_secondary(cases[1:60],
  obs = obs_opts(scale = list(mean = 0.2, sd = 0.2), week_effect = FALSE)
)
plot(inc, primary = TRUE)

# forecast future secondary cases from primary
inc_preds <- forecast_secondary(
  inc, cases[seq(61, .N)][, value := primary]
)
plot(inc_preds, new_obs = cases, from = "2020-05-01")

#### Prevalence data example ####

# make some example prevalence data
cases <- example_confirmed
cases <- as.data.table(cases)[, primary := confirm]
# Assume that only 30 percent of cases are reported
cases[, scaling := 0.3]
# Parameters of the assumed log normal delay distribution
cases[, meanlog := 1.6][, sdlog := 0.8]

# Simulate secondary cases
cases <- convolve_and_scale(cases, type = "prevalence")

# fit model to example prevalence data
prev <- estimate_secondary(cases[1:100],
  secondary = secondary_opts(type = "prevalence"),
  obs = obs_opts(
    week_effect = FALSE,
    scale = list(mean = 0.4, sd = 0.1)
  )
)
plot(prev, primary = TRUE)

# forecast future secondary cases from primary
prev_preds <- forecast_secondary(
```

```
 prev, cases[seq(101, .N)][, value := primary]
)
plot(prev_preds, new_obs = cases, from = "2020-06-01")

options(old_opts)
```

`estimate_truncation` *Estimate Truncation of Observed Data*

Description

[Stable] Estimates a truncation distribution from multiple snapshots of the same data source over time. This distribution can then be used passed to the `truncation` argument in `regional_epinow()`, `epinow()`, and `estimate_infections()` to adjust for truncated data and propagate the uncertainty associated with data truncation into the estimates.

See here for an example of using this approach on Covid-19 data in England. The functionality offered by this function is now available in a more principled manner in the `epinowcast` R package.

The model of truncation is as follows:

1. The truncation distribution is assumed to be discretised log normal wit a mean and standard deviation that is informed by the data.
2. The data set with the latest observations is adjusted for truncation using the truncation distribution.
3. Earlier data sets are recreated by applying the truncation distribution to the adjusted latest observations in the time period of the earlier data set. These data sets are then compared to the earlier observations assuming a negative binomial observation model with an additive noise term to deal with zero observations.

This model is then fit using `stan` with standard normal, or half normal, prior for the mean, standard deviation, 1 over the square root of the overdispersion and additive noise term.

This approach assumes that:

- Current truncation is related to past truncation.
- Truncation is a multiplicative scaling of underlying reported cases.
- Truncation is log normally distributed.

Usage

```
estimate_truncation(
  data,
  max_truncation,
  trunc_max = 10,
  trunc_dist = "lognormal",
 truncation = trunc_opts(LogNormal(meanlog = Normal(0, 1), sdlog = Normal(1, 1), max =
    10)),
```

```
  model = NULL,
  stan = stan_opts(),
  CrIs = c(0.2, 0.5, 0.9),
  filter_leading_zeros = FALSE,
  zero_threshold = Inf,
  weigh_delay_priors = FALSE,
  verbose = TRUE,
  ...,
  obs
)
```

Arguments

data	A list of <data.frame>s each containing a date variable and a confirm (numeric) variable. Each data set should be a snapshot of the reported data over time. All data sets must contain a complete vector of dates.
max_truncation	Deprecated; use truncation instead.
trunc_max	Deprecated; use truncation instead.
trunc_dist	Deprecated; use truncation instead.
truncation	A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.
model	A compiled stan model to override the default model. May be useful for package developers or those developing extensions.
stan	A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.
CrIs	Numeric vector of credible intervals to calculate.
filter_leading_zeros	Logical, defaults to TRUE. Should zeros at the start of the time series be filtered out.
zero_threshold	**[Experimental]** Numeric defaults to Inf. Indicates if detected zero cases are meaningful by using a threshold number of cases based on the 7-day average. If the average is above this threshold then the zero is replaced using fill.
weigh_delay_priors	Deprecated; use the weight_prior option in trunc_opts() instead.
verbose	Logical, should model fitting progress be returned.
...	Additional parameters to pass to rstan::sampling().
obs	Deprecated; use data instead.

Value

A list containing: the summary parameters of the truncation distribution (dist), which could be passed to the truncation argument of epinow(), regional_epinow(), and estimate_infections(),

the estimated CMF of the truncation distribution (`cmf`, can be used to adjusted new data), a `<data.frame>` containing the observed truncated data, latest observed data and the adjusted for truncation observations (`obs`), a `<data.frame>` containing the last observed data (`last_obs`, useful for plotting and validation), the data used for fitting (`data`) and the fit object (`fit`).

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))

# fit model to example data
# See [example_truncated] for more details
est <- estimate_truncation(example_truncated,
  verbose = interactive(),
  chains = 2, iter = 2000
)

# summary of the distribution
est$dist
# summary of the estimated truncation cmf (can be applied to new data)
print(est$cmf)
# observations linked to truncation adjusted estimates
print(est$obs)
# validation plot of observations vs estimates
plot(est)

# Pass the truncation distribution to `epinow()`.
# Note, we're using the last snapshot as the observed data as it contains
# all the previous snapshots. Also, we're using the default options for
# illustrative purposes only.
out <- epinow(
  example_truncated[[5]],
  truncation = trunc_opts(est$dist)
)
plot(out)
options(old_opts)
```

---

example_confirmed *Example Confirmed Case Data Set*

Description

[Stable] An example data frame of observed cases

Usage

```
example_confirmed
```

Format

A data frame containing cases reported on each date.

`example_generation_time`
Example generation time

Description

[Stable] An example of a generation time estimate. See here for details: https://github.com/epiforecasts/EpiNow2/blob/main/
raw/generation-time.R

Usage

```
example_generation_time
```

Format

A `dist_spec` object summarising the distribution

`example_incubation_period`
Example incubation period

Description

[Stable] An example of an incubation period estimate. See here for details: https://github.com/epiforecasts/EpiNow2/blob/ma
raw/incubation-period.R # nolint

Usage

```
example_incubation_period
```

Format

A `dist_spec` object summarising the distribution

example_reporting_delay *Example reporting delay*

Description

[Stable] An example of an reporting delay estimate. See here for details: https://github.com/epiforecasts/EpiNow2/blob/main raw/reporting-delay # nolint

Usage

```
example_reporting_delay
```

Format

A `dist_spec` object summarising the distribution

example_truncated *Example Case Data Set with Truncation*

Description

[Stable] An example dataset of observed cases with truncation applied. This data is generated internally for use in the example of `estimate_truncation()`. For details on how the data is generated, see https://github.com/epiforecasts/EpiNow2/blob/main/data-raw/truncated.R #nolint

Usage

```
example_truncated
```

Format

A list of `data.tables` containing cases reported on each date until a point of truncation. Each element of the list is a `data.table` with the following columns:

date Date of case report.

confirm Number of confirmed cases.

expose_stan_fns *Expose internal package stan functions in R*

Description

[Stable] his function exposes internal stan functions in R from a user supplied list of target files. Allows for testing of stan functions in R and potentially user use in R code.

Usage

```
expose_stan_fns(files, target_dir, ...)
```

Arguments

files	A character vector indicating the target files.
target_dir	A character string indicating the target directory for the file.
...	Additional arguments passed to rstan::expose_stan_functions().

Value

No return value, called for side effects

extract_CrIs *Extract Credible Intervals Present*

Description

[Stable] Helper function to extract the credible intervals present in a <data.frame>.

Usage

```
extract_CrIs(summarised)
```

Arguments

summarised	A <data.frame> as processed by calc_CrIs

Value

A numeric vector of credible intervals detected in the <data.frame>.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
summarised <- calc_CrIs(samples,
  summarise_by = "type",
  CrIs = c(seq(0.05, 0.95, 0.05))
)
extract_CrIs(summarised)
```

`extract_inits` *Generate initial conditions from a Stan fit*

Description

[Experimental] Extracts posterior samples to use to initialise a full model fit. This may be useful for certain data sets where the sampler gets stuck or cannot easily be initialised. In `estimate_infections()`, `epinow()` and `regional_epinow()` this option can be engaged by setting `stan_opts(init_fit = <stanfit>)`.

This implementation is based on the approach taken in epidemia authored by James Scott.

Usage

```
extract_inits(fit, current_inits, exclude_list = NULL, samples = 50)
```

Arguments

`fit`	A `<stanfit>` object.
`current_inits`	A function that returns a list of initial conditions (such as `create_initial_conditions()`). Only used in `exclude_list` is specified.
`exclude_list`	A character vector of parameters to not initialise from the fit object, defaulting to `NULL`.
`samples`	Numeric, defaults to 50. Number of posterior samples.

Value

A function that when called returns a set of initial conditions as a named list.

extract_samples *Extract all samples from a stan fit*

Description

If the object argument is a <stanfit> object, it simply returns the result of rstan::extract(). If it is a <CmdStanMCMC> it returns samples in the same format as rstan::extract() does for <stanfit> objects.

Usage

