

Package ‘epidist’

October 13, 2026

Title Estimate Epidemiological Delay Distributions with 'brms'

Version 1.0.0

Description Understanding and accurately estimating epidemiological delay distributions is important for public health policy. These estimates influence epidemic situational awareness, control strategies, and resource allocation. This package provides methods to address the key challenges in estimating these distributions, including truncation, interval censoring, and dynamical biases, following Park et al. (2024) <[doi:10.1101/2024.01.12.24301247](https://doi.org/10.1101/2024.01.12.24301247)> and Charniga et al. (2024) <[doi:10.1371/journal.pcbi.1012520](https://doi.org/10.1371/journal.pcbi.1012520)>. These issues are frequently overlooked, resulting in biased conclusions. Built on top of 'brms', it allows for flexible modelling including time-varying and spatial effects and partially pooled estimates of demographic characteristics.

License MIT + file LICENSE

URL <https://epidist.epinowcast.org/>,
<https://github.com/epinowcast/epidist/>

BugReports <https://github.com/epinowcast/epidist/issues/>

Depends R (>= 4.1.0)

Imports brms (>= 2.23.0), checkmate, cli, dplyr, lubridate,
primarycensored (>= 1.6.0), purrr, rstan (>= 2.26.0), stats,
tibble, tidyr, utils

Suggests MASS, bayesplot, bookdown, collapse, distspec (>= 0.2.0),
flexsurv, ggplot2, knitr, loo, marginaleffects, patchwork,
posterior, priorsense, reformulas, roxyglobals, scales,
spelling, testthat (>= 3.0.0), tidybayes, withr

Config/Needs/website r-lib/pkgdown, epinowcast/enwtheme

Config/testthat/edition 3

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

LazyData true

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Adam Howes [aut] (ORCID: <<https://orcid.org/0000-0003-2386-4031>>),
 Sang Woo Park [aut] (ORCID: <<https://orcid.org/0000-0003-2202-3361>>),
 Sam Abbott [aut, cre, cph] (ORCID:
 <<https://orcid.org/0000-0001-8057-8037>>),
 Sebastian Funk [ctb] (ORCID: <<https://orcid.org/0000-0002-2842-3406>>)

Maintainer Sam Abbott <contact@samabbott.co.uk>

Repository CRAN

Date/Publication 2026-10-13 14:40:02 UTC

Contents

add_summaries	5
assert_epidist	6
assert_epidist.epidist_aggregate_data	7
assert_epidist.epidist_estimates_data	8
assert_epidist.epidist_linelist_data	8
assert_epidist.epidist_meta_model	9
assert_epidist.epidist_multivariate	10
as_dist_spec.epidist_fit	10
as_epidist_aggregate_data	12
as_epidist_aggregate_data.data.frame	13
as_epidist_aggregate_data.default	14
as_epidist_aggregate_data.epidist_linelist_data	15
as_epidist_estimates_data	16
as_epidist_estimates_data.data.frame	17
as_epidist_estimates_data.epidist_estimates_data	23
as_epidist_estimates_data.epidist_multivariate	24
as_epidist_estimates_data.list	26
as_epidist_latent_model	27
as_epidist_latent_model.epidist_aggregate_data	28
as_epidist_latent_model.epidist_linelist_data	29
as_epidist_linelist_data	30
as_epidist_linelist_data.data.frame	31
as_epidist_linelist_data.default	32
as_epidist_linelist_data.epidist_aggregate_data	33
as_epidist_marginal_model	34
as_epidist_marginal_model.epidist_aggregate_data	35
as_epidist_marginal_model.epidist_linelist_data	36
as_epidist_meta_model	38
as_epidist_meta_model.epidist_aggregate_data	39
as_epidist_meta_model.epidist_estimates_data	41
as_epidist_meta_model.epidist_linelist_data	42

as_epidist_meta_model.NULL	44
as_epidist_multivariate	45
as_epidist_multivariate.data.frame	46
as_epidist_multivariate.matrix	47
as_epidist_naive_model	47
as_epidist_naive_model.epidist_aggregate_data	48
as_epidist_naive_model.epidist_linelist_data	49
delay_parameter_draws	50
delay_summary_draws	51
epidist	53
epidist_data	54
epidist_delay_draws	56
epidist_diagnostics	58
epidist_estimates_epireview	59
epidist_estimates_parameters	62
epidist_estimates_summaries	64
epidist_family	66
epidist_family_model	67
epidist_family_model.default	67
epidist_family_model.epidist_latent_model	68
epidist_family_model.epidist_marginal_model	69
epidist_family_model.epidist_meta_model	69
epidist_family_param.default	70
epidist_family_param.gengamma	71
epidist_family_prior	72
epidist_family_prior.default	73
epidist_family_prior.gengamma	74
epidist_family_prior.lognormal	75
epidist_formula	76
epidist_formula_model	77
epidist_formula_model.default	77
epidist_formula_model.epidist_latent_model	78
epidist_formula_model.epidist_marginal_model	79
epidist_formula_model.epidist_meta_model	80
epidist_formula_model.epidist_naive_model	81
epidist_gen_log_lik	81
epidist_gen_meta_log_lik	82
epidist_gen_meta_predict	83
epidist_gen_posterior_epred	84
epidist_gen_posterior_predict	85
epidist_meta_leave_one_out	86
epidist_model_prior	88
epidist_model_prior.default	89
epidist_model_prior.epidist_latent_model	89
epidist_model_prior.epidist_meta_model	90
epidist_newdata	91
epidist_newdata.default	93
epidist_newdata.epidist_latent_model	93

epidist_newdata.epidist_marginal_model	95
epidist_newdata.epidist_meta_model	96
epidist_newdata.epidist_naive_model	98
epidist_prior	99
epidist_stancode	101
epidist_stancode.default	102
epidist_stancode.epidist_naive_model	102
epidist_strata	103
epidist_transform_data_model	104
epidist_transform_data_model.default	105
epidist_transform_data_model.epidist_marginal_model	105
epidist_transform_data_model.epidist_meta_model	106
epidist_transform_data_model.epidist_naive_model	107
gengamma	108
is_epidist_aggregate_data	109
is_epidist_data	109
is_epidist_estimates_data	110
is_epidist_latent_model	111
is_epidist_linelist_data	111
is_epidist_marginal_model	112
is_epidist_meta_model	112
is_epidist_multivariate	113
is_epidist_naive_model	114
new_epidist_aggregate_data	114
new_epidist_estimates_data	115
new_epidist_latent_model	116
new_epidist_linelist_data	116
new_epidist_marginal_model	117
new_epidist_meta_model	118
new_epidist_multivariate	118
new_epidist_naive_model	120
plot.epidist_delay_draws	120
plot_delays	122
plot_delays.default	124
plot_delays.epidist_fit	124
plot_delays.epidist_linelist_data	126
plot_delays.list	127
plot_events	128
print.epidist_multivariate	129
sierra_leone Ebola_data	130
simulate_dates	130
simulate_exponential_cases	132
simulate_gillespie	133
simulate_secondary	134
simulate_study	134
simulate_uniform_cases	137
vcov.epidist_multivariate	138

add_summaries	<i>Add natural scale summaries of the delay distribution</i>
---------------	--

Description

Adds the mean and standard deviation of the delay distribution implied by each draw of the distributional parameters, and quantiles of that distribution if `probs` is given. Analytic solutions are used for the families that have one. The exponential family is summarised by simulating delays from it.

Usage

```
add_summaries(
  data,
  family = NULL,
  probs = NULL,
  method = c("auto", "analytic", "sample"),
  nsim = 1000
)
```

Arguments

<code>data</code>	A <code>data.frame</code> of draws of the distributional parameters, as returned by <code>delay_parameter_draws()</code> .
<code>family</code>	A model fit with <code>epidist()</code> , a brms family, an epidist family such as <code>gengamma()</code> , or the name of one, giving the delay distribution. If <code>NULL</code> , the default, the family is taken from <code>data</code> , which <code>delay_parameter_draws()</code> records on it. Some dplyr verbs drop that record, so pass the fit or the family if <code>data</code> has been through one of them.
<code>probs</code>	A numeric vector of probabilities to add quantiles of the delay distribution for. If <code>NULL</code> , the default, no quantiles are added.
<code>method</code>	Either "auto", the default, which uses the analytic solution when there is one and simulates otherwise, "analytic", which errors when there is no analytic solution, or "sample", which always simulates.
<code>nsim</code>	The number of delays to simulate per row of data. Defaults to 1000. Only used when simulating.

Details

The summaries describe the delay distribution, not the posterior. A row of data holds one draw of the distributional parameters, and the columns added are the mean, standard deviation and quantiles of the delay distribution those parameters define. Summarise the resulting columns across draws to get posterior summaries of them.

Quantile columns are named as in `posterior::quantile2()`, so `probs = 0.05` gives a `q5` column.

Simulation adds Monte Carlo error to the summaries. The standard error of the mean is the delay standard deviation divided by the square root of `nsim`. Simulation is also memory hungry, because it draws `nsim` delays for every row of data. Reduce the number of rows with `epidist_strata()`, or the number of draws with the `ndraws` argument of `delay_parameter_draws()`, if it is slow.

Value

The input with mean and sd columns added, and one column per element of probs. It has the `epidist_delay_draws` class, which records the family used and gives it a `plot()` method.

See Also

Other postprocess: `as_dist_spec.epidist_fit()`, `delay_parameter_draws()`, `delay_summary_draws()`, `epidist_delay_draws`, `epidist_strata()`

Examples

```
draws <- data.frame(mu = c(1.8, 2.0), sigma = c(0.5, 0.4))
add_summaries(draws, family = "lognormal", probs = c(0.05, 0.95))
```

 assert_epidist

Validation for epidist objects

Description

Validation for epidist objects

Usage

```
assert_epidist(data, ...)
```

Arguments

<code>data</code>	Object to validate
<code>...</code>	Additional arguments

Value

NULL invisibly

`assert_epidist.epidist_aggregate_data`*Assert validity of epidist_aggregate_data objects*

Description

Assert validity of epidist_aggregate_data objects

Usage

```
## S3 method for class 'epidist_aggregate_data'  
assert_epidist(data, ...)
```

Arguments

<code>data</code>	An object to check
<code>...</code>	Additional arguments

Value

NULL, invisibly. Called for the side effect of validating data.

See Also

Other aggregate_data: [as_epidist_aggregate_data\(\)](#), [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.default\(\)](#), [as_epidist_aggregate_data.epidist_linelist_data\(\)](#), [is_epidist_aggregate_data\(\)](#), [new_epidist_aggregate_data\(\)](#)

Examples

```
sierra_leone_ebola_data |>  
  as_epidist_linelist_data(  
    pdate_lwr = "date_of_symptom_onset",  
    sdate_lwr = "date_of_sample_tested"  
  ) |>  
  as_epidist_aggregate_data() |>  
  assert_epidist()
```

```
assert_epidist.epidist_estimates_data
```

Assert validity of epidist_estimates_data objects

Description

Assert validity of epidist_estimates_data objects

Usage

```
## S3 method for class 'epidist_estimates_data'
assert_epidist(data, ...)
```

Arguments

data	An object to check for validity.
...	Additional arguments

Value

NULL, invisibly. Called for the side effect of validating data.

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [as_epidist_estimates_data.list\(\)](#), [epidist_estimates_epireview\(\)](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#), [new_epidist_estimates_data\(\)](#)

```
assert_epidist.epidist_linelist_data
```

Assert validity of epidist_linelist_data objects

Description

Assert validity of epidist_linelist_data objects

Usage

```
## S3 method for class 'epidist_linelist_data'
assert_epidist(data, ...)
```

Arguments

data	An object to check for validity.
...	Additional arguments

Value

NULL, invisibly. Called for the side effect of validating data.

See Also

Other linelist_data: [as_epidist_linelist_data\(\)](#), [as_epidist_linelist_data.data.frame\(\)](#), [as_epidist_linelist_data.default\(\)](#), [as_epidist_linelist_data.epidist_aggregate_data\(\)](#), [is_epidist_linelist_data\(\)](#), [new_epidist_linelist_data\(\)](#)

```
assert_epidist.epidist_meta_model
```

Assert validity of epidist_meta_model objects

Description

Assert validity of epidist_meta_model objects

Usage

```
## S3 method for class 'epidist_meta_model'
assert_epidist(data, ...)
```

Arguments

data	An object to check for validity.
...	Additional arguments

Value

NULL, invisibly. Called for the side effect of validating data.

See Also

Other meta_model: [as_epidist_meta_model\(\)](#), [as_epidist_meta_model.NULL\(\)](#), [as_epidist_meta_model.epidist_a](#), [as_epidist_meta_model.epidist_estimates_data\(\)](#), [as_epidist_meta_model.epidist_linelist_data\(\)](#), [epidist_family_model.epidist_meta_model\(\)](#), [epidist_formula_model.epidist_meta_model\(\)](#), [epidist_meta_leave_one_out\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_newdata.epidist_meta_r](#), [epidist_transform_data_model.epidist_meta_model\(\)](#), [is_epidist_meta_model\(\)](#), [new_epidist_meta_model\(\)](#)

```
assert_epidist.epidist_multivariate
```

Assert validity of epidist_multivariate objects

Description

Assert validity of epidist_multivariate objects

Usage

```
## S3 method for class 'epidist_multivariate'
assert_epidist(data, ...)
```

Arguments

data	An object to check for validity.
...	Additional arguments.

Value

NULL, invisibly. Called for the side effect of validating data.

See Also

Other multivariate: [as_epidist_multivariate\(\)](#), [as_epidist_multivariate.data.frame\(\)](#), [as_epidist_multivariate.matrix\(\)](#), [is_epidist_multivariate\(\)](#), [new_epidist_multivariate\(\)](#), [print.epidist_multivariate\(\)](#), [vcov.epidist_multivariate\(\)](#)

```
as_dist_spec.epidist_fit
```

Export a fitted delay distribution as a distspec distribution

Description

Summarises the posterior of a fitted delay distribution into an uncertain <dist_spec> from the distspec package, for use in packages that take their delay distributions in that form. The natural parameters of the delay distribution are computed for each posterior draw and summarised into a prior on each of them.

Usage

```
## S3 method for class 'epidist_fit'
as_dist_spec(x, newdata = NULL, max = Inf, cdf_max = 1, ...)
```

Arguments

x	A model fit with <code>epidist()</code> .
newdata	A data.frame of data to predict for, with one row per delay distribution wanted. If NULL, the default, <code>epidist_newdata()</code> builds one row per unique combination of the predictors, with no censoring and no truncation. See the details.
max	The maximum of the delay distribution, passed to <code>distspec::bound_dist()</code> . Defaults to Inf, which is no maximum.
cdf_max	The cumulative probability to keep the delay distribution up to, passed to <code>distspec::bound_dist()</code> . Defaults to 1, which keeps the whole distribution.
...	Additional arguments passed to <code>brms::prepare_predictions()</code> , such as <code>ndraws</code> or <code>re_formula</code> .

Details

The lognormal, gamma and Weibull families are supported, as those are the delay distributions `distspec` has. Their `brms` parameters are mapped to the natural parameters of the matching `distspec` constructor for every draw before summarising:

- lognormal: `mu` is `meanlog` and `sigma` is `sdlog` of `distspec::LogNormal()`.
- gamma: `shape` is `shape` and `shape / mu` is rate of `distspec::Gamma()`.
- Weibull: `shape` is `shape` and `mu / gamma(1 + 1 / shape)` is scale of `distspec::Weibull()`.

Open an issue at <https://github.com/epiforecasts/distspec/issues> to ask `distspec` for another distribution.

Each natural parameter gets a `distspec::Normal()` prior with the mean and standard deviation of its marginal posterior. The posterior correlation between the parameters is not represented, so sampling from the result gives a wider range of delay distributions than the posterior does.

The default `newdata` is built with `epidist_newdata()` by expanding the variables in the model formula into a grid, so it has one row per unique combination of the predictors, and gives the delay distribution with no censoring and no truncation. A model with only an intercept gets a single row. A continuous predictor gets a row per distinct value, so pass `newdata` for such a model.

Value

A `<dist_spec>` when `newdata` has one row. A named list of them, one per row of `newdata`, when it has several. The names give the values of the columns of `newdata` that differ between rows, such as `"sex=0"`, and are the row numbers when no column differs.

See Also

`delay_parameter_draws()` for the draws this summarises and `epidist_newdata()` to build `newdata`.