```
extract_samples(stan_fit, pars = NULL, include = TRUE)
```

Arguments

stan_fit	A <stanfit> or <CmdStanMCMC> object as returned by fit_model().
pars	Any selection of parameters to extract
include	whether the parameters specified in pars should be included (TRUE, the default) or excluded (FALSE)

Value

List of data.tables with samples

extract_stan_param *Extract a Parameter Summary from a Stan Object*

Description

[Stable] Extracts summarised parameter posteriors from a stanfit object using rstan::summary() in a format consistent with other summary functions in {EpiNow2}.

Usage

```
extract_stan_param(
  fit,
  params = NULL,
  CrIs = c(0.2, 0.5, 0.9),
  var_names = FALSE
)
```

Arguments

fit	A <stanfit> objec.
params	A character vector of parameters to extract. Defaults to all parameters.
CrIs	Numeric vector of credible intervals to calculate.
var_names	Logical defaults to FALSE. Should variables be named. Automatically set to TRUE if multiple parameters are to be extracted.

Value

A <data.table> summarising parameter posteriors. Contains a following variables: variable, mean, mean_se, sd, median, and lower_, upper_ followed by credible interval labels indicating the credible intervals present.

---

filter_opts *Filter Options for a Target Region*

Description

[Maturing] A helper function that allows the selection of region specific settings if present and otherwise applies the overarching settings.

Usage

```
filter_opts(opts, region)
```

Arguments

opts	Either a list of calls to an _opts() function or a single call to an _opts() function.
region	A character string indicating a region of interest.

Value

A list of options

fix_dist *Fix the parameters of a* <dist_spec>

Description

[Experimental] If the given <dist_spec> has any uncertainty, it is removed and the corresponding distribution converted into a fixed one.

Usage

```
fix_dist(x, strategy = c("mean", "sample"))
```

Arguments

x	A <dist_spec>
strategy	Character; either "mean" (use the mean estimates of the mean and standard deviation) or "sample" (randomly sample mean and standard deviation from uncertainty given in the <dist_spec>

Value

A <dist_spec> object without uncertainty

Examples

```
# An uncertain gamma distribution with mean 3 and sd 2
dist <- LogNormal(
  meanlog = Normal(3, 0.5), sdlog = Normal(2, 0.5), max = 20
)

fix_dist(dist)
```

forecast_infections *Forecast infections from a given fit and trajectory of the time-varying reproduction number*

Description

[Stable] This function simulates infections using an existing fit to observed cases but with a modified time-varying reproduction number. This can be used to explore forecast models or past counterfactuals. Simulations can be run in parallel using `future::plan()`.

Usage

```
forecast_infections(
  estimates,
  R = NULL,
  model = NULL,
  samples = NULL,
  batch_size = 10,
  backend = "rstan",
  verbose = interactive()
)
```

Arguments

estimates	The estimates element of an epinow() run that has been done with output = "fit", or the result of estimate_infections() with return_fit set to TRUE.
R	A numeric vector of reproduction numbers; these will overwrite the reproduction numbers contained in estimates, except elements set to NA. Alternatively accepts a <data.frame> containing at least date and value (integer) variables and optionally sample. More (or fewer) days than in the original fit can be simulated.
model	A compiled stan model as returned by rstan::stan_model().
samples	Numeric, number of posterior samples to simulate from. The default is to use all samples in the estimates input.
batch_size	Numeric, defaults to 10. Size of batches in which to simulate. May decrease run times due to reduced IO costs but this is still being evaluated. If set to NULL then all simulations are done at once.
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
verbose	Logical defaults to interactive(). Should a progress bar (from progressr) be shown.

Value

A list of output as returned by estimate_infections() but based on results from the specified scenario rather than fitting.

See Also

dist_spec() generation_time_opts() delay_opts() rt_opts() estimate_infections() trunc_opts() stan_opts() obs_opts() gp_opts()

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))
```

```
# get example case counts
reported_cases <- example_confirmed[1:50]

# fit model to data to recover Rt estimates
est <- estimate_infections(reported_cases,
  generation_time = generation_time_opts(example_generation_time),
  delays = delay_opts(example_incubation_period + example_reporting_delay),
  rt = rt_opts(prior = list(mean = 2, sd = 0.1), rw = 7),
  stan = stan_opts(control = list(adapt_delta = 0.9)),
  obs = obs_opts(scale = list(mean = 0.1, sd = 0.01)),
  gp = NULL, horizon = 0
)

# update Rt trajectory and simulate new infections using it
R <- c(rep(NA_real_, 26), rep(0.5, 10), rep(0.8, 14))
sims <- forecast_infections(est, R)
plot(sims)

# with a data.frame input of samples
R_dt <- data.frame(
  date = seq(
    min(summary(est, type = "parameters", param = "R")$date),
    by = "day", length.out = length(R)
  ),
  value = R
)
sims <- forecast_infections(est, R_dt)
plot(sims)

#' # with a data.frame input of samples
R_samples <- summary(est, type = "samples", param = "R")
R_samples <- R_samples[,
 .(date, sample, value)][sample <= 1000][date <= "2020-04-10"
]
R_samples <- R_samples[date >= "2020-04-01", value := 1.1]
sims <- forecast_infections(est, R_samples)
plot(sims)

options(old_opts)
```

`forecast_secondary` *Forecast Secondary Observations Given a Fit from estimate_secondary*

Description

[Experimental] This function forecasts secondary observations using the output of `estimate_secondary()` and either observed primary data or a forecast of primary observations. See the examples of

estimate_secondary() for one use case. It can also be combined with estimate_infections() to produce a forecast for a secondary observation from a forecast of a primary observation. See the examples of estimate_secondary() for example use cases on synthetic data. See here for an example of forecasting Covid-19 deaths from Covid-19 cases.

Usage

```
forecast_secondary(
  estimate,
  primary,
  primary_variable = "reported_cases",
  model = NULL,
  backend = "rstan",
  samples = NULL,
  all_dates = FALSE,
  CrIs = c(0.2, 0.5, 0.9)
)
```

Arguments

estimate	An object of class "estimate_secondary" as produced by estimate_secondary().
primary	A <data.frame> containing at least date and value (integer) variables and optionally sample. Used as the primary observation used to forecast the secondary observations. Alternatively, this may be an object of class "estimate_infections" as produced by estimate_infections(). If primary is of class "estimate_infections" then the internal samples will be filtered to have a minimum date ahead of those observed in the estimate object.
primary_variable	A character string indicating the primary variable, defaulting to "reported_cases". Only used when primary is of class <estimate_infections>.
model	A compiled stan model as returned by rstan::stan_model().
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
samples	Numeric, number of posterior samples to simulate from. The default is to use all samples in the primary input when present. If not present the default is to use 1000 samples.
all_dates	Logical, defaults to FALSE. Should a forecast for all dates and not just those in the forecast horizon be returned.
CrIs	Numeric vector of credible intervals to calculate.

Value

A list containing: predictions (a <data.frame> ordered by date with the primary, and secondary observations, and a summary of the forecast secondary observations. For primary observations in the forecast horizon when uncertainty is present the median is used), samples a <data.frame> of forecast secondary observation posterior samples, and forecast a summary of the forecast secondary observation posterior.

See Also

estimate_secondary()

---

generation_times *Literature Estimates of Generation Times*

Description

[Deprecated] Generation time estimates. See here for details: https://github.com/epiforecasts/EpiNow2/blob/main/data-raw/generation-time.R

Usage

```
generation_times
```

Format

A `data.table` of summarising the distribution

---

generation_time_opts *Generation Time Distribution Options*

Description

[Stable] Returns generation time parameters in a format for lower level model use.

Usage

```
generation_time_opts(
  dist = Fixed(1),
  ...,
  disease,
  source,
  max = 14,
  fixed = FALSE,
  tolerance = 0.001,
  weight_prior = TRUE
)
```

Arguments

dist	A delay distribution or series of delay distributions . If no distribution is given a fixed generation time of 1 will be assumed.
...	deprecated; use dist instead
disease	deprecated; use dist instead
source	deprecated; use dist instead
max	deprecated; use dist instead
fixed	deprecated; use dist instead
tolerance	Numeric; the desired tolerance level.
weight_prior	Logical; if TRUE (default), any priors given in dist will be weighted by the number of observation data points, in doing so approximately placing an independent prior at each time step and usually preventing the posteriors from shifting. If FALSE, no weight will be applied, i.e. any parameters in dist will be treated as a single parameters.

Value

A <generation_time_opts> object summarising the input delay distributions.

See Also

convert_to_logmean() convert_to_logsd() bootstrapped_dist_fit() Gamma() LogNormal() Fixed()

Examples

```
# default settings with a fixed generation time of 1
generation_time_opts()

# A fixed gamma distributed generation time
generation_time_opts(Gamma(mean = 3, sd = 2, max = 14))

# An uncertain gamma distributed generation time
generation_time_opts(
  Gamma(
    mean = Normal(mean = 3, sd = 1),
    sd = Normal(mean = 2, sd = 0.5),
    max = 14
  )
)

# An example generation time
generation_time_opts(example_generation_time)
```

get_distribution *Get the distribution of a* dist_spec()

Description

[Experimental]

Usage

```
get_distribution(x, id = NULL)
```

Arguments

x	A <dist_spec>.
id	Integer; the id of the distribution to get parameters of (if x is a composite distribution). If x is a single distribution this is ignored and can be left as NULL.