Other postprocess: `add_summaries()`, `delay_parameter_draws()`, `delay_summary_draws()`, `epidist_delay_draws`, `epidist_strata()`

Examples

```

if (requireNamespace("distspec", quietly = TRUE)) {
  fit <- sierra_leone_ebola_data |>
    as_epidist_linelist_data(
      pdate_lwr = "date_of_symptom_onset",
      sdate_lwr = "date_of_sample_tested"
    ) |>
    as_epidist_aggregate_data() |>
    as_epidist_marginal_model() |>
    epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

  dist <- distspec::as_dist_spec(fit)
  dist

  # The delay distribution at the posterior mean of its parameters
  distspec::fix_parameters(dist, strategy = "mean")

  # Bound the delay distribution at 60 days for a package that takes bounds
  distspec::as_dist_spec(fit, max = 60)
}

```

as_epidist_aggregate_data

Create an epidist_aggregate_data object

Description

Creates an epidist_aggregate_data object from various input formats. This is useful when working with pre-aggregated data where each row represents multiple identical observations. See the specific methods for details on supported input formats and usage examples.

Usage

```
as_epidist_aggregate_data(data, ...)
```

Arguments

data	The data to convert
...	Additional arguments passed to methods

Value

An object of class epidist_aggregate_data.

See Also

Other aggregate_data: [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.default\(\)](#), [as_epidist_aggregate_data.epidist_linelist_data\(\)](#), [assert_epidist.epidist_aggregate_data\(\)](#), [is_epidist_aggregate_data\(\)](#), [new_epidist_aggregate_data\(\)](#)

```
as_epidist_aggregate_data.data.frame
```

Create an epidist_aggregate_data object from a data.frame

Description

This method takes a data.frame containing event dates (primary/secondary event dates and observation date) along with counts and creates an epidist_aggregate_data object. This format is useful when working with pre-aggregated data where each row represents multiple identical observations with the count stored in a specified column. Internally it makes use of [as_epidist_linelist_data.data.frame\(\)](#) to convert the data to a linelist format before adding the count column. See the other methods for other data input options.

Usage

```
## S3 method for class 'data.frame'
as_epidist_aggregate_data(
  data,
  n = NULL,
  pdate_lwr = NULL,
  sdate_lwr = NULL,
  pdate_upr = NULL,
  sdate_upr = NULL,
  obs_date = NULL,
  ...
)
```

Arguments

data	A data.frame containing line list data
n	A character string giving the name of the column containing the counts for each row. If NULL then the column n must be present in the data.
pdate_lwr	A string giving the column of data containing the primary event lower bound as a datetime. Defaults to NULL which assumes that the variable pdate_lwr is present.
sdate_lwr	A string giving the column of data containing the secondary event lower bound as a datetime. Defaults to NULL which assumes that the variable sdate_lwr is present.
pdate_upr	A string giving the column of data containing the primary event upper bound as a datetime. If this column exists in the data it will be used, otherwise if not supplied then the value of pdate_lwr + 1 day is used.

sdate_upr	A string giving the column of data containing the secondary event upper bound as a datetime. If this column exists in the data it will be used, otherwise if not supplied then the value of sdate_lwr + 1 day is used.
obs_date	A string giving the column of data containing the observation time as a datetime. Optional, if not supplied then the maximum of sdate_upr is used.
...	Additional arguments passed to methods

Value

An object of class `epidist_aggregate_data`.

See Also

Other `aggregate_data`: `as_epidist_aggregate_data()`, `as_epidist_aggregate_data.default()`, `as_epidist_aggregate_data.epidist_linelist_data()`, `assert_epidist.epidist_aggregate_data()`, `is_epidist_aggregate_data()`, `new_epidist_aggregate_data()`

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  )
```

```
as_epidist_aggregate_data.default
```

Create an `epidist_aggregate_data` object from vectors of event times

Description

This method takes vectors of event times (primary/secondary event times and observation time) along with counts and creates an `epidist_aggregate_data` object. This format is useful when working with pre-aggregated data where each row represents multiple identical observations with the count stored in the `n` column. Internally it makes use of `as_epidist_linelist_data.default()` to convert the data to a linelist format before adding the count column. See the other methods for other data input options.

Usage

```
## Default S3 method:
as_epidist_aggregate_data(
  data,
  n = NULL,
  ptime_upr = NULL,
```

```

    stime_lwr = NULL,
    stime_upr = NULL,
    obs_time = NULL,
    ...
)

```

Arguments

data	Numeric vector giving lower bounds for primary times.
n	An integerish vector containing the counts for each row. Must be the same length as the input data vector.
ptime_upr	Numeric vector giving upper bounds for primary times.
stime_lwr, stime_upr	Numeric vectors giving lower and upper bounds for secondary times.
obs_time	Numeric vector giving observation times.
...	Additional columns to add to the epidist_linelist_data object.

Value

An object of class `epidist_aggregate_data`.

See Also

Other `aggregate_data`: [as_epidist_aggregate_data\(\)](#), [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.epidist_linelist_data\(\)](#), [assert_epidist.epidist_aggregate_data\(\)](#), [is_epidist_aggregate_data\(\)](#), [new_epidist_aggregate_data\(\)](#)

Examples

```

as_epidist_aggregate_data(
  data = c(1, 2, 3),
  ptime_upr = c(2, 3, 4),
  stime_lwr = c(3, 4, 5),
  stime_upr = c(4, 5, 6),
  obs_time = c(5, 6, 7),
  n = c(1, 2, 3)
)

```

```
as_epidist_aggregate_data.epidist_linelist_data
```

Convert linelist data to aggregate format

Description

This method takes an `epidist_linelist_data` object (see [as_epidist_linelist_data\(\)](#)) and aggregates it by counting unique combinations of the required time variables (primary/secondary event times and observation time) and any additional variables specified in `by`. The result is a more compact representation of the same data where each row represents multiple identical observations with the count stored in the `n` column.

Usage

```
## S3 method for class 'epidist_linelist_data'  
as_epidist_aggregate_data(data, by = NULL, ...)
```

Arguments

data	The data to convert
by	Character vector of additional variables to stratify by, beyond the required time variables.
...	Additional arguments passed to methods

Value

An object of class `epidist_aggregate_data`.

See Also

Other `aggregate_data`: [as_epidist_aggregate_data\(\)](#), [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.default\(\)](#), [assert_epidist.epidist_aggregate_data\(\)](#), [is_epidist_aggregate_data\(\)](#), [new_epidist_aggregate_data\(\)](#)

Examples

```
# Default stratification by required time variables only  
sierra_leone_ebola_data |>  
  as_epidist_linelist_data(  
    pdate_lwr = "date_of_symptom_onset",  
    sdate_lwr = "date_of_sample_tested"  
  ) |>  
  as_epidist_aggregate_data()  
  
# Additional stratification by other variables  
sierra_leone_ebola_data |>  
  as_epidist_linelist_data(  
    pdate_lwr = "date_of_symptom_onset",  
    sdate_lwr = "date_of_sample_tested"  
  ) |>  
  as_epidist_aggregate_data(by = "age")
```

`as_epidist_estimates_data`

Create an `epidist_estimates_data` object

Description

Creates an `epidist_estimates_data` object from published summary estimates of a delay distribution. Each row is a single reported summary component (a mean, a standard deviation, or a quantile) together with the metadata needed to work out what the study that reported it was actually estimating. These objects are the summary data input to `as_epidist_meta_model()`. See the specific methods for details on supported input formats and usage examples.

Usage

```
as_epidist_estimates_data(data, ...)
```

Arguments

<code>data</code>	The data to convert
<code>...</code>	Additional arguments passed to methods

Details

The meta model these objects feed is experimental. Its interface may still change in future releases.

Value

An object of class `epidist_estimates_data`.

See Also

Other `estimates_data`: `as_epidist_estimates_data.data.frame()`, `as_epidist_estimates_data.epidist_estimates_data.epidist_multivariate()`, `as_epidist_estimates_data.list()`, `assert_epidist.epidist_estimates_data()`, `epidist_estimates_epireview()`, `epidist_estimates_parameters()`, `epidist_estimates_summaries()`, `is_epidist_estimates_data()`, `new_epidist_estimates_data()`

```
as_epidist_estimates_data.data.frame
```

Create an epidist_estimates_data object from a data frame

Description

This method takes a `data.frame` of published summary estimates and creates an `epidist_estimates_data` object. Column names may either match the names used below or be supplied via the corresponding argument.

Usage

```
## S3 method for class 'data.frame'
as_epidist_estimates_data(
  data,
  study = NULL,
  type = NULL,
  value = NULL,
  se = NULL,
  n = NULL,
  p = NULL,
  pwindow = NULL,
  swindow = NULL,
  relative_obs_time = NULL,
  trunc_adjusted = NULL,
  trunc_design = NULL,
  cens_adjusted = NULL,
  delay_min = NULL,
  growth_rate = NULL,
  growth_rate_sd = NULL,
  max_delay = NULL,
  advise = TRUE,
  ...
)
```

Arguments

data	A data.frame of published summary estimates.
study	A string giving the column of data containing the study identifier. Defaults to NULL which assumes the variable study is present.
type	A string giving the column of data containing the summary type. Each entry must be one of "mean", "sd", or "quantile". Defaults to NULL which assumes the variable type is present.
value	A string giving the column of data containing the reported value of the summary. Defaults to NULL which assumes the variable value is present.
se	A string giving the column of data containing the reported standard error of the summary. Optional. When supplied it overrides the standard error implied by the sample size, and takes the row out of the joint likelihood as_epidist_meta_model() otherwise uses for summaries a study computed from the same delays. It is always on the scale of the reported value, so for a "quantile" row it is a standard error for the reported delay.
n	A string giving the column of data containing the number of delays the summary was computed from. Required unless se is supplied.
p	A string giving the column of data containing the probability of a reported quantile. Required for rows with type of "quantile" and ignored otherwise.
pwindow, swindow	Strings giving the columns of data containing the primary and secondary event censoring window widths. Default to 1 (daily reporting) when not supplied. A

fully adjusted study (cens_adjusted code 1) does not use them, because its estimand is the continuous delay distribution itself, so its rows may leave them NA. Every other code needs them.

relative_obs_time	A string giving the column of data containing the observation time relative to the primary event, that is the right truncation point on the delay scale for a cohort design, or the length of the collection window for an accrual design. Defaults to Inf, meaning no truncation.
trunc_adjusted	A string giving the column of data containing a logical flag for whether the study corrected for right truncation. Defaults to TRUE for a study with no finite relative_obs_time, which includes every study where no relative_obs_time column is supplied, and FALSE otherwise. A study assumed to have adjusted is warned about, because real time estimates are right truncated unless the study corrected for it and reviews rarely record which studies did. Supply the column to say so yourself.
trunc_design	A string giving the column of data containing how the study stopped collecting data, either "cohort" (it followed every primary event for the same relative_obs_time) or "accrual" (it collected over a window of that length and stopped at its calendar end). Defaults to "cohort", and is only used for studies that did not adjust for right truncation. The accrual weight on the grid of a study that did not adjust for censoring is exact for any relative_obs_time, pwindow and swindow. The weight used for the uniform single interval approximation is exact only for a narrow pwindow, and puts the implied mean about 3% high with a weekly primary window, a collection window of 28 days, a delay of mean 4.6 days and a growth rate of 0.2. See vignette("model").
cens_adjusted	A string giving the column of data containing the censoring adjustment code (0, 1, 2, 3, or 4, as described above). Defaults to 0.
delay_min	A string giving the column of data containing the smallest delay the study counted, its left truncation point, on the scale the study reported. Defaults to 0, meaning the study counted every delay, which for cens_adjusted code 4 includes any delay reported below zero. Must be below the grid cutoff, and no reported mean or quantile may fall below it.
growth_rate	A string giving the column of data containing the exponential growth rate of primary events during the study period. Defaults to 0. It plays two roles. Within each primary window it tilts the primary event towards the end of the window, which for a daily window is negligible. Under trunc_design = "accrual" it also weights the follow up available to each delay, which is the dynamical bias of a growing epidemic and can move the implied mean by a day or more. A non-zero rate is expensive, because the primary censored delay distribution then has no analytical solution and every evaluation becomes a numerical integration. Leave it at 0 unless the study accrued cases over a period of growth. A number is taken as known. NA means the rate is unknown, and the study then uses the pgrowth distributional parameter of the meta model, the same parameter that primary = "expgrowth" estimates from individual level data in as_epidist_marginal_model() . Give growth_rate_sd as well to treat a reported rate as an informative prior on that parameter rather than as a fixed number. See as_epidist_meta_model() and vignette("primary-events").

growth_rate_sd	A string giving the column of data containing the standard deviation of growth_rate, where the study reported the rate with uncertainty. Defaults to NA, meaning the rate is known exactly. A positive value makes the study estimate its rate as the pgrowth distributional parameter, with a normal prior centred on growth_rate with this standard deviation. That prior is set up by <code>as_epidist_meta_model()</code> , see there for the formula it uses. A value of 0 is the same as NA, and a value with an NA growth_rate is an error, because the prior has no centre.
max_delay	A string giving the column of data containing the delay beyond which the implied summaries are truncated when building the discrete grid. Only used when the study adjusted for right truncation. Defaults to the delay beyond which one percent of the second moment of a lognormal matched to the study's summaries lies, through its mean and standard deviation, or its median and largest quantile above the median where it reported only quantiles. The lognormal is used whatever family is fitted later, because the family is not known when the data are built and, for the same mean and standard deviation, its tail is heavier than the gamma's or the Weibull's, so the cutoff is far enough for those families and only longer than they need. That is the yardstick of the short cutoff check in the Checks section, so the default never trips it where it binds. It is rounded up to a whole number of secondary windows, with a minimum of ten and a maximum of twenty times the largest reported value, because for a heavy tail one percent of the second moment lies thousands of delays out, and is five times the largest reported value where nothing can be matched. Raise it for a delay with a longer tail than a lognormal, whose implied standard deviation is biased downwards if the distribution has not decayed by the cutoff, and lower it to fit faster. A message names the studies whose cutoff is too short, see the Checks section.
advise	Whether to run the advisory checks of the Checks section and message about the studies they flag. Defaults to TRUE. The list method sets it to FALSE for each element and runs the checks once on the combined object.
...	Not used in this method.

Value

An object of class `epidist_estimates_data`.

What we need from each study

Published delay estimates are almost never estimates of the true continuous delay distribution. To use them we forward model what the study's estimation procedure would converge to, which means we need to know how the study handled the biases we support, along with the data process it saw. `vignette("model")` derives each estimand, and `vignette("meta")` works through assembling this metadata for simulated and real studies. For each study we need:

- **How it adjusted for censoring** (`cens_adjusted`). A taxonomy of the common ways interval censored delays are summarised, deliberately limited to a few widely used approaches:
 - 0: no adjustment. The study took integer date differences (for example date of onset subtracted from date of report) and summarised them directly. This is the most common case in the literature.

- 1: fully adjusted. The study used a method targeting the underlying continuous distribution, such as a double interval censored likelihood.
- 2: uniform single interval approximation. The study adjusted the secondary interval only, assuming a uniform delay within it, and left the primary interval uncorrected.
- 3: midpoint imputation. The study assigned each delay to the centre of the interval it was observed in, which shifts every reported delay up by half a secondary window without changing its spread.
- 4: midpoint imputation with a uniform interval. The study placed the primary event at the midpoint of its window and integrated the secondary interval. Common where the primary event has a wide exposure window and the secondary date is recorded precisely. With a wide primary window the shortest delays are reported below zero, so the estimand puts mass there. Set `delay_min` if the study dropped them.

Use code 3 for a study that midpointed the secondary interval and left the primary alone. Anything more exotic must be approximated by whichever code is closest, and if you cannot tell which a study used, state the assumption you are making.

- **Whether it adjusted for right truncation** (`trunc_adjusted`) and, if it did not, **the observation time** (`relative_obs_time`) and **how collection stopped** (`trunc_design`). For a cohort the observation time is the truncation point on the delay scale. For an accrual design, where collection stopped at a calendar date, it is the length of the collection window, which is usually easier to read off a paper. Real time estimates are right truncated unless the study corrected for it.
- **The censoring windows** (`pwindow`, `swindow`). The width, in the same time units as the delay, of the interval each event was observed in. Daily reporting gives windows of 1, weekly reporting gives 7. A fully adjusted study does not use them and may leave them NA.
- **The sample size** (`n`), the number of delays the summary was computed from. This sets the sampling uncertainty on the reported value. A reported standard error (`se`) may be given instead, and takes precedence when supplied.
- **The minimum delay it counted** (`delay_min`), where the study dropped delays below some point. Its summaries then describe a left truncated delay distribution. Defaults to 0, meaning the study counted every delay.

Systematic reviews rarely record this metadata, so you must supply your own assumption and say so alongside any results. Where it is missing entirely, a covariate for the phase of the outbreak each estimate was made in is another option, since the `brms` formula makes it a meta-regression that estimates the residual bias rather than correcting it mechanically.

What shape a study reported its estimate in

This method takes a long table with one row per reported summary. Two other shapes are common enough to have their own entry points, each returning an object this one would.

- `epidist_estimates_summaries()` takes one study's summaries in wide form.
- `epidist_estimates_parameters()` takes the parameters of a distribution a study fitted, which studies often publish in place of summaries, and converts them to the summaries the fitted distribution implies. Reported parameters are not a type here, because the family a study fitted need not be the family being fitted to it.

A study that published draws of its parameters, rather than point summaries of them, can report the correlation between the quantities it reports. Pass the draws to `as_epidist_multivariate()` and the result to `as_epidist_estimates_data()`, which is the only route by which a covariance reaches `as_epidist_meta_model()`. Such rows need no `n` and no `se`.

Contributions from several studies combine by passing them in a list, which `as_epidist_estimates_data()` binds into one object.

Checks

Beyond validating its input, this method runs advisory checks on the summaries and messages about the studies they flag. They run once, when the object is built, and not again when it is passed to `as_epidist_meta_model()`. Each message names the studies concerned, and the row of the input where a single summary is meant, and a pointer here follows the messages of a call. Studies combined in a list are checked once, together, and `simulate_study()` leaves the checks to that combined object. Set `advise = FALSE` to skip them.

- **Assumed truncation adjustment.** Where no `trunc_adjusted` column is supplied, a study with no finite `relative_obs_time` is taken to have adjusted for right truncation and every other study not to have. Real time estimates are right truncated unless the study corrected for it, and reviews rarely record which did, so this is the assumption most likely to be wrong and is a warning rather than a message.
- **Short grid cutoff.** The implied summaries of a study that adjusted for right truncation but is evaluated on a grid, which is a study with `cens_adjusted` 0 or 3, or 2 or 4 with a non zero or estimated `growth_rate`, run to `max_delay`. A cutoff the delay distribution has not decayed by biases them downwards, and the standard deviation most, because the tail beyond the cutoff carries a share of the second moment out of all proportion to its mass. A lognormal is matched to what the study reported, through its mean and standard deviation, or its median and largest quantile above the median where it reported only quantiles, and the study is flagged when more than 2% of the second moment of that lognormal lies beyond the cutoff. That is where the standard deviation on the grid falls about 1% short, and the shortfall grows with the share. Studies reporting neither pair are not checked. Raise `max_delay` for the study.
- **Coarse quadrature.** The moments and distribution function of a continuous estimand truncated at the grid cutoff are computed by Simpson's rule on equally spaced intervals from `delay_min` to the cutoff, so the node spacing is set by the cutoff and not by the scale of the delay. The number of intervals is chosen per study so that the spacing is at most a quarter of the spread the study reported, with `options(epidist.meta_n_quad)` as its floor and 2000 as a cap the option lifts when set above it. A study whose cutoff is very long relative to its spread hits the cap and is left with nodes further apart than that, so its implied summaries may be inaccurate. This covers a study that did not adjust for right truncation and used a continuous adjustment (`cens_adjusted` 1, 2 or 4), a study that did adjust but whose primary events were not uniform within their window (`cens_adjusted` 2 or 4 with a non zero or estimated `growth_rate`), and the quantiles of a study reporting a covariance matrix, which are read off the same nodes. Raise the option above the cap before building the model data, or lower `max_delay`.
- **Coarse quantiles.** A study that summarised interval censored delays without adjusting for censoring (`cens_adjusted` 0 or 3) reports quantiles of a discrete distribution, which the model interpolates through the mid points of its cells. The reported value is still rounded to that grid, and what the interpolation leaves behind does not shrink with the study sample size. It is a

few percent once a reported quantile sits a few tens of cells above the smallest delay the study counted, and tens of percent when it sits within about ten. A study is flagged on its smallest reported quantile, the one nearest that edge of the grid, because the residual on that quantile is what biases the fitted spread even when the larger quantiles of the same study sit well up the grid. Check that swindow is the resolution the study worked at. A reported mean and standard deviation of the same delays do not carry this residual, so fit them in preference where the study gives them.

- **Heavy tailed standard deviation.** The sampling standard error of a reported standard deviation is $\sigma \sqrt{(\kappa - 1)/(4n)}$, with κ the kurtosis of the delays. The normal approximation behind it holds while that relative standard error is below about a quarter. Above it the sampling distribution of a sample standard deviation is far from normal, the asymptotic standard error overstates its spread by up to two times, and the joint likelihood of a mean and standard deviation pair is biased by about a standard error. The kurtosis is taken from the reported mean and standard deviation under a lognormal delay, which is a plausible tail for a delay of that coefficient of variation. Where the study reports quantiles inside the body of the distribution, those are safer to fit.

See Also

Other estimates_data: `as_epidist_estimates_data()`, `as_epidist_estimates_data.epidist_estimates_data()`, `as_epidist_estimates_data.epidist_multivariate()`, `as_epidist_estimates_data.list()`, `assert_epidist.epidist_estimates_data()`, `epidist_estimates_epireview()`, `epidist_estimates_parameters()`, `epidist_estimates_summaries()`, `is_epidist_estimates_data()`, `new_epidist_estimates_data()`

Examples