Value

A character string naming the distribution (or "nonparametric")

Examples

```
dist <- Gamma(shape = 3, rate = 2, max = 10)
get_distribution(dist)
```

get_parameters *Get parameters of a parametric distribution*

Description

[Experimental]

Usage

```
get_parameters(x, id = NULL)
```

Arguments

x	A <dist_spec>.
id	Integer; the id of the distribution to get parameters of (if x is a composite distribution). If x is a single distribution this is ignored and can be left as NULL.

Value

A list of parameters of the distribution.

Examples

```
dist <- Gamma(shape = 3, rate = 2)
get_parameters(dist)
```

`get_pmf` *Get the probability mass function of a nonparametric distribution*

Description

[Experimental]

Usage

```
get_pmf(x, id = NULL)
```

Arguments

x	A <dist_spec>.
id	Integer; the id of the distribution to get parameters of (if x is a composite distribution). If x is a single distribution this is ignored and can be left as NULL.

Value

The pmf of the distribution

Examples

```
dist <- discretise(Gamma(shape = 3, rate = 2, max = 10))
get_pmf(dist)
```

`get_regional_results` *Get Combined Regional Results*

Description

[Stable] Summarises results across regions either from input or from disk. See the examples for details.

Usage

```
get_regional_results(
  regional_output,
  results_dir,
  date,
  samples = TRUE,
  forecast = FALSE
)
```

Arguments

regional_output	A list of output as produced by regional_epinow() and stored in the regional list.
results_dir	A character string indicating the folder containing the {EpiNow2} results to extract.
date	A Character string (in the format "yyyy-mm-dd") indicating the date to extract data for. Defaults to "latest" which finds the latest results available.
samples	Logical, defaults to TRUE. Should samples be returned.
forecast	Logical, defaults to FALSE. Should forecast results be returned.

Value

A list of estimates, forecasts and estimated cases by date of report.

Examples

```
# get example multiregion estimates
regional_out <- readRDS(system.file(
    package = "EpiNow2", "extdata", "example_regional_epinow.rds"
))

# from output
results <- get_regional_results(regional_out$regional, samples = FALSE)
```

---

gp_opts *Approximate Gaussian Process Settings*

Description

[Stable] Defines a list specifying the structure of the approximate Gaussian process. Custom settings can be supplied which override the defaults.

Usage

```
gp_opts(
  basis_prop = 0.2,
  boundary_scale = 1.5,
  ls_mean = 21,
  ls_sd = 7,
  ls_min = 0,
  ls_max = 60,
  alpha_sd = 0.05,
  kernel = c("matern_3/2", "se"),
  matern_type = 3/2
)
```

Arguments

basis_prop	Numeric, proportion of time points to use as basis functions. Defaults to 0.2. Decreasing this value results in a decrease in accuracy but a faster compute time (with increasing it having the first effect). In general smaller posterior length scales require a higher proportion of basis functions. See (Riutort-Mayol et al. 2020 https://arxiv.org/abs/2004.11408) for advice on updating this default.
boundary_scale	Numeric, defaults to 1.5. Boundary scale of the approximate Gaussian process. See (Riutort-Mayol et al. 2020 https://arxiv.org/abs/2004.11408) for advice on updating this default.
ls_mean	Numeric, defaults to 21 days. The mean of the lognormal length scale.
ls_sd	Numeric, defaults to 7 days. The standard deviation of the log normal length scale. If `ls_sd = 0`, inverse-gamma prior on Gaussian process length scale will be used with recommended parameters `inv_gamma(1.499007, 0.057277 * ls_max)`.
ls_min	Numeric, defaults to 0. The minimum value of the length scale.
ls_max	Numeric, defaults to 60. The maximum value of the length scale. Updated in `create_gp_data()` to be the length of the input data if this is smaller.
alpha_sd	Numeric, defaults to 0.05. The standard deviation of the magnitude parameter of the Gaussian process kernel. Should be approximately the expected standard deviation of the logged Rt.
kernel	Character string, the type of kernel required. Currently supporting the squared exponential kernel ("se") and the 3 over 2 Matern kernel ("matern", with `matern_type = 3/2`). Defaulting to the Matern 3 over 2 kernel as discontinuities are expected in Rt and infections.
matern_type	Numeric, defaults to 3/2. Type of Matern Kernel to use. Currently only the Matern 3/2 kernel is supported.

Value

A <gp_opts> object of settings defining the Gaussian process

Examples

```
# default settings
gp_opts()

# add a custom length scale
gp_opts(ls_mean = 4)
```

growth_to_R *Convert Growth Rates to Reproduction numbers.*

Description

[Questioning] See here # nolint for justification. Now handled internally by stan so may be removed in future updates if no user demand.

Usage

```
growth_to_R(r, gamma_mean, gamma_sd)
```

Arguments

r	Numeric, rate of growth estimates.
gamma_mean	Numeric, mean of the gamma distribution
gamma_sd	Numeric, standard deviation of the gamma distribution .

Value

Numeric vector of reproduction number estimates

Examples

```
growth_to_R(0.2, 4, 1)
```

incubation_periods *Literature Estimates of Incubation Periods*

Description

[Deprecated] Incubation period estimates. See here for details: https://github.com/epiforecasts/EpiNow2/blob/main/data-raw/incubation-period.R # nolint

Usage

```
incubation_periods
```

Format

A data.table of summarising the distribution

make_conf *Format Credible Intervals*

Description

[Stable] Combines a list of values into formatted credible intervals.

Usage

```
make_conf(value, CrI = 90, reverse = FALSE)
```

Arguments

value	List of value to map into a string. Requires, point, lower, and upper.
CrI	Numeric, credible interval to report. Defaults to 90.
reverse	Logical, defaults to FALSE. Should the reported credible interval be switched.

Value

A character vector formatted for reporting

Examples

```
value <- list(median = 2, lower_90 = 1, upper_90 = 3)
make_conf(value)
```

map_prob_change *Categorise the Probability of Change for Rt*

Description

[Stable] Categorises a numeric variable into "Increasing" (< 0.05), "Likely increasing" (<0.4), "Stable" (< 0.6), "Likely decreasing" (< 0.95), "Decreasing" (<= 1)

Usage

```
map_prob_change(var)
```

Arguments

var	Numeric variable to be categorised

Value

A character variable.

Examples

```
var <- seq(0.01, 1, 0.01)
var

map_prob_change(var)
```

`max.dist_spec` *Returns the maximum of one or more delay distribution*

Description

[Experimental] This works out the maximum of all the (parametric / nonparametric) delay distributions combined in the passed `dist_spec()` (ignoring any uncertainty in parameters)

Usage

```
## S3 method for class 'dist_spec'
max(x, ...)
```

Arguments

`x`	The `dist_spec()` to use
`...`	Not used

Value

A vector of means.

Examples

```
# A fixed gamma distribution with mean 5 and sd 1.
dist1 <- Gamma(mean = 5, sd = 1, max = 20)
max(dist1)

# An uncertain lognormal distribution with mean 3 and sd 2
dist2 <- LogNormal(mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20)
max(dist2)

# The max the sum of two distributions
max(dist1 + dist2)
```

mean.dist_spec *Returns the mean of one or more delay distribution*

Description

[Experimental] This works out the mean of all the (parametric / nonparametric) delay distributions combined in the passed `dist_spec()` (ignoring any uncertainty in parameters)

Usage

```
## S3 method for class 'dist_spec'
mean(x, ..., ignore_uncertainty = FALSE)
```

Arguments

x	The <dist_spec> to use
...	Not used
ignore_uncertainty	Logical; whether to ignore any uncertainty in parameters. If set to FALSE (the default) then the mean of any uncertain parameters will be returned as NA.

Examples

```
# A fixed lognormal distribution with mean 5 and sd 1.
dist1 <- LogNormal(mean = 5, sd = 1, max = 20)
mean(dist1)

# An uncertain gamma distribution with mean 3 and sd 2
dist2 <- Gamma(
 mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20
)
mean(dist2)

# The mean of the sum of two distributions
mean(dist1 + dist2)
```

new_dist_spec *Internal function for generating a* dist_spec *given parameters and a distribution.*

Description

[Experimental] This will convert all parameters to natural parameters before generating a `dist_spec`. If they have uncertainty this will be done using sampling.

Usage

```
new_dist_spec(params, distribution)
```

Arguments

params	Parameters of the distribution (including max)
distribution	Character; the distribution to use.

Value

A dist_spec of the given specification.

Examples

```
new_dist_spec(
  params = list(mean = 2, sd = 1, max = Inf),
  distribution = "normal"
)
```

---

obs_opts *Observation Model Options*

Description

[Stable] Defines a list specifying the structure of the observation model. Custom settings can be supplied which override the defaults.