```
as_epidist_estimates_data(
  data.frame(
    study = c("A", "A", "B"),
    type = c("mean", "sd", "quantile"),
    value = c(7.5, 3.6, 11.2),
    p = c(NA, NA, 0.9),
    n = c(120, 120, 80),
    relative_obs_time = c(20, 20, Inf),
    trunc_adjusted = c(FALSE, FALSE, TRUE),
    cens_adjusted = c(0, 0, 1)
  )
)
```

```
as_epidist_estimates_data.epidist_estimates_data
```

Return an epidist_estimates_data object unchanged

Description

An object carrying the class has already been checked, and the `epidist_data` methods re-check it whenever it is modified, so nothing is done here.

Usage

```
## S3 method for class 'epidist_estimates_data'
as_epidist_estimates_data(data, ...)
```

Arguments

data	The data to convert
...	Not used in this method.

Value

The input, unchanged.

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [as_epidist_estimates_data.list\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview\(\)](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#), [new_epidist_estimates_data\(\)](#)

Examples

```
estimates <- epidist_estimates_summaries(
  "A",
  mean = 7.5, n = 120, trunc_adjusted = TRUE
)
identical(as_epidist_estimates_data(estimates), estimates)
```

```
as_epidist_estimates_data.epidist_multivariate
```

Create an epidist_estimates_data object from a multivariate representation

Description

This is the only route by which a covariance between a study's reported summaries reaches [as_epidist_meta_model\(\)](#). The covariance comes from draws of the study's parameters, so it is one the study could have computed, and the summaries it covers are fitted with a multivariate normal likelihood rather than as independent observations.

Usage

```
## S3 method for class 'epidist_multivariate'
as_epidist_estimates_data(
  data,
  study,
  family = NULL,
```

```

    moments = c("mean", "sd"),
    probs = numeric(0),
    mvn_id = NULL,
    advise = TRUE,
    ...
)

```

Arguments

<code>data</code>	The data to convert
<code>study</code>	A string naming the study the draws come from.
<code>family</code>	The distribution the study fitted, one of "lognormal", "gamma" or "weibull", where the draws hold its natural parameters. Defaults to NULL, meaning the draws already hold reported summaries.
<code>moments</code>	Which moments to report, any of "mean" and "sd". Only used where family is given.
<code>probs</code>	A numeric vector of probabilities to report quantiles at. Only used where family is given.
<code>mvn_id</code>	A string identifying the covariance matrix. Defaults to study, and only needs setting where one study contributes more than one multivariate object.
<code>advise</code>	Whether to run the advisory checks of the Checks section and message about the studies they flag. Defaults to TRUE. The list method sets it to FALSE for each element and runs the checks once on the combined object.
<code>...</code>	Study metadata, as documented in as_epidist_estimates_data.data.frame() .

Details

Where family is given, the draws hold the natural parameters of a distribution the study fitted. Each draw is pushed through to the summaries the fitted distribution implies, over the range of delays the study could have seen, and the covariance is taken over those. This is the exact version of the delta method [epidist_estimates_parameters\(\)](#) applies, and it needs no linearisation.

Where family is NULL, the parameters must already be quantities a study reports. Name them mean, sd, or q followed by a probability, such as `q0.25`.

Draws over more than one index point are not yet supported for fitting, because the linear predictor would have to vary within one likelihood observation.

Value

An object of class `epidist_estimates_data` with one row per reported summary and the covariance matrix attached.

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.list\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview\(\)](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#), [new_epidist_estimates_data\(\)](#)

Examples

```
set.seed(1)
draws <- cbind(mean = rnorm(500, 7.5, 0.3), sd = rnorm(500, 3.6, 0.2))
as_epidist_estimates_data(
  as_epidist_multivariate(draws),
  study = "site A",
  trunc_adjusted = TRUE,
  cens_adjusted = 1
)
```

```
as_epidist_estimates_data.list
```

Combine epidist_estimates_data objects from several studies

Description

Each element is coerced on its own and the results are bound into one object. Combining is associative, so contributions can be assembled in any order and in any grouping. The advisory checks of the Checks section of [as_epidist_estimates_data.data.frame\(\)](#) run once, on the combined object, so each message names every study it applies to.

Usage

```
## S3 method for class 'list'
as_epidist_estimates_data(data, advise = TRUE, ...)
```

Arguments

data	The data to convert
advise	Whether to run the advisory checks of the Checks section and message about the studies they flag. Defaults to TRUE. The list method sets it to FALSE for each element and runs the checks once on the combined object.
...	Passed to the method used for each element.

Value

An object of class `epidist_estimates_data` holding the rows of every element.

See Also

Other `estimates_data`: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview\(\)](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#), [new_epidist_estimates_data\(\)](#)

Examples

```
as_epidist_estimates_data(list(
  epidist_estimates_summaries(
    "A",
    mean = 7.5, sd = 3.6, n = 120, trunc_adjusted = TRUE
  ),
  epidist_estimates_summaries(
    "B",
    mean = 6.9, n = 80, relative_obs_time = 20, trunc_adjusted = FALSE
  )
))
```

as_epidist_latent_model

Convert an object to an epidist_latent_model object

Description

Creates an `epidist_latent_model` object from various input formats. This enables fitting latent variable models for epidemiological delays using `epidist()`, as described in Park et al. (2024) and Charniga et al. (2024) The latent model approach accounts for double interval censoring and right truncation in delay data.

Usage

```
as_epidist_latent_model(data, ...)
```

Arguments

<code>data</code>	An object to be converted to the class <code>epidist_latent_model</code>
<code>...</code>	Additional arguments passed to methods.

Value

An object of class `epidist_latent_model`.

References

- Park et al. (2024) [doi:10.1101/2024.01.12.24301247](https://doi.org/10.1101/2024.01.12.24301247)
- Charniga et al. (2024) [doi:10.1371/journal.pcbi.1012520](https://doi.org/10.1371/journal.pcbi.1012520)

See Also

Other latent_model: `as_epidist_latent_model.epidist_aggregate_data()`, `as_epidist_latent_model.epidist_li`, `epidist_family_model.epidist_latent_model()`, `epidist_formula_model.epidist_latent_model()`, `epidist_model_prior.epidist_latent_model()`, `epidist_newdata.epidist_latent_model()`, `is_epidist_latent_model()`, `new_epidist_latent_model()`

```
as_epidist_latent_model.epidist_aggregate_data
```

The latent model method for epidist_aggregate_data objects

Description

This method converts aggregate data to a latent model format by first converting it to linelist format using `as_epidist_linelist_data.epidist_aggregate_data()` and then passing it to `as_epidist_latent_model.epidist_linelist_data()`. This ensures that the counts in the aggregate data are properly expanded into individual observations before fitting the latent model.

Usage

```
## S3 method for class 'epidist_aggregate_data'
as_epidist_latent_model(data, primary = .primary_choices(), ...)
```

Arguments

<code>data</code>	An <code>epidist_aggregate_data</code> object
<code>primary</code>	The distribution of the primary event within its censoring window. "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the <code>pgrowth</code> distributional parameter.
<code>...</code>	Not used in this method.

Value

An object of class `epidist_latent_model`.

See Also

Other `latent_model`: `as_epidist_latent_model()`, `as_epidist_latent_model.epidist_linelist_data()`, `epidist_family_model.epidist_latent_model()`, `epidist_formula_model.epidist_latent_model()`, `epidist_model_prior.epidist_latent_model()`, `epidist_newdata.epidist_latent_model()`, `is_epidist_latent_model()`, `new_epidist_latent_model()`

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  ) |>
  as_epidist_latent_model()
```

as_epidist_latent_model.epidist_linelist_data

The latent model method for epidist_linelist_data objects

Description

This method takes an `epidist_linelist_data` object and converts it to a format suitable for fitting latent variable models. It calculates key variables needed for the latent variable method described in Park et al. (2024) and Charniga et al. (2024). This approach adjusts for double interval censoring and right truncation in the data.

Usage

```
## S3 method for class 'epidist_linelist_data'
as_epidist_latent_model(data, primary = .primary_choices(), ...)
```

Arguments

<code>data</code>	An <code>epidist_linelist_data</code> object containing individual-level observations with primary and secondary event times. See as_epidist_linelist_data() for details on creating this object.
<code>primary</code>	The distribution of the primary event within its censoring window. "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the <code>pgrowth</code> distributional parameter.
<code>...</code>	Not used in this method.

Value

An object of class `epidist_latent_model`.

References

- Park et al. (2024) [doi:10.1101/2024.01.12.24301247](#)
- Charniga et al. (2024) [doi:10.1371/journal.pcbi.1012520](#)

See Also

Other `latent_model`: [as_epidist_latent_model\(\)](#), [as_epidist_latent_model.epidist_aggregate_data\(\)](#), [epidist_family_model.epidist_latent_model\(\)](#), [epidist_formula_model.epidist_latent_model\(\)](#), [epidist_model_prior.epidist_latent_model\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [is_epidist_latent_model\(\)](#), [new_epidist_latent_model\(\)](#)

Examples

```
sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_latent_model()
```

as_epidist_linelist_data

Create an epidist_linelist_data object

Description

Creates an `epidist_linelist_data` object from various input formats. This is useful when working with individual-level data where each row represents a single observation with primary and secondary event times. See the specific methods for details on supported input formats and usage examples.

Usage

```
as_epidist_linelist_data(data, ...)
```

Arguments

<code>data</code>	The data to convert
<code>...</code>	Additional arguments passed to methods

Value

An object of class `epidist_linelist_data`.

See Also

Other `linelist_data`: [as_epidist_linelist_data.data.frame\(\)](#), [as_epidist_linelist_data.default\(\)](#), [as_epidist_linelist_data.epidist_aggregate_data\(\)](#), [assert_epidist.epidist_linelist_data\(\)](#), [is_epidist_linelist_data\(\)](#), [new_epidist_linelist_data\(\)](#)

```
as_epidist_linelist_data.data.frame
```

Create an epidist_linelist_data object from a data frame with event dates

Description

This method takes a data.frame containing event dates (primary/secondary event dates and observation date) and creates an epidist_linelist_data object. This format is useful when working with individual-level data where each row represents a single observation. Internally it converts dates to numeric times relative to the earliest primary event date and uses `as_epidist_linelist_data.default()` to create the final object. See the other methods for other data input options.

Usage

```
## S3 method for class 'data.frame'
as_epidist_linelist_data(
  data,
  pdate_lwr = NULL,
  sdate_lwr = NULL,
  pdate_upr = NULL,
  sdate_upr = NULL,
  obs_date = NULL,
  ...
)
```

Arguments

data	A data.frame containing line list data
pdate_lwr	A string giving the column of data containing the primary event lower bound as a datetime. Defaults to NULL which assumes that the variable pdate_lwr is present.
sdate_lwr	A string giving the column of data containing the secondary event lower bound as a datetime. Defaults to NULL which assumes that the variable sdate_lwr is present.
pdate_upr	A string giving the column of data containing the primary event upper bound as a datetime. If this column exists in the data it will be used, otherwise if not supplied then the value of pdate_lwr + 1 day is used.
sdate_upr	A string giving the column of data containing the secondary event upper bound as a datetime. If this column exists in the data it will be used, otherwise if not supplied then the value of sdate_lwr + 1 day is used.
obs_date	A string giving the column of data containing the observation time as a datetime. Optional, if not supplied then the maximum of sdate_upr is used.
...	Additional arguments passed to methods

Value

An object of class `epidist_linelist_data`.

See Also

Other `linelist_data`: `as_epidist_linelist_data()`, `as_epidist_linelist_data.default()`, `as_epidist_linelist_data.epidist_aggregate_data()`, `assert_epidist.epidist_linelist_data()`, `is_epidist_linelist_data()`, `new_epidist_linelist_data()`

Examples

```
sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  )
```

```
as_epidist_linelist_data.default
```

Create an `epidist_linelist_data` object from vectors of event times

Description

This method takes vectors of event times (primary/secondary event times and observation time) and creates an `epidist_linelist_data` object. This format is useful when working with individual-level data where each row represents a single observation. See the other methods for other data input options.

Usage

```
## Default S3 method:
as_epidist_linelist_data(
  data,
  ptime_upr = NULL,
  stime_lwr = NULL,
  stime_upr = NULL,
  obs_time = NULL,
  ...
)
```

Arguments

<code>data</code>	Numeric vector giving lower bounds for primary times.
<code>ptime_upr</code>	Numeric vector giving upper bounds for primary times.
<code>stime_lwr</code> , <code>stime_upr</code>	Numeric vectors giving lower and upper bounds for secondary times.
<code>obs_time</code>	Numeric vector giving observation times.
<code>...</code>	Additional columns to add to the <code>epidist_linelist_data</code> object.

Value

An object of class epidist_linelist_data.

See Also

Other linelist_data: [as_epidist_linelist_data\(\)](#), [as_epidist_linelist_data.data.frame\(\)](#), [as_epidist_linelist_data.epidist_aggregate_data\(\)](#), [assert_epidist.epidist_linelist_data\(\)](#), [is_epidist_linelist_data\(\)](#), [new_epidist_linelist_data\(\)](#)

Examples

```
as_epidist_linelist_data(
  data = c(1, 2, 3),
  ptime_upr = c(2, 3, 4),
  stime_lwr = c(3, 4, 5),
  stime_upr = c(4, 5, 6),
  obs_time = c(5, 6, 7)
)
```

```
as_epidist_linelist_data.epidist_aggregate_data
```

Convert aggregate data to linelist format

Description

This method expands an epidist_aggregate_data object into individual observations by uncounting the n column, then converts it to linelist format using [as_epidist_linelist_data.data.frame\(\)](#).

Usage

```
## S3 method for class 'epidist_aggregate_data'
as_epidist_linelist_data(data, ...)
```

Arguments

data	The data to convert
...	Additional arguments passed to methods

Value

An object of class epidist_linelist_data.

See Also

Other linelist_data: [as_epidist_linelist_data\(\)](#), [as_epidist_linelist_data.data.frame\(\)](#), [as_epidist_linelist_data.default\(\)](#), [assert_epidist.epidist_linelist_data\(\)](#), [is_epidist_linelist_data\(\)](#), [new_epidist_linelist_data\(\)](#)

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  ) |>
  as_epidist_linelist_data()
```

as_epidist_marginal_model

Convert an object to an epidist_marginal_model object

Description

Creates an `epidist_marginal_model` object from various input formats. This enables fitting marginal models for epidemiological delays using `epidist()`. The marginal model approach uses the likelihood from the `primarycensored` package to efficiently handle censoring in both primary and secondary events as well as truncation due to observation times. See the specific methods `as_epidist_marginal_model.epidist_linelist_data()` and `as_epidist_marginal_model.epidist_aggregate_data()` for details on supported input formats and usage examples.

Usage

```
as_epidist_marginal_model(data, ...)
```

Arguments

<code>data</code>	An object to be converted to the class <code>epidist_marginal_model</code>
<code>...</code>	Additional arguments passed to the methods. These set the primary event distribution and other fitting options; see the methods for the arguments they take.

Value

An object of class `epidist_marginal_model`.

See Also

Other `marginal_model`: `as_epidist_marginal_model.epidist_aggregate_data()`, `as_epidist_marginal_model.epidist_family_model.epidist_marginal_model()`, `epidist_formula_model.epidist_marginal_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`, `new_epidist_marginal_model()`

as_epidist_marginal_model.epidist_aggregate_data

The marginal model method for epidist_aggregate_data objects

Description

This method converts aggregate data to a marginal model format by passing it to `as_epidist_marginal_model.epidist_li` with the `n` column used as weights. This ensures that the likelihood is weighted by the counts in the aggregate data.

Usage

```
## S3 method for class 'epidist_aggregate_data'
as_epidist_marginal_model(
  data,
  obs_time_threshold = 2,
  delay_min = NULL,
  primary = .primary_choices(),
  ...
)
```

Arguments

<code>data</code>	An <code>epidist_aggregate_data</code> object
<code>obs_time_threshold</code>	Ratio used to determine threshold for setting relative observation times to Inf. Observation times greater than <code>obs_time_threshold</code> times the maximum delay will be set to Inf to improve model efficiency by reducing the number of unique observation times. Default is 2.
<code>delay_min</code>	Minimum delay (left truncation point). Can be: • NULL (default): uses a <code>delay_min</code> column from the data if present, otherwise defaults to 0 (no left truncation). • A numeric scalar: applied to all observations. • A column name string: looks up the named column in the data. This is passed as the <code>L</code> parameter to <code>primarycensored::dpcens()</code>.
<code>primary</code>	The distribution of the primary event within its censoring window. "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the <code>pgrowth</code> distributional parameter.
<code>...</code>	Not used in this method.

Value

An object of class `epidist_marginal_model`.

See Also

Other marginal_model: `as_epidist_marginal_model()`, `as_epidist_marginal_model.epidist_linelist_data()`, `epidist_family_model.epidist_marginal_model()`, `epidist_formula_model.epidist_marginal_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`, `new_epidist_marginal_model()`

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  ) |>
  as_epidist_marginal_model()
```

`as_epidist_marginal_model.epidist_linelist_data`

The marginal model method for epidist_linelist_data objects

Description

This method converts linelist data to a marginal model format by calculating delays between primary and secondary events, along with observation times and censoring windows. The likelihood used is imported from the **primarycensored** package which handles censoring in both primary and secondary events as well as truncation due to observation times. In principle, this method should be more accurate and more computationally efficient than the latent model (`as_epidist_latent_model()`) approach in most settings except when the number of unique strata approaches the number of observations.

Usage

```
## S3 method for class 'epidist_linelist_data'
as_epidist_marginal_model(
  data,
  obs_time_threshold = 2,
  weight = NULL,
  primary = .primary_choices(),
  delay_min = NULL,
  ...
)
```

Arguments

`data` An `epidist_linelist_data` object

obs_time_threshold	Ratio used to determine threshold for setting relative observation times to Inf. Observation times greater than obs_time_threshold times the maximum delay will be set to Inf to improve model efficiency by reducing the number of unique observation times. Default is 2.
weight	A column name containing counts of identical linelist items. When specified, the user is declaring that rows with the same values represent the same observation occurring multiple times. This allows for efficient data representation by storing unique patterns with their counts rather than repeating identical rows. The marginal model will further aggregate these counts based on the formula specification. Default is NULL, which assigns a count of 1 to each row. Internally this is used to define the 'n' column of the returned object.
primary	The distribution of the primary event within its censoring window. "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the pgrowth distributional parameter.
delay_min	Minimum delay (left truncation point). Can be: • NULL (default): uses a delay_min column from the data if present, otherwise defaults to 0 (no left truncation). • A numeric scalar: applied to all observations. • A column name string: looks up the named column in the data. This is passed as the L parameter to <code>primarycensored::dpcens()</code>.
...	Not used in this method.

Details

The marginal model performs internal aggregation to optimize computational efficiency while preserving all statistical information. If your data already contains repeated observations with identical characteristics, you can use the weight parameter to provide counts of these duplicates. This allows for more efficient data representation without any loss of information.

When a formula is specified in `epidist()`, the data will be transformed using `epidist_transform_data_model.epidist_marginal_model()` to prepare it for model fitting. This transformation summarises the data by counting unique combinations of delays, observation times, censoring windows and any variables in the model formula.

Value

An object of class `epidist_marginal_model`.

See Also

Other marginal_model: `as_epidist_marginal_model()`, `as_epidist_marginal_model.epidist_aggregate_data()`, `epidist_family_model.epidist_marginal_model()`, `epidist_formula_model.epidist_marginal_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`, `new_epidist_marginal_model()`

Examples

```
sierra_leone Ebola_data |>
  as_epidist_linelist_data(
```

```

  pdate_lwr = "date_of_symptom_onset",
  sdate_lwr = "date_of_sample_tested"
) |>
as_epidist_marginal_model()

```

as_epidist_meta_model *Convert an object to an epidist_meta_model object*

Description

Creates an `epidist_meta_model` object from individual level data, published summary estimates, or a mix of the two. This enables fitting a single delay distribution to all of the evidence available using `epidist()`.

Usage

```
as_epidist_meta_model(data = NULL, estimates = NULL, ...)
```

Arguments

<code>data</code>	An <code>epidist_linelist_data</code> or <code>epidist_aggregate_data</code> object of individual level observations, an <code>epidist_estimates_data</code> object of published summary estimates, or <code>NULL</code> .
<code>estimates</code>	An <code>epidist_estimates_data</code> object of published summary estimates, or <code>NULL</code> .
<code>...</code>	Additional arguments passed to methods.

Details

The meta model is experimental. Its interface may still change in future releases.

Individual level rows use the same likelihood as the marginal model (see `as_epidist_marginal_model()`), imported from the `primarycensored` package. Summary rows are instead forward modelled. Given the delay distribution, the model works out what the study's own estimation procedure would have converged to, and fits the reported value to that. Published estimates that did not adjust for right truncation, or that treated interval censored data as continuous, can therefore still contribute unbiased information. That holds only where the metadata describing what each study did is correct. It is usually the analyst's judgement rather than something the study reported, so state it explicitly and vary it in a sensitivity analysis. `vignette("model")` gives the forward model, the sampling likelihoods and the accuracy of the approximations, and `vignette("meta")` works through a simulated and a real example.

At least one of `data` and `estimates` must be supplied. Study level heterogeneity is specified through the brms formula in `epidist()`, for example $\mu \sim 1 + (1 \mid \text{study})$, rather than through this function. Individual level rows are labelled "individual" in the `study` column so that they form their own level of any such term.

Value

An object of class `epidist_meta_model`.

What this means in practice

Summaries that one study computed from the same delays are correlated, so they are fitted jointly. Two are grouped when they agree on every column of `as_epidist_estimates_data()` other than the summary itself, and a summary supplied with its own se is fitted alone. A study that reported integer date differences (cens_adjusted 0 or 3) is the exception: its mean and standard deviation form one group and its quantiles another. One observation is therefore a group rather than a single reported value, so `log_lik()` and `loo::loo()` report per group, and loo only compares fits to the same studies and the same mix of individual and summary rows. Summary rows are evaluated in R for each posterior draw, so both are slow for a fit with many draws and summary rows. Pass `ndraws` to use fewer draws, as `epidist_gen_meta_log_lik()` explains. See `vignette("faq")`. `epidist_meta_leave_one_out()` asks the study level question instead, refitting with each study held out.

The sampling standard errors are plug in quantities that depend on the parameters, so allow for genuine differences between studies with a term such as $\mu \sim 1 + (1 \mid \text{study})$ rather than relying on them alone. Supply a reported se in `as_epidist_estimates_data()` for quantiles read off a fitted distribution rather than the empirical data. Where a study reported integer date differences, keep its mean and standard deviation and drop its quantiles. `as_epidist_estimates_data()` warns for the studies the approximations serve least well, and documents the two settings, `max_delay` and `options(epidist.meta_n_quad = )`, that trade accuracy against speed.

Advanced: an estimated growth rate

A summary row tilts its primary event, and weights the follow up of an accrual design, by the `growth_rate` of its study. Where that rate is NA in `as_epidist_estimates_data()`, or was given there with a `growth_rate_sd`, the study estimates it as the `pgrowth` distributional parameter instead. That is the parameter `primary = "expgrowth"` estimates from individual level rows, so the two can share it. The model adds `pgrowth ~ 0 + study` unless a `pgrowth` formula is given, and `epidist_model_prior()` sets the priors from the reported rates. Summaries carry little information about the rate on their own, so share the coefficient with rows that do inform it, for example `pgrowth ~ 1` with a line list from the same outbreak. `vignette("model")` gives the details.

See Also

`as_epidist_estimates_data()` for preparing the summary estimates, and for the checks and the settings the approximations here depend on.

Other meta_model: `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_aggregate_data()`, `as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_model()`, `epidist_transform_data_model.epidist_meta_model()`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

`as_epidist_meta_model.epidist_aggregate_data`

The meta model method for epidist_aggregate_data objects

Description

This method converts aggregate data to a meta model format by passing it to `as_epidist_meta_model.epidist_linelist_data` with the `n` column used as weights.

Usage

```
## S3 method for class 'epidist_aggregate_data'
as_epidist_meta_model(
  data,
  estimates = NULL,
  obs_time_threshold = 2,
  delay_min = NULL,
  primary = .primary_choices(),
  ...
)
```

Arguments

<code>data</code>	An <code>epidist_aggregate_data</code> object.
<code>estimates</code>	An <code>epidist_estimates_data</code> object of published summary estimates, or <code>NULL</code> .
<code>obs_time_threshold</code>	Ratio used to determine threshold for setting relative observation times to <code>Inf</code> . Observation times greater than <code>obs_time_threshold</code> times the maximum delay will be set to <code>Inf</code> to improve model efficiency by reducing the number of unique observation times. Default is 2.
<code>delay_min</code>	Minimum delay (left truncation point). Can be: • <code>NULL</code> (default): uses a <code>delay_min</code> column from the data if present, otherwise defaults to 0 (no left truncation). • A numeric scalar: applied to all observations. • A column name string: looks up the named column in the data. This is passed as the <code>L</code> parameter to <code>primarycensored::dpcens()</code>.
<code>primary</code>	The distribution of the primary event within its censoring window for the individual level rows, as in <code>as_epidist_marginal_model.epidist_linelist_data()</code> . "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the <code>pgrowth</code> distributional parameter. Summary rows are unaffected. They tilt the primary event by the <code>growth_rate</code> metadata of their study, or estimate it as the same <code>pgrowth</code> parameter where that is <code>NA</code> or has a <code>growth_rate_sd</code> , see <code>as_epidist_meta_model()</code> .
<code>...</code>	Additional arguments passed to methods.

Value

An object of class `epidist_meta_model`.

See Also

Other meta_model: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`
`as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`,
`epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_model()`,
`epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_`
`epidist_transform_data_model.epidist_meta_model()`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  ) |>
  as_epidist_meta_model()
```

`as_epidist_meta_model.epidist_estimates_data`

The meta model method for epidist_estimates_data objects

Description

Allows summary estimates to be passed as the first argument, so that `as_epidist_meta_model(estimates)` and `as_epidist_meta_model(estimates = estimates)` are equivalent.

Usage

```
## S3 method for class 'epidist_estimates_data'
as_epidist_meta_model(data, estimates = NULL, ...)
```

Arguments

<code>data</code>	An <code>epidist_estimates_data</code> object.
<code>estimates</code>	An <code>epidist_estimates_data</code> object of published summary estimates, or <code>NULL</code> .
<code>...</code>	Additional arguments passed to methods.

Details

A model built from summaries alone takes no primary argument, because summary rows tilt the primary event by the `growth_rate` metadata of their study, estimating it only where that is NA or has a `growth_rate_sd`. Passing one is an error.

Value

An object of class `epidist_meta_model`.

See Also

Other meta_model: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`
`as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`,
`epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_model()`,
`epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_`
`epidist_transform_data_model.epidist_meta_model()`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

Examples

```
estimates <- as_epidist_estimates_data(
  data.frame(
    study = c("A", "A"),
    type = c("mean", "sd"),
    value = c(7.5, 3.6),
    n = c(120, 120),
    relative_obs_time = c(20, 20),
    trunc_adjusted = c(FALSE, FALSE),
    cens_adjusted = c(0, 0)
  )
)
as_epidist_meta_model(estimates)
```

```
as_epidist_meta_model.epidist_linelist_data
```

The meta model method for epidist_linelist_data objects

Description

Prepares individual level data exactly as `as_epidist_marginal_model.epidist_linelist_data()` does and then stacks it with any supplied summary estimates.

Usage