Usage

```
obs_opts(
  family = c("negbin", "poisson"),
  phi = list(mean = 0, sd = 1),
  weight = 1,
  week_effect = TRUE,
  week_length = 7,
  scale = 1,
  na = c("missing", "accumulate"),
  likelihood = TRUE,
  return_likelihood = FALSE
)
```

Arguments

family	Character string defining the observation model. Options are Negative binomial ("negbin"), the default, and Poisson.

phi — Overdispersion parameter of the reporting process, used only if familiy is "negbin". Can be supplied either as a single numeric value (fixed overdispersion) or a list with numeric elements mean (mean) and standard deviation (sd) defining a normally distributed overdispersion. Defaults to a list with elements mean = 0 and sd = 1.

weight — Numeric, defaults to 1. Weight to give the observed data in the log density.

week_effect — Logical defaulting to TRUE. Should a day of the week effect be used in the observation model.

week_length — Numeric assumed length of the week in days, defaulting to 7 days. This can be modified if data aggregated over a period other than a week or if data has a non-weekly periodicity.

scale — Scaling factor to be applied to map latent infections (convolved to date of report). Can be supplied either as a single numeric value (fixed scale) or a list with numeric elements mean (mean) and standard deviation (sd) defining a normally distributed scaling factor. Defaults to 1, i.e. no scaling.

na — Character. Options are "missing" (the default) and "accumulate". This determines how NA values in the data are interpreted. If set to "missing", any NA values in the observation data set will be interpreted as missing and skipped in the likelihood. If set to "accumulate", modelled observations will be accumulated and added to the next non-NA data point. This can be used to model incidence data that is reported at less than daily intervals. If set to "accumulate", the first data point is not included in the likelihood but used only to reset modelled observations to zero.

likelihood — Logical, defaults to TRUE. Should the likelihood be included in the model.

return_likelihood — Logical, defaults to FALSE. Should the likelihood be returned by the model.

Value

An <obs_opts> object of observation model settings.

Examples

```
# default settings
obs_opts()

# Turn off day of the week effect
obs_opts(week_effect = TRUE)

# Scale reported data
obs_opts(scale = list(mean = 0.2, sd = 0.02))
```

opts_list *Return an _opts List per Region*

Description

[Maturing] Define a list of `_opts()` to pass to `regional_epinow()` `_opts()` accepting arguments. This is useful when different settings are needed between regions within a single `regional_epinow()` call. Using `opts_list()` the defaults can be applied to all regions present with an override passed to regions as necessary (either within `opts_list()` or externally).

Usage

```
opts_list(opts, reported_cases, ...)
```

Arguments

`opts`	An `_opts()` function call such as `rt_opts()`.
`reported_cases`	A data frame containing a `region` variable indicating the target regions.
`...`	Optional override for region defaults. See the examples for use case.

Value

A named list of options per region which can be passed to the `_opt` accepting arguments of `regional_epinow`.

See Also

`regional_epinow()` `rt_opts()`

Examples

```
# uses example case vector
cases <- example_confirmed[1:40]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]
))

# default settings
opts_list(rt_opts(), cases)

# add a weekly random walk in realland
opts_list(rt_opts(), cases, realland = rt_opts(rw = 7))

# add a weekly random walk externally
rt <- opts_list(rt_opts(), cases)
rt$realland$rw <- 7
rt
```

plot.dist_spec *Plot PMF and CDF for a dist_spec object*

Description

[Experimental] This function takes a <dist_spec> object and plots its probability mass function (PMF) and cumulative distribution function (CDF) using {ggplot2}. Note that currently uncertainty in distributions is not plot.

Usage

```
## S3 method for class 'dist_spec'
plot(x, ...)
```

Arguments

x	A <dist_spec> object
...	Additional arguments to pass to {ggplot}.

Examples

```
#' # A fixed lognormal distribution with mean 5 and sd 1.
dist1 <- LogNormal(mean = 1.6, sd = 0.5, max = 20)
plot(dist1)

# An uncertain gamma distribution with mean 3 and sd 2
dist2 <- Gamma(
  mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20
)
plot(dist2)

# Multiple distributions
plot(dist1 + dist2 + dist1)

# A combination of the two fixed distributions
plot(dist1 + dist1)
```

plot.epinow *Plot method for epinow*

Description

[Maturing] plot method for class <epinow>.

Usage

```
## S3 method for class 'epinow'
plot(x, type = "summary", ...)
```

Arguments

x	A list of output as produced by epinow().
type	A character vector indicating the name of the plot to return. Defaults to "summary" with supported options being "infections", "reports", "R", "growth_rate", "summary", "all". If "all" is supplied all plots are generated.
...	Pass additional arguments to report_plots

Value

List of plots as produced by report_plots()

See Also

plot plot.estimate_infections report_plots estimate_infections

---

plot.estimate_infections

Plot method for estimate_infections

Description

[Maturing] plot method for class <estimate_infections>.

Usage

```
## S3 method for class 'estimate_infections'
plot(
  x,
  type = c("summary", "infections", "reports", "R", "growth_rate", "all"),
  ...
)
```

Arguments

x	A list of output as produced by estimate_infections
type	A character vector indicating the name of the plot to return. Defaults to "summary" with supported options being "infections", "reports", "R", "growth_rate", "summary", "all". If "all" is supplied all plots are generated.
...	Pass additional arguments to report_plots

Value

List of plots as produced by report_plots()

See Also

plot report_plots estimate_infections

plot.estimate_secondary

Plot method for estimate_secondary

Description

[Experimental] `plot` method for class "estimate_secondary".

Usage

```
## S3 method for class 'estimate_secondary'
plot(x, primary = FALSE, from = NULL, to = NULL, new_obs = NULL, ...)
```

Arguments

`x`	A list of output as produced by `estimate_secondary`
`primary`	Logical, defaults to `FALSE`. Should `primary` reports also be plot?
`from`	Date object indicating when to plot from.
`to`	Date object indicating when to plot up to.
`new_obs`	A `<data.frame>` containing the columns `date` and `secondary` which replace the secondary observations stored in the `estimate_secondary` output.
`...`	Pass additional arguments to plot function. Not currently in use.

Value

A `ggplot` object.

See Also

plot estimate_secondary

plot.estimate_truncation

Plot method for estimate_truncation

Description

[Experimental] `plot()` method for class `<estimate_truncation>`. Returns a plot faceted over each dataset used in fitting with the latest observations as columns, the data observed at the time (and so truncated) as dots and the truncation adjusted estimates as a ribbon.

Usage

```
## S3 method for class 'estimate_truncation'
plot(x, ...)
```

Arguments

x	A list of output as produced by estimate_truncation()
...	Pass additional arguments to plot function. Not currently in use.

Value

ggplot2 object

See Also

plot estimate_truncation

---

plot_CrIs *Plot EpiNow2 Credible Intervals*

Description

[Stable] Adds lineranges for user specified credible intervals

Usage

```
plot_CrIs(plot, CrIs, alpha, linewidth)
```

Arguments

plot	A {ggplot2} plot
CrIs	Numeric list of credible intervals present in the data. As produced by extract_CrIs().
alpha	Numeric, overall alpha of the target line range
linewidth	Numeric, line width of the default line range.

Value

A {ggplot2} plot.

plot_estimates *Plot Estimates*

Description

[Questioning] Allows users to plot the output from `estimate_infections()` easily. In future releases it may be depreciated in favour of increasing the functionality of the S3 plot methods.

Usage

```
plot_estimates(
  estimate,
  reported,
  ylab,
  hline,
  obs_as_col = TRUE,
  max_plot = 10,
  estimate_type = c("Estimate", "Estimate based on partial data", "Forecast")
)
```

Arguments

estimate	A <data.table> of estimates containing the following variables: date, type (must contain "estimate", "estimate based on partial data" and optionally "forecast").
reported	A <data.table> of reported cases with the following variables: date, confirm.
ylab	Character string. Title for the plot y axis.
hline	Numeric, if supplied gives the horizontal intercept for a indicator line.
obs_as_col	Logical, defaults to TRUE. Should observed data, if supplied, be plotted using columns or as points (linked using a line).
max_plot	Numeric, defaults to 10. A multiplicative upper bound on the\ number of cases shown on the plot. Based on the maximum number of reported cases.
estimate_type	Character vector indicating the type of data to plot. Default to all types with supported options being: "Estimate", "Estimate based on partial data", and "Forecast".

Value

A ggplot2 object

Examples

```
# get example model results
out <- readRDS(system.file(
    package = "EpiNow2", "extdata", "example_estimate_infections.rds"
))
```

```
# plot infections
plot_estimates(
  estimate = out$summarised[variable == "infections"],
  reported = out$observations,
  ylab = "Cases", max_plot = 2
) + ggplot2::facet_wrap(~type, scales = "free_y")

# plot reported cases estimated via Rt
plot_estimates(
  estimate = out$summarised[variable == "reported_cases"],
  reported = out$observations,
  ylab = "Cases"
)

# plot Rt estimates
plot_estimates(
  estimate = out$summarised[variable == "R"],
  ylab = "Effective Reproduction No.",
  hline = 1
)

#' # plot Rt estimates without forecasts
plot_estimates(
  estimate = out$summarised[variable == "R"],
  ylab = "Effective Reproduction No.",
  hline = 1, estimate_type = "Estimate"
)
```

`plot_summary` *Plot a Summary of the Latest Results*

Description

[Questioning] Used to return a summary plot across regions (using results generated by `summarise_results()`). May be depreciated in later releases in favour of enhanced S3 methods.

Usage

```
plot_summary(summary_results, x_lab = "Region", log_cases = FALSE, max_cases)
```

Arguments

`summary_results`	A data.table as returned by `summarise_results()` (the `data` object).
`x_lab`	A character string giving the label for the x axis, defaults to region.
`log_cases`	Logical, should cases be shown on a logged scale. Defaults to `FALSE`.
`max_cases`	Numeric, no default. The maximum number of cases to plot.

Value

A {ggplot2} object

`print.dist_spec` *Prints the parameters of one or more delay distributions*

Description

[Experimental] This displays the parameters of the uncertain and probability mass functions of fixed delay distributions combined in the passed `dist_spec()`.

Usage

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

Arguments

x	The <dist_spec> to use
...	Not used

Value

invisible

Examples

```
#' # A fixed lognormal distribution with mean 5 and sd 1.
dist1 <- LogNormal(mean = 1.5, sd = 0.5, max = 20)
print(dist1)

# An uncertain gamma distribution with mean 3 and sd 2
dist2 <- Gamma(
  mean = Normal(3, 0.5), sd = Normal(2, 0.5), max = 20
)
print(dist2)
```

regional_epinow *Real-time Rt Estimation, Forecasting and Reporting by Region*

Description

[Maturing] Efficiently runs `epinow()` across multiple regions in an efficient manner and conducts basic data checks and cleaning such as removing regions with fewer than `non_zero_points` as these are unlikely to produce reasonable results whilst consuming significant resources. See the documentation for `epinow()` for further information.

By default all arguments supporting input from `_opts()` functions are shared across regions (including delays, truncation, Rt settings, stan settings, and gaussian process settings). Region specific settings are supported by passing a named list of `_opts()` calls (with an entry per region) to the relevant argument. A helper function (`opts_list()`) is available to facilitate building this list.

Regions can be estimated in parallel using the `{future}` package (see `setup_future()`). The progress of producing estimates across multiple regions is tracked using the `{progressr}` package. Modify this behaviour using `progressr::handlers()` and enable it in batch by setting `R_PROGRESSR_ENABLE=TRUE` as an environment variable.

Usage

```
regional_epinow(
  data,
  generation_time = generation_time_opts(),
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  target_folder = NULL,
  target_date,
  non_zero_points = 2,
  output = c("regions", "summary", "samples", "plots", "latest"),
  return_output = FALSE,
  summary_args = list(),
  verbose = FALSE,
  logs = tempdir(check = TRUE),
  ...,
  reported_cases
)
```

Arguments

data	A <data.frame> of confirmed cases (confirm) by date (date), and region (region).

generation_time
: A call to generation_time_opts() defining the generation time distribution used. For backwards compatibility a list of summary parameters can also be passed.

delays
: A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.

truncation
: A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.

rt
: A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.

backcalc
: A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().

gp
: A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts(). Set to NULL to disable the Gaussian process.

obs
: A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().

stan
: A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.

horizon
: Numeric, defaults to 7. Number of days into the future to forecast.

CrIs
: Numeric vector of credible intervals to calculate.

target_folder
: Character string specifying where to save results (will create if not present).

target_date
: Date, defaults to maximum found in the data if not specified.

non_zero_points
: Numeric, the minimum number of time points with non-zero cases in a region required for that region to be evaluated. Defaults to 7.

output
: A character vector of optional output to return. Supported options are the individual regional estimates ("regions"), samples ("samples"), plots ("plots"), copying the individual region dated folder into a latest folder (if target_folder is not null, set using "latest"), the stan fit of the underlying model ("fit"), and an overall summary across regions ("summary"). The default is to return samples and plots alongside summarised estimates and summary statistics. If target_folder is not NULL then the default is also to copy all results into a latest folder.

return_output
: Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.

summary_args
: A list of arguments passed to regional_summary(). See the regional_summary() documentation for details.

verbose
: Logical defaults to FALSE. Outputs verbose progress messages to the console from epinow().

logs	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if logs is set to NULL. If specifying a custom logging setup then the code for setup_default_logging() and the setup_logging() function are a sensible place to start.
...	Pass additional arguments to epinow(). See the documentation for epinow() for details.
reported_cases	Deprecated; use data instead.

Value

A list of output stratified at the top level into regional output and across region output summary output

See Also

epinow() estimate_infections() setup_future() regional_summary()

Examples

```
# set number of cores to use
old_opts <- options()
options(mc.cores = ifelse(interactive(), 4, 1))

# uses example case vector
cases <- example_confirmed[1:60]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]
))

# run epinow across multiple regions and generate summaries
# samples and warmup have been reduced for this example
# for more examples, see the "estimate_infections examples" vignette
def <- regional_epinow(
  data = cases,
  generation_time = generation_time_opts(example_generation_time),
  delays = delay_opts(example_incubation_period + example_reporting_delay),
  rt = rt_opts(prior = list(mean = 2, sd = 0.2)),
  stan = stan_opts(
    samples = 100, warmup = 200,
    control = list(adapt_delta = 0.95)
  ),
  verbose = interactive()
)
options(old_opts)
```

regional_summary *Regional Summary Output*

Description

[Maturing] Used to produce summary output either internally in `regional_epinow` or externally.

Usage

```
regional_summary(
  regional_output = NULL,
  data,
  results_dir = NULL,
  summary_dir = NULL,
  target_date = NULL,
  region_scale = "Region",
  all_regions = TRUE,
  return_output = FALSE,
  plot = TRUE,
  max_plot = 10,
  ...
)
```

Arguments

`regional_output`	A list of output as produced by `regional_epinow()` and stored in the `regional` list.
`data`	A <`data.frame`> of confirmed cases (confirm) by date (date), and region (`region`).
`results_dir`	An optional character string indicating the location of the results directory to extract results from.
`summary_dir`	A character string giving the directory in which to store summary of results.
`target_date`	A character string giving the target date for which to extract results (in the format "yyyy-mm-dd"). Defaults to latest available estimates.
`region_scale`	A character string indicating the name to give the regions being summarised.
`all_regions`	Logical, defaults to `TRUE`. Should summary plots for all regions be returned rather than just regions of interest.
`return_output`	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
`plot`	Logical, defaults to `TRUE`. Should regional summary plots be produced.
`max_plot`	Numeric, defaults to 10. A multiplicative upper bound on the\ number of cases shown on the plot. Based on the maximum number of reported cases.
`...`	Additional arguments passed to `report_plots`.

Value

A list of summary measures and plots

See Also

regional_epinow

Examples

```
# get example output from regional_epinow model
regional_out <- readRDS(system.file(
    package = "EpiNow2", "extdata", "example_regional_epinow.rds"
))

regional_summary(
  regional_output = regional_out$regional,
  data = regional_out$summary$reported_cases
)
```

`report_plots` *Report plots*

Description

[Questioning] Returns key summary plots for estimates. May be depreciated in later releases as current S3 methods are enhanced.

Usage

```
report_plots(summarised_estimates, reported, target_folder = NULL, ...)
```

Arguments

`summarised_estimates`	A data.table of summarised estimates containing the following variables: variable, median, bottom, and top. It should also contain the following estimates: R, infections, reported_cases_rt, and r (rate of growth).
`reported`	A `<data.table>` of reported cases with the following variables: date, confirm.
`target_folder`	Character string specifying where to save results (will create if not present).
`...`	Additional arguments passed to `plot_estimates()`.

Value

A named list of ggplot2 objects, `list(infections, reports, R, growth_rate, summary)`, which correspond to a summary combination (last item) and for the leading items.

See Also

plot_estimates() of summarised_estimates[variable == "infections"], summarised_estimates[variable == "reported_cases"], summarised_estimates[variable == "R"], and summarised_estimates[variable == "growth_rate"], respectively.

Examples

```
# get example output form estimate_infections
out <- readRDS(system.file(
    package = "EpiNow2", "extdata", "example_estimate_infections.rds"
))

# plot infections
plots <- report_plots(
  summarised_estimates = out$summarised,
  reported = out$observations
)
plots
```

---

report_summary *Provide Summary Statistics for Estimated Infections and Rt*

Description

[Questioning] Creates a snapshot summary of estimates. May be removed in later releases as S3 methods are enhanced.

Usage

```
report_summary(
  summarised_estimates,
  rt_samples,
  target_folder = NULL,
  return_numeric = FALSE
)
```

Arguments

summarised_estimates	A data.table of summarised estimates containing the following variables: variable, median, bottom, and top. It should contain the following estimates: R, infections, and r (rate of growth).
rt_samples	A data.table containing Rt samples with the following variables: sample and value.
target_folder	Character string specifying where to save results (will create if not present).
return_numeric	Should numeric summary information be returned.

Value

A data.table containing formatted and numeric summary measures

---

`rstan_opts` *Rstan Options*

Description

[Deprecated] Deprecated; specify options in `stan_opts()` instead.

Usage

```
rstan_opts(object = NULL, samples = 2000, method = c("sampling", "vb"), ...)
```

Arguments

`object`	Stan model object. By default uses the compiled package default.
`samples`	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
`method`	A character string, defaulting to sampling. Currently supports `rstan::sampling()` ("sampling") or `rstan::vb()`.
`...`	Additional parameters to pass underlying option functions.