```
## S3 method for class 'epidist_linelist_data'
as_epidist_meta_model(
  data,
  estimates = NULL,
  obs_time_threshold = 2,
  weight = NULL,
  delay_min = NULL,
  primary = .primary_choices(),
  ...
)
```

Arguments

<code>data</code>	An <code>epidist_linelist_data</code> object.
<code>estimates</code>	An <code>epidist_estimates_data</code> object of published summary estimates, or <code>NULL</code> .
<code>obs_time_threshold</code>	Ratio used to determine threshold for setting relative observation times to <code>Inf</code> . Observation times greater than <code>obs_time_threshold</code> times the maximum delay will be set to <code>Inf</code> to improve model efficiency by reducing the number of unique observation times. Default is 2.
<code>weight</code>	A column name containing counts of identical linelist items. When specified, the user is declaring that rows with the same values represent the same observation occurring multiple times. This allows for efficient data representation by storing unique patterns with their counts rather than repeating identical rows. The marginal model will further aggregate these counts based on the formula specification. Default is <code>NULL</code> , which assigns a count of 1 to each row. Internally this is used to define the 'n' column of the returned object.
<code>delay_min</code>	Minimum delay (left truncation point). Can be: • <code>NULL</code> (default): uses a <code>delay_min</code> column from the data if present, otherwise defaults to 0 (no left truncation). • A numeric scalar: applied to all observations. • A column name string: looks up the named column in the data. This is passed as the <code>L</code> parameter to <code>primarycensored::dpcens()</code>.
<code>primary</code>	The distribution of the primary event within its censoring window for the individual level rows, as in <code>as_epidist_marginal_model.epidist_linelist_data()</code> . "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the <code>pgrowth</code> distributional parameter. Summary rows are unaffected. They tilt the primary event by the <code>growth_rate</code> metadata of their study, or estimate it as the same <code>pgrowth</code> parameter where that is <code>NA</code> or has a <code>growth_rate_sd</code> , see <code>as_epidist_meta_model()</code> .
<code>...</code>	Additional arguments passed to methods.

Value

An object of class `epidist_meta_model`.

See Also

Other `meta_model`: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_a`, `as_epidist_meta_model.epidist_estimates_data()`, `assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_model()`, `epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_r`, `epidist_transform_data_model.epidist_meta_model()`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

Examples

```
sierra_leone Ebola_data |>
  as_epidist_linelist_data(
```

```

    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_meta_model()

```

as_epidist_meta_model.NULL

The meta model method for summary estimates only

Description

Used when no individual level data is available and only the estimates argument is supplied. It takes no primary argument, because summary rows tilt the primary event by the growth_rate metadata of their study, estimating it only where that is NA or has a growth_rate_sd. Passing one is an error.

Usage

```

## S3 method for class '`NULL`'
as_epidist_meta_model(data = NULL, estimates = NULL, ...)

```

Arguments

data	NULL.
estimates	An epidist_estimates_data object of published summary estimates, or NULL.
...	Additional arguments passed to methods.

Value

An object of class epidist_meta_model.

See Also

Other meta_model: [as_epidist_meta_model\(\)](#), [as_epidist_meta_model.epidist_aggregate_data\(\)](#), [as_epidist_meta_model.epidist_estimates_data\(\)](#), [as_epidist_meta_model.epidist_linelist_data\(\)](#), [assert_epidist.epidist_meta_model\(\)](#), [epidist_family_model.epidist_meta_model\(\)](#), [epidist_formula_model.epidist_meta_leave_one_out\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_newdata.epidist_meta_model\(\)](#), [epidist_transform_data_model.epidist_meta_model\(\)](#), [is_epidist_meta_model\(\)](#), [new_epidist_meta_model\(\)](#)

Examples

```

estimates <- as_epidist_estimates_data(
  data.frame(
    study = c("A", "A"),
    type = c("mean", "sd"),
    value = c(7.5, 3.6),
    n = c(120, 120),
    relative_obs_time = c(20, 20),

```

```

        trunc_adjusted = c(FALSE, FALSE),
        cens_adjusted = c(0, 0)
    )
)
as_epidist_meta_model(estimates = estimates)

```

as_epidist_multivariate

Create an epidist_multivariate object

Description

Summarises draws of a set of parameters by their mean vector and covariance matrix. The result is a multivariate normal approximation to whatever the draws describe, and it is the only route by which a covariance between reported quantities reaches `as_epidist_meta_model()`.

Usage

```
as_epidist_multivariate(draws, ...)
```

Arguments

<code>draws</code>	Draws of the parameters. See the methods for supported formats.
<code>...</code>	Additional arguments passed to methods.

Details

The draws may be of any parameters. Posterior draws of a delay mean and standard deviation, of a meanlog and sdlog, or of a shape and a scale all give the same object. Nothing here checks that a parameter is something a study could report, because that only matters when the object is converted with `as_epidist_estimates_data()`.

Draws may also cover a trajectory, with the parameters varying over an index. The elements are then ordered index major and parameter minor, so a mean and an sd over two index points give `mean[1]`, `sd[1]`, `mean[2]`, `sd[2]`. The order is fixed because the covariance is indexed by it.

There is no method for a vector of individual delays. Resampling delays manufactures a covariance the study never estimated, which is not what this represents.

Value

An object of class `epidist_multivariate`.

See Also

Other multivariate: `as_epidist_multivariate.data.frame()`, `as_epidist_multivariate.matrix()`, `assert_epidist.epidist_multivariate()`, `is_epidist_multivariate()`, `new_epidist_multivariate()`, `print.epidist_multivariate()`, `vcov.epidist_multivariate()`

```
as_epidist_multivariate.data.frame
```

Create an epidist_multivariate object from a data frame of draws

Description

`delay_parameter_draws()` returns exactly this shape, with a `.draw` column, a `.row` column and one column per parameter, so its output can be passed straight in. Add `add_summaries()` to report the natural scale mean and standard deviation rather than the distributional parameters.

Usage

```
## S3 method for class 'data.frame'
as_epidist_multivariate(draws, params = NULL, index = NULL, draw = NULL, ...)
```

Arguments

<code>draws</code>	Draws of the parameters. See the methods for supported formats.
<code>params</code>	A character vector of the columns holding the parameter draws. Defaults to every numeric column other than <code>draw</code> and <code>index</code> . Supply it wherever the draws carry more columns than the parameters being reported, because parameters that are functions of each other have a singular covariance.
<code>index</code>	A string giving the column that identifies the trajectory point. Defaults to <code>".row"</code> where such a column exists, as <code>delay_parameter_draws()</code> returns, and otherwise to <code>NULL</code> , meaning the draws describe a single point.
<code>draw</code>	A string giving the column that identifies the draw. Defaults to <code>".draw"</code> where such a column exists, and otherwise to <code>NULL</code> , meaning row order.
<code>...</code>	Not used in this method.

Value

An object of class `epidist_multivariate`.

See Also

Other multivariate: `as_epidist_multivariate()`, `as_epidist_multivariate.matrix()`, `assert_epidist.epidist_mu`, `is_epidist_multivariate()`, `new_epidist_multivariate()`, `print.epidist_multivariate()`, `vcov.epidist_multivariate()`

Examples

```
set.seed(1)
draws <- data.frame(
  mean = rnorm(500, 7.5, 0.3), sd = rnorm(500, 3.6, 0.2)
)
as_epidist_multivariate(draws)
```

as_epidist_multivariate.matrix

Create an epidist_multivariate object from a matrix of draws

Description

Rows are draws and columns are parameters, describing a single point.

Usage

```
## S3 method for class 'matrix'
as_epidist_multivariate(draws, params = NULL, ...)
```

Arguments

draws	Draws of the parameters. See the methods for supported formats.
params	A character vector naming the columns of draws. Defaults to the column names of draws.
...	Not used in this method.

Value

An object of class `epidist_multivariate`.

See Also

Other multivariate: [as_epidist_multivariate\(\)](#), [as_epidist_multivariate.data.frame\(\)](#), [assert_epidist_epidist_multivariate\(\)](#), [is_epidist_multivariate\(\)](#), [new_epidist_multivariate\(\)](#), [print.epidist_multivariate\(\)](#), [vcov.epidist_multivariate\(\)](#)

Examples

```
set.seed(1)
draws <- cbind(mean = rnorm(500, 7.5, 0.3), sd = rnorm(500, 3.6, 0.2))
as_epidist_multivariate(draws)
```

as_epidist_naive_model

Convert an object to an epidist_naive_model object

Description

Creates an `epidist_naive_model` object from various input formats. This enables fitting naive models for epidemiological delays using [epidist\(\)](#). The naive model approach ignores censoring and truncation in the data, using only the lower bounds of the intervals as point estimates. This is the simplest approach but may lead to biased estimates if there is substantial censoring or truncation in the data.

Usage

```
as_epidist_naive_model(data, ...)
```

Arguments

data	An object to be converted to the class <code>epidist_naive_model</code>
...	Additional arguments passed to methods.

Value

An object of class `epidist_naive_model`.

See Also

Other `naive_model`: `as_epidist_naive_model.epidist_aggregate_data()`, `as_epidist_naive_model.epidist_line`, `epidist_formula_model.epidist_naive_model()`, `epidist_newdata.epidist_naive_model()`, `epidist_stancode.epidist_naive_model()`, `epidist_transform_data_model.epidist_naive_model()`, `is_epidist_naive_model()`, `new_epidist_naive_model()`

```
as_epidist_naive_model.epidist_aggregate_data
```

The naive model method for `epidist_aggregate_data` objects

Description

This method converts aggregate data to a naive model format by passing it to `as_epidist_naive_model.epidist_linelist` with the `n` column used as weights. This ensures that the likelihood is weighted by the counts in the aggregate data.

Usage

```
## S3 method for class 'epidist_aggregate_data'
as_epidist_naive_model(data, ...)
```

Arguments

data	An <code>epidist_aggregate_data</code> object.
...	Additional arguments passed to methods.

Value

An object of class `epidist_naive_model`.

See Also

Other naive_model: [as_epidist_naive_model\(\)](#), [as_epidist_naive_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_naive_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#), [epidist_stancode.epidist_naive_model\(\)](#), [epidist_transform_data_model.epidist_naive_model\(\)](#), [is_epidist_naive_model\(\)](#), [new_epidist_naive_model\(\)](#)

Examples

```
sierra_leone_ebola_data |>
  dplyr::count(date_of_symptom_onset, date_of_sample_tested) |>
  as_epidist_aggregate_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested",
    n = "n"
  ) |>
  as_epidist_naive_model()
```

```
as_epidist_naive_model.epidist_linelist_data
```

The naive model method for epidist_linelist_data objects

Description

This method converts linelist data to a naive model format by calculating delays between primary and secondary events to enable model fitting in [epidist\(\)](#). If the input data contains an n column (e.g. from aggregated data), the likelihood will be weighted by these counts.

Usage

```
## S3 method for class 'epidist_linelist_data'
as_epidist_naive_model(data, weight = NULL, ...)
```

Arguments

data	An epidist_linelist_data object.
weight	A column name to use for weighting. If NULL, weights default to 1. Internally this is used to define the 'n' column of the returned object.
...	Additional arguments passed to methods.

Details

When a formula is specified in [epidist\(\)](#), the data will be transformed using [epidist_transform_data_model.epidist_n](#) to prepare it for model fitting. This transformation summarises the data by counting unique combinations of delays and any variables in the model formula.

The naive model is the simplest approach but ignores censoring and truncation in the data by using only lower bounds as point estimates. For data with substantial censoring or truncation, consider using [as_epidist_latent_model\(\)](#) or [as_epidist_marginal_model\(\)](#) which properly account for these features.

Value

An object of class `epidist_naive_model`.

See Also

Other `naive_model`: [as_epidist_naive_model\(\)](#), [as_epidist_naive_model.epidist_aggregate_data\(\)](#), [epidist_formula_model.epidist_naive_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#), [epidist_stancode.epidist_naive_model\(\)](#), [epidist_transform_data_model.epidist_naive_model\(\)](#), [is_epidist_naive_model\(\)](#), [new_epidist_naive_model\(\)](#)

Examples

```
sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_naive_model()
```

`delay_parameter_draws` *Posterior draws of the delay distribution parameters*

Description

Returns posterior draws of the parameters of the delay distribution in the long format used by `tidybayes`. The delay parameters are the distributional parameters of the `brms` family, evaluated on the response scale for each row of `newdata`. For a lognormal model they are `mu` and `sigma`. They are the parameters of the delay distribution itself, so they do not describe the censoring or truncation of the observation process, and they are not the natural scale mean and standard deviation of the delay. Use [add_summaries\(\)](#) to add those.

`add_delay_parameter_draws()` is the same function with `newdata` first, for use at the start of a pipeline as with [tidybayes::add_epred_draws\(\)](#).

Usage

```
delay_parameter_draws(object, newdata = NULL, ...)
```

```
add_delay_parameter_draws(newdata, object, ...)
```

Arguments

<code>object</code>	A model fit with epidist() .
<code>newdata</code>	A <code>data.frame</code> of data to predict for. If <code>NULL</code> , the default, the data the model was fitted to is used. The <code>brms</code> models <code>epidist</code> fits need the model variables as well as the predictors, so build <code>newdata</code> with epidist_strata() rather than from the predictors alone.
<code>...</code>	Additional arguments passed to brms::prepare_predictions() , such as <code>ndraws</code> or <code>re_formula</code> .

Details

The returned columns follow the tidybayes conventions. The columns of newdata come first, followed by .row, .chain, .iteration and .draw, followed by one column per distributional parameter. The result is grouped by the columns of newdata and by .row. .chain and .iteration are NA when the draws have been subset, because the chain a subset draw came from is not recoverable.

Every row of newdata gets its own draws, so passing the data the model was fitted to produces many identical draws when the model has few unique combinations of predictors. `epidist_strata()` returns one row per unique combination and is usually the better input.

Value

A tibble of posterior draws of the delay distribution parameters, grouped by the columns of newdata and by .row. It has the `epidist_delay_draws` class, which records the delay distribution family and gives it a `plot()` method.

See Also

`add_summaries()` to add natural scale summaries of the delay, `epidist_strata()` to build newdata, and `plot.epidist_delay_draws()` to plot the draws.

Other postprocess: `add_summaries()`, `as_dist_spec.epidist_fit()`, `delay_summary_draws()`, `epidist_delay_draws`, `epidist_strata()`

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

fit |>
  epidist_strata() |>
  add_delay_parameter_draws(fit) |>
  add_summaries(probs = c(0.05, 0.95))
```

Description

The usual last step of a fit. Builds one row per unique combination of the predictors with `epidist_strata()`, draws the delay distribution parameters for each with `delay_parameter_draws()`, and adds the natural scale mean and standard deviation, and any quantiles asked for, with `add_summaries()`.

Each of those three steps is available on its own where a step needs different arguments, or where newdata is built some other way.

Usage

```
delay_summary_draws(
  object,
  newdata = NULL,
  vars = NULL,
  probs = NULL,
  method = c("auto", "analytic", "sample"),
  nsim = 1000,
  ...
)
```

Arguments

<code>object</code>	A model fit with <code>epidist()</code> .
<code>newdata</code>	A data.frame of data to predict for. If NULL, the default, <code>epidist_strata()</code> builds one row per unique combination of the predictors the model uses.
<code>vars</code>	A character vector of variables to stratify by, passed to <code>epidist_strata()</code> . Only used when newdata is NULL.
<code>probs</code>	A numeric vector of probabilities to add quantiles of the delay distribution for. If NULL, the default, no quantiles are added.
<code>method</code>	Passed to <code>add_summaries()</code> . Either "auto", the default, which uses the analytic solution when there is one and simulates otherwise, "analytic", or "sample".
<code>nsim</code>	The number of delays to simulate per row. Passed to <code>add_summaries()</code> and only used when simulating.
<code>...</code>	Additional arguments passed to <code>brms::prepare_predictions()</code> , such as <code>ndraws</code> or <code>re_formula</code> .

Value

A tibble of posterior draws of the delay distribution parameters with mean and sd columns added, and one column per element of probs. Grouped as `delay_parameter_draws()` returns it.

See Also

`delay_parameter_draws()` for the parameters alone, `add_summaries()` to summarise draws you already have, and `epidist_strata()` to build newdata.

Other postprocess: `add_summaries()`, `as_dist_spec.epidist_fit()`, `delay_parameter_draws()`, `epidist_delay_draws`, `epidist_strata()`

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

delay_summary_draws(fit, probs = c(0.05, 0.95))
```

epidist

Fit epidemiological delay distributions using a brms interface

Description

Fit epidemiological delay distributions using a brms interface

Usage

```
epidist(
  data,
  formula = mu ~ 1,
  family = lognormal(),
  prior = NULL,
  merge_priors = TRUE,
  fn = brms::brm,
  ...
)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see <code>brms::brmsfamily()</code> . Commonly used, such as <code>brms::lognormal()</code> , are also reexported as part of <code>epidist</code> .

prior	One or more <code>brmsprior</code> objects created by <code>brms::set_prior()</code> or related functions. These priors are passed to <code>epidist_prior()</code> in the <code>prior</code> argument. Some models have default priors that are automatically added (see <code>epidist_model_prior()</code>). These can be merged with user-provided priors using the <code>merge_priors</code> argument.
merge_priors	If TRUE then merge user priors with default priors, if FALSE only use user priors. Defaults to TRUE. This may be useful if the built in approaches for merging priors are not flexible enough for a particular use case.
fn	The internal function to be called. By default this is <code>brms::brm()</code> which performs inference for the specified model. Other options are <code>brms::make_stancode()</code> which returns the Stan code for the specified model, or <code>brms::make_standata()</code> which returns the data passed to Stan. These two later options may be useful for model debugging and extensions.
...	Additional arguments passed to <code>fn</code> method.

Value

A fitted model of class `epidist_fit`.