Value

A list of arguments to pass to the appropriate rstan functions.

See Also

`rstan_sampling_opts()` `rstan_vb_opts()`

`rstan_sampling_opts` *Rstan Sampling Options*

Description

[Deprecated] Deprecated; use `stan_sampling_opts()` instead.

Usage

```
rstan_sampling_opts(
  cores = getOption("mc.cores", 1L),
  warmup = 250,
  samples = 2000,
  chains = 4,
  control = list(),
  save_warmup = FALSE,
  seed = as.integer(runif(1, 1, 1e+08)),
  future = FALSE,
  max_execution_time = Inf,
  ...
)
```

Arguments

cores	Number of cores to use when executing the chains in parallel, which defaults to 1 but it is recommended to set the mc.cores option to be as many processors as the hardware and RAM allow (up to the number of chains).
warmup	Numeric, defaults to 250. Number of warmup samples per chain.
samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
chains	Numeric, defaults to 4. Number of MCMC chains to use.
control	List, defaults to empty. control parameters to pass to underlying `rstan` function. By default `adapt_delta` = `0.95` and `max_treedepth` = `15` though these settings can be overwritten.
save_warmup	Logical, defaults to FALSE. Should warmup progress be saved.
seed	Numeric, defaults uniform random number between 1 and 1e8. Seed of sampling process.
future	Logical, defaults to `FALSE`. Should stan chains be run in parallel using `future`. This allows users to have chains fail gracefully (i.e when combined with `max_execution_time`). Should be combined with a call to `future::plan()`.
max_execution_time	Numeric, defaults to Inf (seconds). If set wil kill off processing of each chain if not finished within the specified timeout. When more than 2 chains finish successfully estimates will still be returned. If less than 2 chains return within the allowed time then estimation will fail with an informative error.
...	Additional parameters to pass to `rstan::sampling()`.

Value

A list of arguments to pass to `rstan::sampling()`.

rstan_vb_opts *Rstan Variational Bayes Options*

Description

[Deprecated] Deprecated; use `stan_vb_opts()` instead.

Usage

```
rstan_vb_opts(samples = 2000, trials = 10, iter = 10000, ...)
```

Arguments

`samples`	Numeric, default 2000. Overall number of approximate posterior samples.
`trials`	Numeric, defaults to 10. Number of attempts to use rstan::vb()] before failing.
`iter`	Numeric, defaulting to 10000. Number of iterations to use in `rstan::vb()`.
`...`	Additional parameters to pass to `rstan::vb()` or `cmdstanr::variational()`, depending on the chosen backend.

Value

A list of arguments to pass to `rstan::vb()`.

rt_opts *Time-Varying Reproduction Number Options*

Description

[Stable] Defines a list specifying the optional arguments for the time-varying reproduction number. Custom settings can be supplied which override the defaults.

Usage

```
rt_opts(
  prior = list(mean = 1, sd = 1),
  use_rt = TRUE,
  rw = 0,
  use_breakpoints = TRUE,
  future = "latest",
  gp_on = c("R_t-1", "R0"),
  pop = 0
)
```

Arguments

prior	List containing named numeric elements "mean" and "sd". The mean and standard deviation of the log normal Rt prior. Defaults to mean of 1 and standard deviation of 1.
use_rt	Logical, defaults to TRUE. Should Rt be used to generate infections and hence reported cases.
rw	Numeric step size of the random walk, defaults to 0. To specify a weekly random walk set rw = 7. For more custom break point settings consider passing in a breakpoints variable as outlined in the next section.
use_breakpoints	Logical, defaults to TRUE. Should break points be used if present as a breakpoint variable in the input data. Break points should be defined as 1 if present and otherwise 0. By default breakpoints are fit jointly with a global non-parametric effect and so represent a conservative estimate of break point changes (alter this by setting gp = NULL).
future	A character string or integer. This argument indicates how to set future Rt values. Supported options are to project using the Rt model ("project"), to use the latest estimate based on partial data ("latest"), to use the latest estimate based on data that is over 50% complete ("estimate"). If an integer is supplied then the Rt estimate from this many days into the future (or past if negative) past will be used forwards in time.
gp_on	Character string, defaulting to "R_t-1". Indicates how the Gaussian process, if in use, should be applied to Rt. Currently supported options are applying the Gaussian process to the last estimated Rt (i.e Rt = Rt-1 * GP), and applying the Gaussian process to a global mean (i.e Rt = R0 * GP). Both should produced comparable results when data is not sparse but the method relying on a global mean will revert to this for real time estimates, which may not be desirable.
pop	Integer, defaults to 0. Susceptible population initially present. Used to adjust Rt estimates when otherwise fixed based on the proportion of the population that is susceptible. When set to 0 no population adjustment is done.

Value

An <rt_opts> object with settings defining the time-varying reproduction number.

Examples

```
# default settings
rt_opts()

# add a custom length scale
rt_opts(prior = list(mean = 2, sd = 1))

# add a weekly random walk
rt_opts(rw = 7)
```

`run_region` *Run epinow with Regional Processing Code*

Description

[Maturing] Internal function that handles calling `epinow()`. Future work will extend this function to better handle stan logs and allow the user to modify settings between regions.

Usage

```
run_region(
  target_region,
  generation_time,
  delays,
  truncation,
  rt,
  backcalc,
  gp,
  obs,
  stan,
  horizon,
  CrIs,
  data,
  target_folder,
  target_date,
  return_output,
  output,
  complete_logger,
  verbose,
  progress_fn,
  ...
)
```

Arguments

`target_region`	Character string indicating the region being evaluated
`generation_time`	A call to `generation_time_opts()` defining the generation time distribution used. For backwards compatibility a list of summary parameters can also be passed.
`delays`	A call to `delay_opts()` defining delay distributions and options. See the documentation of `delay_opts()` and the examples below for details.
`truncation`	A call to `trunc_opts()` defining the truncation of the observed data. Defaults to `trunc_opts()`, i.e. no truncation. See the `estimate_truncation()` help file for an approach to estimating this from data where the `dist` list element returned by `estimate_truncation()` is used as the `truncation` argument here, thereby propagating the uncertainty in the estimate.

rt	A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().
gp	A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts(). Set to NULL to disable the Gaussian process.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().
stan	A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
data	A <data.frame> of confirmed cases (confirm) by date (date), and region (region).
target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
output	A character vector of optional output to return. Supported options are the individual regional estimates ("regions"), samples ("samples"), plots ("plots"), copying the individual region dated folder into a latest folder (if target_folder is not null, set using "latest"), the stan fit of the underlying model ("fit"), and an overall summary across regions ("summary"). The default is to return samples and plots alongside summarised estimates and summary statistics. If target_folder is not NULL then the default is also to copy all results into a latest folder.
complete_logger	Character string indicating the logger to output the completion of estimation to.
verbose	Logical defaults to FALSE. Outputs verbose progress messages to the console from epinow().
progress_fn	Function as returned by progressr::progressor(). Allows the use of a progress bar.
...	Pass additional arguments to epinow(). See the documentation for epinow() for details.

Value

A list of processed output as produced by process_region()

See Also

regional_epinow()

R_to_growth *Convert Reproduction Numbers to Growth Rates*

Description

[Questioning] See here # nolint for justification. Now handled internally by stan so may be removed in future updates if no user demand.

Usage

```
R_to_growth(R, gamma_mean, gamma_sd)
```

Arguments

R	Numeric, Reproduction number estimates
gamma_mean	Numeric, mean of the gamma distribution
gamma_sd	Numeric, standard deviation of the gamma distribution .

Value

Numeric vector of reproduction number estimates

Examples

```
R_to_growth(2.18, 4, 1)
```

secondary_opts *Secondary Reports Options*

Description

[Stable] Returns a list of options defining the secondary model used in `estimate_secondary()`. This model is a combination of a convolution of previously observed primary reports combined with current primary reports (either additive or subtractive). It can optionally be cumulative. See the documentation of `type` for sensible options to cover most use cases and the returned values of `secondary_opts()` for all currently supported options.

Usage

```
secondary_opts(type = c("incidence", "prevalence"), ...)
```

Arguments

type	A character string indicating the type of observation the secondary reports are. Options include: • "incidence": Assumes that secondary reports equal a convolution of previously observed primary reported cases. An example application is deaths from an infectious disease predicted by reported cases of that disease (or estimated infections). • "prevalence": Assumes that secondary reports are cumulative and are defined by currently observed primary reports minus a convolution of secondary reports. An example application is hospital bed usage predicted by hospital admissions.
...	Overwrite options defined by type. See the returned values for all options that can be passed.

Value

A <secondary_opts> object of binary options summarising secondary model used in estimate_secondary(). Options returned are cumulative (should the secondary report be cumulative), historic (should a convolution of primary reported cases be used to predict secondary reported cases), primary_hist_additive (should the historic convolution of primary reported cases be additive or subtractive), current (should currently observed primary reported cases contribute to current secondary reported cases), primary_current_additive (should current primary reported cases be additive or subtractive).