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

summary(fit)
```

epidist_data

Keep epidist objects in their class

Description

Every `epidist` data object also carries the `epidist_data` class. The methods documented here re-check an object after it has been modified and drop any `epidist` class whose requirements the modified object no longer meets, warning about what was dropped and why. An object that still carries an `epidist` class is therefore a valid object of that class, so functions which accept one do not need to re-check it.

Usage

```

## S3 method for class 'epidist_data'
x[...]

## S3 replacement method for class 'epidist_data'
x[...] <- value

## S3 replacement method for class 'epidist_data'
x[[...]] <- value

## S3 replacement method for class 'epidist_data'
x$... <- value

## S3 replacement method for class 'epidist_data'
names(x) <- value

## S3 method for class 'epidist_data'
rbind(..., deparse.level = 1)

## S3 method for class 'epidist_data'
dplyr_reconstruct(data, template)

## S3 method for class 'epidist_data'
dplyr_row_slice(data, i, ...)

## S3 method for class 'epidist_data'
dplyr_col_modify(data, cols)

## S3 method for class 'epidist_data'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))

## S3 method for class 'epidist_data'
ungroup(x, ...)

```

Arguments

<code>x, .data</code>	An object with the <code>epidist_data</code> class.
<code>...</code>	Passed to the underlying method.
<code>value</code>	A replacement value.
<code>deparse.level</code>	Passed to <code>base::rbind()</code> .
<code>data, template</code>	Passed to <code>dplyr::dplyr_reconstruct()</code> .
<code>i, cols</code>	Passed to <code>dplyr::dplyr_row_slice()</code> and <code>dplyr::dplyr_col_modify()</code> .
<code>.add, .drop</code>	Passed to <code>dplyr::group_by()</code> .

Details

Methods are provided for base subsetting and replacement, and for `dplyr::dplyr_reconstruct()`, which dplyr verbs such as `dplyr::mutate()`, `dplyr::filter()` and `dplyr::select()` use to restore the class of their input. `dplyr::group_by()` and `dplyr::ungroup()` build a new tibble rather than restoring the class of their input, as do the grouped_df methods for `dplyr::dplyr_row_slice()` and `dplyr::dplyr_col_modify()`, so each has a method of its own that puts the classes back. A grouped object keeps the grouped_df class after the epidist classes, and the dplyr verbs keep both. `dplyr::summarise()` builds a new object from the groups rather than modifying its input, so its result does not carry the epidist classes.

A result with no columns is unclassed without a warning. Such a result is almost always the prototype `vctrs` takes internally, in `dplyr::bind_cols()` for example, rather than something the user asked for, and it cannot be told apart from a deliberate empty selection.

Value

The modified object with any epidist class whose requirements it no longer meets removed.

See Also

Other epidist_data: `is_epidist_data()`

Examples

```
linelist_data <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  )

# Subsetting rows keeps the class
class(linelist_data[1:10, ])

# Dropping a required column drops the class
class(dplyr::select(linelist_data, ~"obs_time"))

# Grouping keeps the class alongside the grouped_df class
class(dplyr::group_by(linelist_data, obs_time))
```

epidist_delay_draws *Keep the epidist_delay_draws class through dplyr verbs*

Description

The `epidist_delay_draws` class records the delay distribution family and the variables that define the strata, which `add_summaries()` and `plot.epidist_delay_draws()` read. Most dplyr verbs build a new object rather than keeping the class of their input, so the methods documented here put the class and what it records back.

Usage

```
## S3 method for class 'epidist_delay_draws'
x[...]
```

```
## S3 replacement method for class 'epidist_delay_draws'
names(x) <- value
```

```
## S3 method for class 'epidist_delay_draws'
dplyr_reconstruct(data, template)
```

```
## S3 method for class 'epidist_delay_draws'
dplyr_row_slice(data, i, ...)
```

```
## S3 method for class 'epidist_delay_draws'
dplyr_col_modify(data, cols)
```

```
## S3 method for class 'epidist_delay_draws'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))
```

```
## S3 method for class 'epidist_delay_draws'
ungroup(x, ...)
```

Arguments

<code>x, .data</code>	An <code>epidist_delay_draws</code> object.
<code>...</code>	Passed to the underlying method.
<code>value</code>	A replacement value.
<code>data, template</code>	Passed to <code>dplyr::dplyr_reconstruct()</code> .
<code>i, cols</code>	Passed to <code>dplyr::dplyr_row_slice()</code> and <code>dplyr::dplyr_col_modify()</code> .
<code>.add, .drop</code>	Passed to <code>dplyr::group_by()</code> .

Details

Methods are provided for base subsetting and renaming, and for `dplyr::dplyr_reconstruct()`, which verbs such as `dplyr::mutate()` and `dplyr::bind_rows()` use to restore the class of their input. `dplyr::group_by()` and `dplyr::ungroup()` build a new tibble rather than restoring the class of their input, as do the grouped_df methods for `dplyr::dplyr_row_slice()` and `dplyr::dplyr_col_modify()`, so each has a method of its own that puts the class back. A grouped object keeps the `epidist_delay_draws` class ahead of grouped_df, and the dplyr verbs keep both. `dplyr::summarise()` builds a new object from the groups rather than modifying its input, so its result does not carry the class.

`dplyr::bind_rows()` restores the class from its first argument, so combining draws keeps the family and the stratum variables of the first set of draws. Combining draws from two fits of different families therefore describes the result by the family of the first. Pass the family to `add_summaries()` or `plot()` with the family argument when the draws combined are not all from the same family.

Value

The modified object with the `epidist_delay_draws` class, and the family and stratum variables it records, put back.

See Also

Other postprocess: `add_summaries()`, `as_dist_spec.epidist_fit()`, `delay_parameter_draws()`, `delay_summary_draws()`, `epidist_strata()`

Examples

```
draws <- data.frame(mu = c(1.8, 2.0), sigma = c(0.5, 0.4)) |>
  add_summaries(family = "lognormal")

# Adding a column keeps the class
class(dplyr::mutate(draws, model = "a"))

# Combining two sets of draws keeps the class
class(dplyr::bind_rows(draws, draws))
```

epidist_diagnostics *Diagnostics for epidist_fit models*

Description

This function computes diagnostics to assess the quality of a fitted model. When the fitting algorithm used is "sampling" (HMC) then the output of `epidist_diagnostics` is a `data.frame` containing:

- `time`: the total time taken to fit all chains
- `samples`: the total number of samples across all chains
- `max_rhat`: the highest value of the Gelman-Rubin statistic
- `divergent_transitions`: the total number of divergent transitions
- `per_divergent_transitions`: the proportion of samples which had divergent transitions
- `max_tredepth`: the highest value of the tredepth HMC parameter
- `no_at_max_tredepth`: the number of samples which attained the `max_tredepth`
- `per_at_max_tredepth`: the proportion of samples which attained the `max_tredepth`

Usage

```
epidist_diagnostics(fit)
```

Arguments

`fit` A fitted model of class `epidist_fit`

Details

When the fitting algorithm is not "sampling" (see `brms::brm()` for other possible algorithms) then diagnostics are yet to be implemented.

Value

A data.frame of summary diagnostics for the fit.

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))
epidist_diagnostics(fit)
```

epidist_estimates_epireview

Report studies from an epireview parameter table

Description

Maps a table of published delay estimates from the `epireview` package, one row per reported estimate, to the long format `as_epidist_estimates_data()` takes, and returns the `epidist_estimates_data` object it builds. `epireview` collates the parameter estimates gathered by the Pathogen Epidemiology Review Group. It is not on CRAN, and is installed from the `mrc-ide r-universe`. Filter the `params` table of `epireview::load_epidata()` to one delay before passing it here, because the rows of one object all describe the same delay.

Usage

```
epidist_estimates_epireview(
  data,
  study = "article_label",
  metadata = NULL,
  keep = NULL,
  advise = TRUE,
  ...
)
```

Arguments

data	A data.frame of epireview delay estimates, such as the params element of <code>epireview::load_epidata()</code> filtered to one delay. It needs the <code>parameter_value</code> , <code>parameter_value_type</code> and <code>population_sample_size</code> columns and the column named by <code>study</code> . The other <code>parameter_*</code> and <code>distribution_*</code> columns are used where present.
study	A string naming the column of data that identifies the study. Defaults to "article_label".
metadata	A data.frame of study metadata, or NULL. It needs a <code>study</code> column holding values of the <code>study</code> column of data, and takes any of <code>n</code> , <code>pwindow</code> , <code>swindow</code> , <code>relative_obs_time</code> , <code>trunc_adjusted</code> , <code>trunc_design</code> , <code>cens_adjusted</code> , <code>delay_min</code> and <code>growth_rate</code> , as documented in <code>as_epidist_estimates_data.data.frame()</code> . Each value replaces the value of that column for the study's records, and an NA leaves it as it is.
keep	A character vector of columns of data to carry onto the rows of the result, for use as covariates. Defaults to NULL.
advise	Whether to run the advisory checks of <code>as_epidist_estimates_data.data.frame()</code> . Defaults to TRUE.
...	Study metadata applied to every record, each a single value, as documented in <code>as_epidist_estimates_data.data.frame()</code> . Any of <code>pwindow</code> , <code>swindow</code> , <code>relative_obs_time</code> , <code>trunc_adjusted</code> , <code>trunc_design</code> , <code>cens_adjusted</code> , <code>delay_min</code> and <code>growth_rate</code> .

Value

An `epidist_estimates_data` object.

What each record becomes

A record is mapped by its `parameter_value_type`.

- A "Mean" becomes a "mean" row. A "Standard Deviation" in `parameter_uncertainty_singe_type` becomes a matching "sd" row, as does the second parameter of a fitted distribution that epireview records as a mean and a standard deviation. A "Standard Error" becomes the se of the mean row instead, and so does a 95% confidence or credible interval of the mean, as its width over $2 * \text{qnorm}(0.975)$.
- A "Median" becomes a "quantile" row at $p = 0.5$, with further rows at $p = 0.25$ and $p = 0.75$ where `parameter_uncertainty_type` is "IQR".
- A record whose `distribution_type` is a gamma, a Weibull or a lognormal reported by its natural parameters (a shape with a scale or a rate, or a meanlog with a sdlog) becomes the mean and standard deviation the fitted distribution implies over the delays the study could have seen, as `epidist_estimates_parameters()` computes them. Its reported value and spread are not used again, because they are functions of the same parameters. epireview records no uncertainty for the parameters, so the rows take their sampling uncertainty from the sample size.

A range is the smallest and largest delay a study saw rather than a summary of the distribution, and is not used. A spread that does not match the value type, such as a standard deviation reported

alongside a median, is not used either, because the two kinds of summary from a study that reported integer date differences are fitted as though they were independent.

Records that cannot be mapped are dropped with a message naming them. These are records reporting an inverse rate, units other than days, a value with a scaling exponent, no mean or median and no usable distribution parameters, and records left with no sample size and no standard error. Fill a missing sample size through metadata.

Several records may share a study, for example estimates stratified by outbreak or by group, and they are then fitted as separate summaries of the same study.

Study metadata

epireview does not record how a study handled censoring or right truncation, its censoring windows or its observation time. Give the metadata that applies to every record through `...`, and the metadata that differs by study through `metadata`, a data frame with one row per study whose values replace those of `...` for that study's records. Anything given for neither is assumed as `as_epidist_estimates_data()` assumes it, with the same messages, and a study that metadata leaves blank for a column it holds is assumed in the same way and named. The Checks section of `as_epidist_estimates_data.data.frame()` says which assumptions matter most.

The result is an `epidist_estimates_data` object, so metadata learnt later can be edited in with `dplyr::mutate()` or `[<-`. The object is checked again after each change and drops its class if the change breaks a requirement, so make related changes, such as setting `trunc_adjusted` to `FALSE` and giving the `relative_obs_time` it then needs, in one call. See [epidist_data](#).

See Also

Other estimates_data: `as_epidist_estimates_data()`, `as_epidist_estimates_data.data.frame()`, `as_epidist_estimates_data.epidist_estimates_data()`, `as_epidist_estimates_data.epidist_multivariate()`, `as_epidist_estimates_data.list()`, `assert_epidist.epidist_estimates_data()`, `epidist_estimates_parameter`, `epidist_estimates_summaries()`, `is_epidist_estimates_data()`, `new_epidist_estimates_data()`

Examples

```
# A table with the columns epireview uses
records <- data.frame(
  article_label = c("A 2015", "B 2016", "B 2016"),
  parameter_value = c(10.6, 14, 8.9),
  parameter_value_type = c("Mean", "Median", "Mean"),
  parameter_unit = "Days",
  parameter_uncertainty_single_value = c(3.2, NA, NA),
  parameter_uncertainty_singe_type = c("Standard Deviation", NA, NA),
  parameter_uncertainty_lower_value = c(NA, 11, NA),
  parameter_uncertainty_upper_value = c(NA, 15, NA),
  parameter_uncertainty_type = c(NA, "IQR", NA),
  population_sample_size = c(76, 20, 92)
)
epidist_estimates_epireview(
  records,
  metadata = data.frame(
    study = "A 2015", relative_obs_time = 60, trunc_adjusted = FALSE
  ),
)
```

```

    trunc_adjusted = TRUE,
    cens_adjusted = 0
  )

```

epidist_estimates_parameters

Report a study that published the parameters of a distribution it fitted

Description

Studies often publish the parameters of a distribution they fitted rather than summaries of the delays themselves. This converts those parameters into the summaries the fitted distribution implies, which is what `as_epidist_meta_model()` fits to.

Usage

```

epidist_estimates_parameters(
  study,
  family,
  parameters,
  moments = c("mean", "sd"),
  probs = numeric(0),
  se = NULL,
  n = NULL,
  ...
)

```

Arguments

<code>study</code>	A string naming the study.
<code>family</code>	The distribution the study fitted, one of "lognormal", "gamma" or "weibull".
<code>parameters</code>	A named numeric vector of the reported parameters. The names must be meanlog and sdlog for a lognormal, shape with either scale or rate for a gamma, and shape and scale for a weibull.
<code>moments</code>	Which moments to report, any of "mean" and "sd".
<code>probs</code>	A numeric vector of probabilities to report quantiles at.
<code>se</code>	A numeric vector of the reported standard errors of parameters, in the same order. Optional.
<code>n</code>	The number of delays the study fitted. Optional, and used for the sampling uncertainty of the summaries where no se is given.
<code>...</code>	Study metadata, as documented in <code>as_epidist_estimates_data.data.frame()</code> .

Value

An `epidist_estimates_data` object.

What the reported parameters are taken to mean

The reported parameters describe the distribution the study's own estimation procedure converged to. Where the procedure was correct, that is the delay distribution itself. Where it was not, it is the biased distribution the procedure targeted, which is exactly what the meta model forward models from the study metadata. Converting to summaries therefore covers both cases with one route, and the study metadata documented in `as_epidist_estimates_data.data.frame()` is supplied in the same way as for any other summary.

Because the returned rows are summaries, the family the study fitted need not match the family being fitted to it. A study reporting a gamma can be used in a lognormal meta model. Parameters are never compared across families, which would be meaningless. For a two parameter family the map from parameters to a mean and a standard deviation is one to one, so a study that fitted correctly loses nothing by being converted.

What this does assume is that summaries of the study's reported distribution are the summaries of what its procedure targeted. That holds exactly when the study reported a distribution of the same shape as its estimand, and approximately otherwise, for example where a study fitted a continuous distribution to integer date differences.

The range the summaries are taken over

A study that did not correct for right truncation fitted a distribution to delays that stop at its observation time, but the families studies fit have a tail running past that point. Taking the reported distribution's summaries over its whole support would then charge it with spread its data never had, which is worst for a standard deviation and can reach tens of percent for a short observation time.

Summaries are therefore taken over the range of delays the study could have seen, conditioning the reported distribution on falling between `delay_min` and `relative_obs_time`. This is the same range the meta model computes its implied summaries over, so the two sides are the same functional of the two distributions. Both fields are read from the metadata passed through `...`, and a study that adjusted for right truncation is left unconditioned above. A reported quantile at probability p is the value the conditioned distribution puts p below, so it too stays inside the observed range.

Where the study fitted a family that cannot represent its own estimand, quantiles inside the body of the distribution are more reliable than a standard deviation, which depends on a tail the study never saw.

Uncertainty

Supply the standard errors the study reported on its parameters through `se`. They are carried onto the summary scale by the delta method, using a numerical Jacobian of the map from parameters to summaries, and the summaries are reported with the covariance matrix that implies between them, so they are fitted jointly as a multivariate normal in the same way as the output of `as_epidist_multivariate()`. For a fit with as many summaries as parameters this carries the study's information about its parameters exactly, which fitting each summary with its own standard error would not.