See Also

estimate_secondary()

Examples

```
# incidence model
secondary_opts("incidence")

# prevalence model
secondary_opts("prevalence")
```

---

setup_default_logging *Setup Default Logging*

Description

[Questioning] Sets up default logging. Usage of logging is currently being explored as the current setup cannot log stan errors or progress.

Usage

```
setup_default_logging(
  logs = tempdir(check = TRUE),
  mirror_epinow = FALSE,
  target_date = NULL
)
```

Arguments

logs	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if logs is set to NULL. If specifying a custom logging setup then the code for setup_default_logging() and the setup_logging() function are a sensible place to start.
mirror_epinow	Logical, defaults to FALSE. Should internal logging be returned from epinow() to the console.
target_date	Date, defaults to maximum found in the data if not specified.

Value

No return value, called for side effects

Examples

```
setup_default_logging()
```

---

setup_future *Set up Future Backend*

Description

[Stable] A utility function that aims to streamline the set up of the required future backend with sensible defaults for most users of regional_epinow(). More advanced users are recommended to setup their own {future} backend based on their available resources.

Usage

```
setup_future(
  data,
  strategies = c("multisession", "multisession"),
  min_cores_per_worker = 4
)
```

Arguments

data	A <data.frame> of confirmed cases (confirm) by date (date), and region (region).
strategies	A vector length 1 to 2 of strategies to pass to future::plan(). Nesting of parallelisation is from the top level down. The default is to set up nesting parallelisation with both using future::multisession() (future::multicore() will likely be a faster option on supported platforms). For single level parallelisation use a single strategy or future::plan() directly. See future::plan() for options.
min_cores_per_worker	Numeric, the minimum number of cores per worker. Defaults to 4 which assumes 4 MCMC chains are in use per region.

Value

Numeric number of cores to use per worker. If greater than 1 pass to stan_args = list(cores = "output from setup future") or use future = TRUE. If only a single strategy is used then nothing is returned.

---

setup_logging *Setup Logging*

Description

[Questioning] Sets up {futile.logger} logging, which is integrated into {EpiNow2}. See the documentation for {futile.logger} for full details. By default {EpiNow2} prints all logs at the "INFO" level and returns them to the console. Usage of logging is currently being explored as the current setup cannot log stan errors or progress.

Usage

```
setup_logging(
  threshold = "INFO",
  file = NULL,
  mirror_to_console = FALSE,
  name = "EpiNow2"
)
```

Arguments

threshold	Character string indicating the logging level see (?futile.logger for details of the available options). Defaults to "INFO".
file	Character string indicating the path to save logs to. By default logs will be written to the console.
mirror_to_console	Logical, defaults to FALSE. If saving logs to a file should they also be duplicated in the console.

name — Character string defaulting to EpiNow2. This indicates the name of the logger to setup. The default logger for EpiNow2 is called EpiNow2. Nested options include: Epinow2.epinow which controls all logging for `epinow()` and nested functions, EpiNow2.epinow.estimate_infections (logging in `estimate_infections()`, and EpiNow2.epinow.estimate_infections.fit (logging in fitting functions).

Value

Nothing

`simulate_infections` *Simulate infections using the renewal equation*

Description

Simulations are done from given initial infections and, potentially time-varying, reproduction numbers. Delays and parameters of the observation model can be specified using the same options as in `estimate_infections()`.

Usage

```
simulate_infections(
  estimates,
  R,
  initial_infections,
  day_of_week_effect = NULL,
  generation_time = generation_time_opts(),
  delays = delay_opts(),
  truncation = trunc_opts(),
  obs = obs_opts(),
  CrIs = c(0.2, 0.5, 0.9),
  backend = "rstan",
  seeding_time = NULL,
  pop = 0,
  ...
)
```

Arguments

estimates — deprecated; use `forecast_infections()` instead

R — a data frame of reproduction numbers (column `R`) by date (column `date`). Column `R` must be numeric and `date` must be in date format. If not all days between the first and last day in the `date` are present, it will be assumed that R stays the same until the next given date.

initial_infections — numeric; the initial number of infections (i.e. before `R` applies). Note that results returned start the day after, i.e. the initial number of infections is not reported again. See also `seeding_time`

day_of_week_effect
: either NULL (no day of the week effect) or a numerical vector of length specified in obs_opts() as week_length (default: 7) if week_effect is set to TRUE. Each element of the vector gives the weight given to reporting on this day (normalised to 1). The default is NULL.

generation_time
: A call to generation_time_opts() defining the generation time distribution used. For backwards compatibility a list of summary parameters can also be passed.

delays
: A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.

truncation
: A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.

obs
: A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().

CrIs
: Numeric vector of credible intervals to calculate.

backend
: Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".

seeding_time
: Integer; the number of days before the first time point of R; default is NULL, in which case it is set to the maximum of the generation time. The minimum is 1 , i.e. the first reproduction number given applies on the day after the index cases given by initial_infections. If the generation time is longer than 1 day on average, a seeding time of 1 will always lead to an initial decline (as there are no infections before the initial ones). Instead, if this is greater than 1, an initial part of the epidemic (before the first value of R given) of seeding_time days is assumed to have followed exponential growth roughly in line with the growth rate implied by the first value of R.

pop
: Integer, defaults to 0. Susceptible population initially present. Used to adjust Rt estimates when otherwise fixed based on the proportion of the population that is susceptible. When set to 0 no population adjustment is done.

...
: deprecated; only included for backward compatibility

Details

In order to simulate, all parameters that are specified such as the mean and standard deviation of delays or observation scaling, must be fixed. Uncertain parameters are not allowed.

A previous function called simulate_infections() that simulates from a given model fit has been renamed forecast_infections(). Using simulate_infections() with existing estimates is now deprecated. This option will be removed in the next version.

Value

A data.table of simulated infections (variable infections) and reported cases (variable reported_cases) by date.

Examples

```
R <- data.frame(
  date = seq.Date(as.Date("2023-01-01"), length.out = 14, by = "day"),
  R = c(rep(1.2, 7), rep(0.8, 7))
)
sim <- simulate_infections(
  R = R,
  initial_infections = 100,
  generation_time = generation_time_opts(
    fix_dist(example_generation_time)
  ),
  delays = delay_opts(fix_dist(example_reporting_delay)),
  obs = obs_opts(family = "poisson")
)
```

---

simulate_secondary *Simulate secondary observations from primary observations*

Description

Simulations are done from a given trajectory of primary observations by applying any given delays and observation parameters.

Usage

```
simulate_secondary(
  primary,
  day_of_week_effect = NULL,
  secondary = secondary_opts(),
  delays = delay_opts(),
  truncation = trunc_opts(),
  obs = obs_opts(),
  CrIs = c(0.2, 0.5, 0.9),
  backend = "rstan",
  ...
)
```

Arguments

primary	a data frame of primary reports (column `primary`) by date (column `date`). Column `primary` must be numeric and `date` must be in date format. it will be assumed that `primary` is zero on the missing days.
day_of_week_effect	either `NULL` (no day of the week effect) or a numerical vector of length specified in `obs_opts()` as `week_length` (default: 7) if `week_effect` is set to TRUE. Each element of the vector gives the weight given to reporting on this day (normalised to 1). The default is `NULL`.

secondary	A call to secondary_opts() or a list containing the following binary variables: cumulative, historic, primary_hist_additive, current, primary_current_additive. These parameters control the structure of the secondary model, see secondary_opts() for details.
delays	A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.
truncation	A call to trunc_opts() defining the truncation of the observed data. Defaults to trunc_opts(), i.e. no truncation. See the estimate_truncation() help file for an approach to estimating this from data where the dist list element returned by estimate_truncation() is used as the truncation argument here, thereby propagating the uncertainty in the estimate.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().
CrIs	Numeric vector of credible intervals to calculate.
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
...	deprecated; only included for backward compatibility

Details

In order to simulate, all parameters that are specified such as the mean and standard deviation of delays or observation scaling, must be fixed. Uncertain parameters are not allowed.

A function of the same name that was previously based on a reimplementation of that model in R with potentially time-varying scalings and delays is available as ‘convolve_and_scale()

Value

A data.table of simulated secondary observations (column secondary) by date.

Examples

```
## load data.table to manipulate `example_confirmed` below
library(data.table)
cases <- as.data.table(example_confirmed)[, primary := confirm]
sim <- simulate_secondary(
  cases,
  delays = delay_opts(fix_dist(example_reporting_delay)),
  obs = obs_opts(family = "poisson")
)
```

stan_laplace_opts *Stan Laplace algorithm Options*

Description

[Experimental] Defines a list specifying the arguments passed to `cmdstanr::laplace()`.

Usage

```
stan_laplace_opts(backend = "cmdstanr", trials = 10, ...)
```

Arguments

backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
trials	Numeric, defaults to 10. Number of attempts to use rstan::vb()] before failing.
...	Additional parameters to pass to `cmdstanr::laplace()`.

Value

A list of arguments to pass to `cmdstanr::laplace()`.

Examples

```
stan_laplace_opts()
```

stan_opts *Stan Options*

Description

[Stable] Defines a list specifying the arguments passed to underlying stan backend functions via `stan_sampling_opts()` and `stan_vb_opts()`. Custom settings can be supplied which override the defaults.

Usage

```
stan_opts(
  object = NULL,
  samples = 2000,
  method = c("sampling", "vb", "laplace", "pathfinder"),
  backend = c("rstan", "cmdstanr"),
  init_fit = NULL,
  return_fit = TRUE,
  ...
)
```

Arguments

object	Stan model object. By default uses the compiled package default if using the "rstan" backend, and the default model obtained using epinow2_cmdstan_model() if using the "cmdstanr" backend.
samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
method	A character string, defaulting to sampling. Currently supports MCMC sampling ("sampling") or approximate posterior sampling via variational inference ("vb") and, as experimental features if the "cmdstanr" backend is used, approximate posterior sampling with the laplace algorithm ("laplace") or pathfinder ("pathfinder").
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
init_fit	**[Experimental]** Character string or stanfit object, defaults to NULL. Should an initial fit be used to initialise the full fit. An example scenario would be using a national level fit to parametrise regional level fits. Optionally a character string can be passed with the currently supported option being "cumulative". This fits the model to cumulative cases and may be useful for certain data sets where the sampler gets stuck or struggles to initialise. See init_cumulative_fit() for details. This implementation is based on the approach taken in epidemia authored by James Scott. This argument is deprecated and the default (NULL) will be used from the next version.
return_fit	Logical, defaults to TRUE. Should the fit stan model be returned.
...	Additional parameters to pass to underlying option functions, stan_sampling_opts() or stan_vb_opts(), depending on the method

Value

A <stan_opts> object of arguments to pass to the appropriate rstan functions.

See Also

stan_sampling_opts() stan_vb_opts()

Examples

```
# using default of [rstan::sampling()]
stan_opts(samples = 1000)

# using vb
stan_opts(method = "vb")
```

stan_pathfinder_opts *Stan pathfinder algorithm Options*

Description

[Experimental] Defines a list specifying the arguments passed to `cmdstanr::laplace()`.

Usage

```
stan_pathfinder_opts(backend = "cmdstanr", samples = 2000, trials = 10, ...)
```

Arguments

backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
trials	Numeric, defaults to 10. Number of attempts to use rstan::vb()] before failing.
...	Additional parameters to pass to `cmdstanr::laplace()`.

Value

A list of arguments to pass to `cmdstanr::laplace()`.

Examples

```
stan_laplace_opts()
```

stan_sampling_opts *Stan Sampling Options*

Description

[Stable] Defines a list specifying the arguments passed to either `rstan::sampling()` or `cmdstanr::sample()`. Custom settings can be supplied which override the defaults.

Usage

```
stan_sampling_opts(
  cores = getOption("mc.cores", 1L),
  warmup = 250,
  samples = 2000,
  chains = 4,
  control = list(),
  save_warmup = FALSE,
```

```
  seed = as.integer(runif(1, 1, 1e+08)),
  future = FALSE,
  max_execution_time = Inf,
  backend = c("rstan", "cmdstanr"),
  ...
)
```

Arguments

cores	Number of cores to use when executing the chains in parallel, which defaults to 1 but it is recommended to set the mc.cores option to be as many processors as the hardware and RAM allow (up to the number of chains).
warmup	Numeric, defaults to 250. Number of warmup samples per chain.
samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
chains	Numeric, defaults to 4. Number of MCMC chains to use.
control	List, defaults to empty. control parameters to pass to underlying `rstan` function. By default `adapt_delta = 0.95` and `max_treedepth = 15` though these settings can be overwritten.
save_warmup	Logical, defaults to FALSE. Should warmup progress be saved.
seed	Numeric, defaults uniform random number between 1 and 1e8. Seed of sampling process.
future	Logical, defaults to `FALSE`. Should stan chains be run in parallel using `future`. This allows users to have chains fail gracefully (i.e when combined with `max_execution_time`). Should be combined with a call to `future::plan()`.
max_execution_time	Numeric, defaults to Inf (seconds). If set wil kill off processing of each chain if not finished within the specified timeout. When more than 2 chains finish successfully estimates will still be returned. If less than 2 chains return within the allowed time then estimation will fail with an informative error.
backend	Character string indicating the backend to use for fitting stan models. Supported arguments are "rstan" (default) or "cmdstanr".
...	Additional parameters to pass to `rstan::sampling()`.

Value

A list of arguments to pass to `rstan::sampling()` or [cmdstanr::sample().

Examples

```
stan_sampling_opts(samples = 2000)
```

stan_vb_opts *Stan Variational Bayes Options*

Description

[Stable] Defines a list specifying the arguments passed to `rstan::vb()` or `cmdstanr::variational()`. Custom settings can be supplied which override the defaults.

Usage

```
stan_vb_opts(samples = 2000, trials = 10, iter = 10000, ...)
```

Arguments

`samples`	Numeric, default 2000. Overall number of approximate posterior samples.
`trials`	Numeric, defaults to 10. Number of attempts to use rstan::vb()] before failing.
`iter`	Numeric, defaulting to 10000. Number of iterations to use in `rstan::vb()`.
`...`	Additional parameters to pass to `rstan::vb()` or `cmdstanr::variational()`, depending on the chosen backend.

Value

A list of arguments to pass to `rstan::vb()` or `cmdstanr::variational()`, depending on the chosen backend.

Examples

```
stan_vb_opts(samples = 1000)
```

summary.epinow *Summary output from epinow*

Description

[Stable] `summary` method for class "epinow".

Usage

```
## S3 method for class 'epinow'
summary(
  object,
  output = c("estimates", "forecast", "estimated_reported_cases"),
  date = NULL,
  params = NULL,
  ...
)
```

Arguments

object	A list of output as produced by "epinow".
output	A character string of output to summarise. Defaults to "estimates" but also supports "forecast", and "estimated_reported_cases".
date	A date in the form "yyyy-mm-dd" to inspect estimates for.
params	A character vector of parameters to filter for.
...	Pass additional summary arguments to lower level methods

Value

Returns a <data.frame> of summary output

See Also

summary.estimate_infections epinow

---

summary.estimate_infections

Summary output from estimate_infections

Description

[Stable] summary method for class "estimate_infections".

Usage

```
## S3 method for class 'estimate_infections'
summary(
  object,
  type = c("snapshot", "parameters", "samples"),
  date = NULL,
  params = NULL,
  ...
)
```

Arguments

object	A list of output as produced by "estimate_infections".
type	A character vector of data types to return. Defaults to "snapshot" but also supports "parameters", and "samples". "snapshot" return a summary at a given date (by default the latest date informed by data). "parameters" returns summarised parameter estimates that can be further filtered using params to show just the parameters of interest and date. "samples" similarly returns posterior samples.
date	A date in the form "yyyy-mm-dd" to inspect estimates for.
params	A character vector of parameters to filter for.
...	Pass additional arguments to report_summary

Value

Returns a <data.frame> of summary output

See Also

summary estimate_infections report_summary

`trunc_opts` *Truncation Distribution Options*

Description

[Stable] Returns a truncation distribution formatted for usage by downstream functions. See `estimate_truncation()` for an approach to estimate these distributions.

Usage

```
trunc_opts(dist = Fixed(0), tolerance = 0.001, weight_prior = FALSE)
```

Arguments

`dist`	A delay distribution or series of delay distributions reflecting the truncation. It can be specified using the probability distributions interface in `EpiNow2` (See `?EpiNow2::Distributions`) or estimated using `estimate_truncation()`, which returns a `dist` object, suited for use here out-of-box. Default is a fixed distribution with maximum 0, i.e. no truncation.
`tolerance`	Numeric; the desired tolerance level.
`weight_prior`	Logical; if TRUE, the truncation prior will be weighted by the number of observation data points, in doing so approximately placing an independent prior at each time step and usually preventing the posteriors from shifting. If FALSE (default), no weight will be applied, i.e. the truncation distribution will be treated as a single parameter.

Value

A <trunc_opts> object summarising the input truncation distribution.

See Also

`convert_to_logmean()` `convert_to_logsd()` `bootstrapped_dist_fit()` `dist_spec()`

Examples

```
# no truncation
trunc_opts()

# truncation dist
trunc_opts(dist = LogNormal(mean = 3, sd = 2, max = 10))
```

update_secondary_args *Update estimate_secondary default priors*

Description

[Stable] This functions allows the user to more easily specify data driven or model based priors for `estimate_secondary()` from example from previous model fits using a <data.frame> to overwrite other default settings. Note that default settings are still required.

Usage

```
update_secondary_args(data, priors, verbose = TRUE)
```

Arguments

data	A list of data and arguments as returned by `create_stan_data()`.
priors	A <data.frame> of named priors to be used in model fitting rather than the defaults supplied from other arguments. This is typically useful if wanting to inform a estimate from the posterior of another model fit. Priors that are currently use to update the defaults are the scaling fraction ("frac_obs"), and delay parameters ("delay_params"). The <data.frame> should have the following variables: `variable`, `mean`, and `sd`.
verbose	Logical, defaults to `FALSE`. Should verbose progress messages be returned.

Value

A list as produced by `create_stan_data()`.

Examples

```
priors <- data.frame(variable = "frac_obs", mean = 3, sd = 1)
data <- list(obs_scale_mean = 4, obs_scale_sd = 3)
update_secondary_args(data, priors)
```

Index