Summaries of a two parameter fit are deterministic functions of two numbers, so at most two of them can carry the reported uncertainty, and asking for more with `se` is an error. A study that published a full parameter covariance rather than standard errors can draw from it, push each draw through to the summaries, and use `as_epidist_multivariate()`, which needs no linearisation.

Without se the summaries fall back to the sample size likelihoods, which derive their sampling uncertainty from n, and any number of them may be reported.

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [as_epidist_estimates_data.list\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview.epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#), [new_epidist_estimates_data\(\)](#)

Examples

```
epidist_estimates_parameters(
  "study A",
  family = "gamma",
  parameters = c(shape = 4.1, rate = 0.55),
  se = c(0.4, 0.06),
  relative_obs_time = 20,
  trunc_adjusted = FALSE,
  cens_adjusted = 0
)
```

```
epidist_estimates_summaries
```

Report a study that published summaries of its delays

Description

Builds the rows [as_epidist_meta_model\(\)](#) takes from the summaries a study published in wide form, one argument per kind of summary.

Usage

```
epidist_estimates_summaries(
  study,
  mean = NULL,
  sd = NULL,
  quantiles = NULL,
  probs = NULL,
  se = NULL,
  n = NULL,
  ...
)
```

Arguments

<code>study</code>	A string naming the study.
<code>mean</code>	The reported mean delay. Optional.
<code>sd</code>	The reported standard deviation of the delays. Optional.
<code>quantiles</code>	A numeric vector of reported quantiles. Optional.
<code>probs</code>	The probabilities of quantiles, in the same order. Required where quantiles is given.
<code>se</code>	A numeric vector of the reported standard errors of the summaries, ordered mean, standard deviation, then quantiles, skipping any that was not reported. Optional.
<code>n</code>	The number of delays the study summarised. Optional.
<code>...</code>	Study metadata, as documented in <code>as_epidist_estimates_data.data.frame()</code> .

Details

Give the uncertainty of each summary through `se`, or the number of delays the study summarised through `n`, which the model uses to derive a sampling uncertainty instead. One of the two is needed for every row.

Value

An `epidist_estimates_data` object.

See Also

Other `estimates_data`: `as_epidist_estimates_data()`, `as_epidist_estimates_data.data.frame()`, `as_epidist_estimates_data.epidist_estimates_data()`, `as_epidist_estimates_data.epidist_multivariate()`, `as_epidist_estimates_data.list()`, `assert_epidist.epidist_estimates_data()`, `epidist_estimates_epireview`, `epidist_estimates_parameters()`, `is_epidist_estimates_data()`, `new_epidist_estimates_data()`

Examples

```
epidist_estimates_summaries(
  "study A",
  mean = 7.5, sd = 3.6, n = 120,
  relative_obs_time = 20, trunc_adjusted = FALSE, cens_adjusted = 0
)
```

epidist_family	<i>Define epidist family</i>
----------------	------------------------------

Description

This function is used within `epidist()` to create a model specific custom brms family object. This custom family is passed to brms. It is unlikely that as a user you will need this function, but we export it nonetheless to be transparent about what happens inside of a call to `epidist()`.

Usage

```
epidist_family(data, family = lognormal(), ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see <code>brms::brmsfamily()</code> . Commonly used, such as <code>brms::lognormal()</code> , are also reexported as part of <code>epidist</code> .
<code>...</code>	Additional arguments passed to fn method.

Details

The family may be any brms family of a positive response, such as `brms::lognormal()`, `Gamma(link = "log")` or `brms::weibull()`, or a family `epidist` defines itself, such as `gengamma()`.

Value

A brms custom family object.

See Also

Other family: `epidist_family_model()`, `epidist_family_model.default()`, `epidist_family_param()`, `epidist_family_param.default()`, `epidist_family_param.gengamma()`, `gengamma()`

Examples

```
data <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model()
family <- epidist_family(data, family = lognormal())
family
```

`epidist_family_model` *The model-specific parts of an `epidist_family()` call*

Description

The model-specific parts of an `epidist_family()` call

Usage

```
epidist_family_model(data, family, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	Output of a call to <code>brms::brmsfamily()</code> with additional information as provided by <code>.add_dpar_info()</code>
<code>...</code>	Additional arguments passed to <code>fn</code> method.

Value

A brms custom family object.

See Also

Other family: [epidist_family\(\)](#), [epidist_family_model.default\(\)](#), [epidist_family_param\(\)](#), [epidist_family_param.default\(\)](#), [epidist_family_param.gengamma\(\)](#), [gengamma\(\)](#)

`epidist_family_model.default`

Default method for defining a model specific family

Description

Default method for defining a model specific family

Usage

```
## Default S3 method:
epidist_family_model(data, family, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	Output of a call to <code>brms::brmsfamily()</code> with additional information as provided by <code>.add_dpar_info()</code>
<code>...</code>	Additional arguments passed to <code>fn</code> method.

Value

A brms custom family object.

See Also

Other family: [epidist_family\(\)](#), [epidist_family_model\(\)](#), [epidist_family_param\(\)](#), [epidist_family_param.default](#), [epidist_family_param.gengamma\(\)](#), [gengamma\(\)](#)

```
epidist_family_model.epidist_latent_model
```

Create the model-specific component of an epidist custom family

Description

Create the model-specific component of an epidist custom family

Usage

```
## S3 method for class 'epidist_latent_model'
epidist_family_model(data, family, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	Output of a call to <code>brms::brmsfamily()</code> with additional information as provided by <code>.add_dpar_info()</code>
<code>...</code>	Additional arguments passed to method.

Value

A brms custom family object.

See Also

Other latent_model: [as_epidist_latent_model\(\)](#), [as_epidist_latent_model.epidist_aggregate_data\(\)](#), [as_epidist_latent_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_latent_model\(\)](#), [epidist_model_prior.epidist_latent_model\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [is_epidist_latent_model\(\)](#), [new_epidist_latent_model\(\)](#)

```
epidist_family_model.epidist_marginal_model
```

Create the model-specific component of an epidist custom family

Description

Create the model-specific component of an epidist custom family

Usage

```
## S3 method for class 'epidist_marginal_model'
epidist_family_model(data, family, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	Output of a call to <code>brms::brmsfamily()</code> with additional information as provided by <code>.add_dpar_info()</code>
<code>...</code>	Additional arguments passed to method.

Value

A brms custom family object.

See Also

Other marginal_model: [as_epidist_marginal_model\(\)](#), [as_epidist_marginal_model.epidist_aggregate_data\(\)](#), [as_epidist_marginal_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_marginal_model\(\)](#), [epidist_newdata.epidist_marginal_model\(\)](#), [epidist_transform_data_model.epidist_marginal_model\(\)](#), [is_epidist_marginal_model\(\)](#), [new_epidist_marginal_model\(\)](#)

```
epidist_family_model.epidist_meta_model
```

Create the model-specific component of an epidist custom family

Description

Create the model-specific component of an epidist custom family

Usage

```
## S3 method for class 'epidist_meta_model'
epidist_family_model(data, family, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	Output of a call to <code>brms::brmsfamily()</code> with additional information as provided by <code>.add_dpar_info()</code>
<code>...</code>	Additional arguments passed to method.

Value

A brms custom family object.

See Also

Other meta_model: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_a`, `as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`, `epidist_formula_model.epidist_meta_model()`, `epidist_meta_leave_or`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_model()`, `epidist_transform_data`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

`epidist_family_param.default`

Default method for families which do not require a reparameterisation

Description

This function extracts the Stan parameterisation for a given brms family by creating a dummy model and parsing its Stan code. It looks for the log probability density function (lpdf) call in the Stan code and extracts the parameter order used by Stan. This is needed because brms and Stan may use different parameter orderings for the same distribution.

Usage

```
## Default S3 method:
epidist_family_param(family, ...)
```

Arguments

<code>family</code>	A brms family object containing at minimum a family element specifying the distribution family name.
<code>...</code>	Additional arguments passed to methods (not used)

Details

The function works by:

1. Creating a minimal dummy model using the specified family
2. Extracting the Stan code for this model
3. Finding the lpdf function call for the family
4. Parsing out the parameter ordering used in Stan
5. Adding this as the param element to the family object

Value

The input family object with an additional param element containing the Stan parameter ordering as a string

See Also

Other family: [epidist_family\(\)](#), [epidist_family_model\(\)](#), [epidist_family_model.default\(\)](#), [epidist_family_param\(\)](#), [epidist_family_param.gengamma\(\)](#), [gengamma\(\)](#)

`epidist_family_param.gengamma`

Method for the [gengamma\(\)](#) family

Description

The latent and naive models call the Stan density in the order brms declares the parameters. The marginal and meta models pass pcd_param, the Stacy form primarycensored takes.

Usage

```
## S3 method for class 'gengamma'
epidist_family_param(family, ...)
```

Arguments

family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see brms::brmsfamily() . Commonly used, such as brms::lognormal() , are also reexported as part of epidist.
...	Additional arguments passed to fn method.

Value

The family with param and pcd_param elements.

See Also

Other family: `epidist_family()`, `epidist_family_model()`, `epidist_family_model.default()`, `epidist_family_param()`, `epidist_family_param.default()`, `gengamma()`

`epidist_family_prior` *Family specific prior distributions*

Description

This function contains brms prior distributions which are specific to particular likelihood families e.g. `brms::lognormal()`.

Usage

```
epidist_family_prior(family, ...)
```

Arguments

<code>family</code>	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see <code>brms::brmsfamily()</code> . Commonly used, such as <code>brms::lognormal()</code> , are also reexported as part of <code>epidist</code> .
<code>...</code>	Additional arguments passed to fn method.

Value

A `brmsprior` object, or `NULL` when the model adds no priors.

See Also

Other prior: `epidist_family_prior.default()`, `epidist_family_prior.gengamma()`, `epidist_family_prior.lognormal()`, `epidist_model_prior()`, `epidist_model_prior.default()`, `epidist_model_prior.epidist_meta_model()`, `epidist_prior()`

epidist_family_prior.default

Default family specific prior distributions

Description

By default, we do not return any family specific prior distributions.

Usage

```
## Default S3 method:
epidist_family_prior(family, formula, ...)
```

Arguments

family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see brms::brmsfamily() . Commonly used, such as brms::lognormal() , are also reexported as part of epidist.
formula	An object of class stats::formula or brms::brmsformula (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to fn method.

Value

A brmsprior object, or NULL when the model adds no priors.

See Also

Other prior: [epidist_family_prior\(\)](#), [epidist_family_prior.gengamma\(\)](#), [epidist_family_prior.lognormal\(\)](#), [epidist_model_prior\(\)](#), [epidist_model_prior.default\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_prior\(\)](#)

epidist_family_prior.gengamma

Family specific prior distributions for the generalised gamma family

Description

mu and sigma get the intercept priors of the lognormal family. The intercept of Q, on the log scale, gets `normal(0, 0.5)`, centred on the Weibull case.

Usage

```
## S3 method for class 'gengamma'
epidist_family_prior(family, formula, ...)
```

Arguments

family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see <code>brms::brmsfamily()</code> . Commonly used, such as <code>brms::lognormal()</code> , are also reexported as part of <code>epidist</code> .
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to fn method.

Value

A `brmsprior` object.

See Also

Other prior: `epidist_family_prior()`, `epidist_family_prior.default()`, `epidist_family_prior.lognormal()`, `epidist_model_prior()`, `epidist_model_prior.default()`, `epidist_model_prior.epidist_meta_model()`, `epidist_prior()`

epidist_family_prior.lognormal

Family specific prior distributions for the lognormal family

Description

We suggest priors to overwrite the brms defaults for the lognormal family.

Usage

```
## S3 method for class 'lognormal'
epidist_family_prior(family, formula, ...)
```

Arguments

family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see brms::brmsfamily() . Commonly used, such as brms::lognormal() , are also reexported as part of epidist.
formula	An object of class stats::formula or brms::brmsformula (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to fn method.

Value

A brmsprior object, or NULL when the model adds no priors.

See Also

Other prior: [epidist_family_prior\(\)](#), [epidist_family_prior.default\(\)](#), [epidist_family_prior.gengamma\(\)](#), [epidist_model_prior\(\)](#), [epidist_model_prior.default\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_prior\(\)](#)

epidist_formula	<i>Define a model specific formula</i>
-----------------	--

Description

This function is used within `epidist()` to create the formula object passed to brms. It is unlikely that as a user you will need this function, but we export it nonetheless to be transparent about what exactly is happening inside of a call to `epidist()`.

Usage

```
epidist_formula(data, family, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
family	A description of the response distribution and link function to be used in the model created using <code>epidist_family()</code> .
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to fn method.

Value

A brmsformula object.

See Also

Other formula: `epidist_formula_model()`, `epidist_formula_model.default()`

Examples

```
data <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model()
family <- epidist_family(data, family = lognormal())
formula <- epidist_formula(data, family = family, formula = mu ~ 1)
formula
```

`epidist_formula_model` *The model-specific parts of an `epidist_formula()` call*

Description

The model-specific parts of an `epidist_formula()` call

Usage

```
epidist_formula_model(data, formula, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>formula</code>	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
<code>...</code>	Additional arguments passed to <code>fn</code> method.

Value

A `brmsformula` object.

See Also

Other formula: `epidist_formula()`, `epidist_formula_model.default()`

`epidist_formula_model.default`
Default method for defining a model specific formula

Description

Default method for defining a model specific formula

Usage

```
## Default S3 method:  
epidist_formula_model(data, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to fn method.

Value

A brmsformula object.

See Also

Other formula: `epidist_formula()`, `epidist_formula_model()`

```
epidist_formula_model.epidist_latent_model
```

Define the model-specific component of an epidist custom formula for the latent model

Description

Define the model-specific component of an epidist custom formula for the latent model

Usage

```
## S3 method for class 'epidist_latent_model'
epidist_formula_model(data, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to method.

Value

A brmsformula object.

See Also

Other latent_model: `as_epidist_latent_model()`, `as_epidist_latent_model.epidist_aggregate_data()`, `as_epidist_latent_model.epidist_linelist_data()`, `epidist_family_model.epidist_latent_model()`, `epidist_model_prior.epidist_latent_model()`, `epidist_newdata.epidist_latent_model()`, `is_epidist_latent_model()`, `new_epidist_latent_model()`

```
epidist_formula_model.epidist_marginal_model
```

Define the model-specific component of an epidist custom formula for the marginal model

Description

Define the model-specific component of an epidist custom formula for the marginal model

Usage

```
## S3 method for class 'epidist_marginal_model'
epidist_formula_model(data, formula, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>formula</code>	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
<code>...</code>	Additional arguments passed to method.

Value

A `brmsformula` object.

See Also

Other marginal_model: `as_epidist_marginal_model()`, `as_epidist_marginal_model.epidist_aggregate_data()`, `as_epidist_marginal_model.epidist_linelist_data()`, `epidist_family_model.epidist_marginal_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`, `new_epidist_marginal_model()`

```
epidist_formula_model.epidist_meta_model
```

Define the model-specific component of an epidist custom formula for the meta model

Description

Adds the response and the slots of the meta model to the formula. Where a summary row estimates its growth rate as the pgrowth distributional parameter, see [as_epidist_meta_model\(\)](#), and the formula has no term for pgrowth, $\text{pgrowth} \sim 0 + \text{study}$ is added so that each study has its own rate and the priors [epidist_model_prior\(\)](#) builds from the reported rates have a coefficient to act on. Data holding a single study get $\text{pgrowth} \sim 1$ instead, because brms cannot code a factor with one level. A pgrowth formula given by the user is kept as it is.

Usage

```
## S3 method for class 'epidist_meta_model'
epidist_formula_model(data, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class stats::formula or brms::brmsformula (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter mu, and may optionally be provided for other distributional parameters.
...	Additional arguments passed to method.

Value

A brmsformula object.

See Also

Other meta_model: [as_epidist_meta_model\(\)](#), [as_epidist_meta_model.NULL\(\)](#), [as_epidist_meta_model.epidist_a](#), [as_epidist_meta_model.epidist_estimates_data\(\)](#), [as_epidist_meta_model.epidist_linelist_data\(\)](#), [assert_epidist.epidist_meta_model\(\)](#), [epidist_family_model.epidist_meta_model\(\)](#), [epidist_meta_leave_one](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_newdata.epidist_meta_model\(\)](#), [epidist_transform_data](#), [is_epidist_meta_model\(\)](#), [new_epidist_meta_model\(\)](#)

```
epidist_formula_model.epidist_naive_model
```

Define the model-specific component of an epidist custom formula for the naive model

Description

Define the model-specific component of an epidist custom formula for the naive model

Usage

```
## S3 method for class 'epidist_naive_model'
epidist_formula_model(data, formula, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>formula</code>	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
<code>...</code>	Additional arguments passed to method.

Value

A `brmsformula` object.

See Also

Other `naive_model`: `as_epidist_naive_model()`, `as_epidist_naive_model.epidist_aggregate_data()`, `as_epidist_naive_model.epidist_linelist_data()`, `epidist_newdata.epidist_naive_model()`, `epidist_stancode.epidist_naive_model()`, `epidist_transform_data_model.epidist_naive_model()`, `is_epidist_naive_model()`, `new_epidist_naive_model()`

<code>epidist_gen_log_lik</code>	<i>Create a function to calculate the marginalised log likelihood for double censored and truncated delay distributions</i>
----------------------------------	---

Description

This function creates a log likelihood function that calculates the marginal likelihood for a single observation by integrating over the latent primary and secondary event windows. Where analytical solutions exist in `primarycensored::dpcens()` these are used, otherwise the integration is performed numerically. `primarycensored::dpcens()` handles the double censoring and truncation of the delay distribution.

Usage

```
epidist_gen_log_lik(family)
```

Arguments

family A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see [brms::brmsfamily\(\)](#). Commonly used, such as [brms::lognormal\(\)](#), are also reexported as part of epidist.

Details

The marginal likelihood accounts for uncertainty in both the primary and secondary event windows by integrating over their possible values, weighted by their respective uniform distributions.

Value

A function that calculates the marginal log likelihood for a single observation. The prep object must have the following variables:

- `vreal1`: relative observation time
- `vreal2`: primary event window
- `vreal3`: secondary event window
- `vreal4`: delay upper bound
- `vreal5`: minimum delay (left truncation point; defaults to 0 if absent)

See Also

[brms::log_lik\(\)](#) for details on the brms log likelihood interface.

Other gen: [epidist_gen_meta_log_lik\(\)](#), [epidist_gen_meta_predict\(\)](#), [epidist_gen_posterior_epred\(\)](#), [epidist_gen_posterior_predict\(\)](#)

```
epidist_gen_meta_log_lik
```

Create a function to calculate the meta model log likelihood

Description

Individual level rows use the marginal model log likelihood created by [epidist_gen_log_lik\(\)](#). Summary rows use the sampling distributions described in [as_epidist_meta_model\(\)](#), evaluated at the implied summaries for each posterior draw. A row that stands for several summaries reported by one study returns the joint log likelihood of all of them, so an observation here is a group of summaries rather than a single reported value.

Usage

```
epidist_gen_meta_log_lik(family)
```

Arguments

family A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see `brms::brmsfamily()`. Commonly used, such as `brms::lognormal()`, are also reexported as part of `epidist`.

Details

Summary rows are evaluated in R, one posterior draw at a time, and each evaluation can run a quadrature of up to 2000 intervals by default. The cost of `brms::log_lik()` and `loo::loo()` therefore grows with the number of draws times the number of summary rows. Pass `ndraws` to either to use fewer draws, at the price of noisier estimates.

Value

A function that calculates the log likelihood for a single observation. The prep object must have the meta model `vint` and `vreal` slots.

See Also

`brms::log_lik()` for details on the brms log likelihood interface.

Other gen: `epidist_gen_log_lik()`, `epidist_gen_meta_predict()`, `epidist_gen_posterior_epred()`, `epidist_gen_posterior_predict()`

```
epidist_gen_meta_predict
```

Create a function to draw from the meta model posterior predictive distribution

Description

Individual level rows are predicted as in the marginal model using `epidist_gen_posterior_predict()`. For summary rows the predicted quantity is the reported summary itself, that is a simulated reported mean, standard deviation, or, for quantile rows, cumulative probability at the reported value. A row that stands for several summaries reported by one study predicts the first of them, drawn from its marginal. Predictions for summary rows are therefore not on the delay scale and should not be compared directly with individual level predictions. They come from the normal approximations described in `as_epidist_meta_model()`, so a predicted cumulative probability for a quantile row can fall outside $[0, 1]$ when the study sample size is small.

Usage

```
epidist_gen_meta_predict(family)
```

Arguments

family A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see `brms::brmsfamily()`. Commonly used, such as `brms::lognormal()`, are also reexported as part of `epidist`.

Value

A function that takes a prep argument from `brms` and returns a matrix of posterior predictions.

See Also

`brms::posterior_predict()` for details on how this is used within `brms`.

Other gen: `epidist_gen_log_lik()`, `epidist_gen_meta_log_lik()`, `epidist_gen_posterior_epred()`, `epidist_gen_posterior_predict()`

```
epidist_gen_posterior_epred
```

Create a function to draw from the expected value of the posterior predictive distribution for a model

Description

This function creates a function that calculates the expected value of the posterior predictive distribution for a latent model. The returned function takes a prep argument (from `brms`) and returns posterior expected values. This is used internally by `brms::posterior_epred()` to calculate expected values for latent models.

Usage

```
epidist_gen_posterior_epred(family)
```

Arguments

family A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see `brms::brmsfamily()`. Commonly used, such as `brms::lognormal()`, are also reexported as part of `epidist`.

Value

A function that takes a prep argument from brms and returns a matrix of posterior expected values, with one row per posterior draw and one column per observation.

See Also

`brms::posterior_epred()` for details on how this is used within brms.

Other gen: `epidist_gen_log_lik()`, `epidist_gen_meta_log_lik()`, `epidist_gen_meta_predict()`, `epidist_gen_posterior_predict()`

epidist_gen_posterior_predict

Create a function to draw from the posterior predictive distribution for a double censored and truncated delay distribution

Description

This function creates a function that draws from the posterior predictive distribution for a latent model using `primarycensored::rpcens()` to handle censoring and truncation. The returned function takes a prep argument from brms and returns posterior predictions. This is used internally by `brms::posterior_predict()` to generate predictions for latent models.

Usage

```
epidist_gen_posterior_predict(family)
```

Arguments

family	A description of the response distribution and link function to be used in the model. Every family function has a link argument allowing users to specify the link function to be applied on the response variable. If not specified, default links are used. For details of all supported families see <code>brms::brmsfamily()</code> . Commonly used, such as <code>brms::lognormal()</code> , are also reexported as part of epidist.
--------	---

Value

A function that takes a prep argument from brms and returns a matrix of posterior predictions, with one row per posterior draw and one column per observation. The prep object must have the following variables:

- vreal1: relative observation time
- vreal2: primary event window
- vreal3: secondary event window
- vreal4: delay upper bound
- vreal5: minimum delay (left truncation point; defaults to 0 if absent)

See Also

`brms::posterior_predict()` for details on how this is used within brms, `primarycensored::rpcens()` for details on the censoring approach

Other gen: `epidist_gen_log_lik()`, `epidist_gen_meta_log_lik()`, `epidist_gen_meta_predict()`, `epidist_gen_posterior_epred()`

epidist_meta_leave_one_out

Refit a meta model leaving each study out in turn

Description

Refits a meta model once per study with that study removed, and compares the delay mean and standard deviation of each refit with the full fit.

Usage

```
epidist_meta_leave_one_out(
  fit,
  data = NULL,
  newdata = NULL,
  re_formula = NA,
  width = 0.95,
  keep_fits = FALSE,
  studies = NULL,
  cores = NULL,
  ...
)
```

Arguments

<code>fit</code>	A meta model fitted with <code>epidist()</code> to an <code>epidist_meta_model</code> object.
<code>data</code>	The <code>epidist_meta_model</code> object <code>fit</code> was fitted to. Only needed when <code>fit\$data</code> has no study column, which happens when the formula does not use study. If <code>NULL</code> , the default, the model data stored in <code>fit</code> is used.
<code>newdata</code>	A <code>data.frame</code> of data to predict the delay for, passed to <code>delay_summary_draws()</code> . If <code>NULL</code> , the default, a single row giving the population level delay with no censoring and no truncation is built with <code>epidist_newdata()</code> .
<code>re_formula</code>	Passed to <code>brms::prepare_predictions()</code> through <code>delay_summary_draws()</code> . The default <code>NA</code> switches off any study level term, so the summaries are of the population level delay.
<code>width</code>	The width of the central posterior intervals. Defaults to 0.95.
<code>keep_fits</code>	If <code>TRUE</code> , the refits are returned in the <code>fits</code> attribute of the result as a list named by the held out study. Defaults to <code>FALSE</code> .

studies	A character vector of the studies to hold out, one refit each. If NULL, the default, every study is held out in turn. Holding out a few studies, such as the largest, keeps the cost down when a model has many.
cores	The number of cores each refit uses. If NULL, the default, the <code>mc.cores</code> option where it is set, and otherwise one core per chain, so the chains of each refit run in parallel.
...	Additional arguments passed to <code>brms::update.brmsfit()</code> and so to <code>brms::brm()</code> , such as <code>chains</code> , <code>iter</code> , <code>refresh</code> and <code>silent</code> .

Details

Individual level rows are held out together as the "individual" study. If the formula does not use study, pass the `epidist_meta_model` object as data, because `brms` drops unused variables from `fit$data`. Each refit reuses the compiled model through `brms::update.brmsfit()`. The refits are exact because Pareto smoothed importance sampling (Vehtari et al. 2017) is often unreliable when a whole study is left out.

The comparison is reported as estimates with intervals and no threshold, as in `marginaleffects` (Arel-Bundock et al. 2024). `difference` is the refit posterior minus the full fit's posterior median. The two fits share most of their data, so their draws are not differenced.

Value

A tibble with one row per held out study, row of `newdata` and summary ("mean" or "sd"). It gives the posterior median (estimate) and interval (lower, upper) of the refit and of the full fit (`full_*`), and the difference from the full fit with its interval. Predictors of the model in `newdata` other than study are included. The `width` attribute records the interval width.

References

- Vehtari et al. (2017) [doi:10.1007/s1122201696964](https://doi.org/10.1007/s1122201696964)
- Arel-Bundock et al. (2024) [doi:10.18637/jss.v111.i09](https://doi.org/10.18637/jss.v111.i09)

See Also

`delay_summary_draws()` for the summaries the comparison uses and `epidist_newdata()` to build `newdata`.

Other `meta_model`: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`, `as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_model()`, `epidist_transform_data.epidist_meta_model()`, `new_epidist_meta_model()`

Examples

```
estimates <- as_epidist_estimates_data(
  data.frame(
    study = c("A", "A", "B", "B", "C", "C"),
    type = c("mean", "sd", "mean", "sd", "mean", "sd"),
```

```

    value = c(7.5, 3.6, 6.4, 3.1, 8.2, 4.0),
    n = c(120, 120, 80, 80, 150, 150),
    relative_obs_time = c(20, 20, 25, 25, Inf, Inf),
    trunc_adjusted = c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE),
    cens_adjusted = c(0, 0, 0, 0, 1, 1)
  )
)
meta <- as_epidist_meta_model(estimates = estimates)
fit <- epidist(
  meta,
  chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0)
)

# The formula does not use study, so pass the model data
epidist_meta_leave_one_out(fit, data = meta, refresh = 0)

```

epidist_model_prior *Model specific prior distributions*

Description

This function contains brms prior distributions which are specific to particular epidist models e.g. the latent_lognormal model.

Usage

```
epidist_model_prior(data, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
...	Additional arguments passed to fn method.

Value

A brmsprior object, or NULL when the model adds no priors.

See Also

Other prior: [epidist_family_prior\(\)](#), [epidist_family_prior.default\(\)](#), [epidist_family_prior.gengamma\(\)](#), [epidist_family_prior.lognormal\(\)](#), [epidist_model_prior.default\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_prior\(\)](#)

```
epidist_model_prior.default
```

Default model specific prior distributions

Description

By default, we do not return any model specific prior distributions.

Usage

```
## Default S3 method:
epidist_model_prior(data, formula, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>formula</code>	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
<code>...</code>	Additional arguments passed to <code>fn</code> method.

Value

A `brmsprior` object, or `NULL` when the model adds no priors.

See Also

Other prior: `epidist_family_prior()`, `epidist_family_prior.default()`, `epidist_family_prior.gengamma()`, `epidist_family_prior.lognormal()`, `epidist_model_prior()`, `epidist_model_prior.epidist_meta_model()`, `epidist_prior()`

```
epidist_model_prior.epidist_latent_model
```

Model specific prior distributions for latent models

Description

Defines prior distributions for the latent model parameters `pwindow_raw` and `swindow_raw` which control the width of the observation windows.

Usage

```
## S3 method for class 'epidist_latent_model'
epidist_model_prior(data, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
...	Additional arguments passed to <code>fn</code> method.

Value

A `brmsprior` object, or `NULL` when the model adds no priors.

See Also

Other `latent_model`: `as_epidist_latent_model()`, `as_epidist_latent_model.epidist_aggregate_data()`, `as_epidist_latent_model.epidist_linelist_data()`, `epidist_family_model.epidist_latent_model()`, `epidist_formula_model.epidist_latent_model()`, `epidist_newdata.epidist_latent_model()`, `is_epidist_latent_model()`, `new_epidist_latent_model()`

`epidist_model_prior.epidist_meta_model`

Model specific prior distributions for the meta model

Description

The response column of a meta model is a placeholder on every summary row, and `brms` centres its default prior for the intercept of `mu` on the response. For a model fitted to summaries alone that default is centred on a delay of zero. This method puts a `normal(1, 1)` prior on the intercept instead, the scale of the lognormal family prior in `epidist_family_prior()`, so that a Gamma or Weibull meta fit gets a prior on the same scale as a lognormal one. On the log scale it is a median delay of about 3 days with a 95% range of roughly 0.4 to 20 days.

Usage

```
## S3 method for class 'epidist_meta_model'
epidist_model_prior(data, formula, default = NULL, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
formula	An object of class <code>stats::formula</code> or <code>brms::brmsformula</code> (or one that can be coerced to those classes). A symbolic description of the model to be fitted. A formula must be provided for the distributional parameter <code>mu</code> , and may optionally be provided for other distributional parameters.
default	The default prior distributions from <code>brms::default_prior()</code> , which <code>epidist_prior()</code> passes so that they are not built twice. Built here where missing.
...	Additional arguments passed to <code>fn</code> method.

Details

The centre is fixed rather than taken from the reported values, because a prior chosen from the data is not a prior. It would put the posterior of a small review where the data already sit and understate how much the studies disagree. The prior is added where μ is on the log scale, which is the lognormal family, whose μ is the log of the median under an identity link, and any family with a log link. Nothing is added for other links, and a model with individual level rows only adds no prior, so the family or brms default applies as it does for the marginal model.

The between study spread of any group level term, such as $(1 | \text{study})$, gets a half normal prior with a standard deviation of 0.25 on the scale of the linear predictor, so that a small review cannot fit that spread from almost nothing under the wide brms default. It is dropped where the formula has no group level term. The prior on the intercept of the other distributional parameters is left to the family or to brms.

Where a summary row estimates its growth rate as the pgrowth distributional parameter, see `as_epidist_meta_model()`, the coefficients and intercept of pgrowth get a $\text{normal}(0, 0.25)$ prior, which is weakly informative for a delay measured in days, because the brms default is flat and the summaries carry little information about the rate. A study that reported its rate with a `growth_rate_sd` gets a normal prior with that centre and spread on its own coefficient, which exists under the default $\text{pgrowth} \sim 0 + \text{study}$ formula, or on the intercept of pgrowth where it is the only study. Under another pgrowth formula the reported rates have no coefficient to act on and are dropped with a warning, so set their priors yourself.

Value

A brmsprior object, or NULL when the model adds no priors.

See Also

Other prior: `epidist_family_prior()`, `epidist_family_prior.default()`, `epidist_family_prior.gengamma()`, `epidist_family_prior.lognormal()`, `epidist_model_prior()`, `epidist_model_prior.default()`, `epidist_prior()`

Other meta_model: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`, `as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_model()`, `epidist_meta_leave_one_out()`, `epidist_newdata.epidist_meta_model()`, `epidist_transform_data_model.epidist_meta_model()`, `new_epidist_meta_model()`

epidist_newdata

Build newdata for prediction from an epidist data object

Description

Prediction with brms and tidybayes needs a newdata argument holding every variable the model uses. As well as the variables in your formula that means the response and the observation process variables that epidist adds. Their names depend on the model, so building newdata by hand means knowing how each model is specified. This function builds it for you. It expands the variables you supply into a grid, in the same way as `tidyr::expand()`, and adds the rest with values you set through named arguments.

Usage

```
epidist_newdata(data, ...)
```

Arguments

data	An epidist data object, such as one returned by as_epidist_marginal_model() , as_epidist_latent_model() , as_epidist_naive_model() or as_epidist_meta_model() .
...	Variables to expand into a grid, passed to tidyr::expand() . Supply the variables used in the model formula, such as sex. Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.

Details

The defaults give the delay distribution with no censoring and no truncation. Set `pwindow` and `swindow` to ask for censoring, and `relative_obs_time` to ask for truncation. See the method for each model for the arguments it takes.

The result is a plain [tibble::tibble\(\)](#). Pass it to [brms::posterior_epred\(\)](#), [brms::posterior_predict\(\)](#), [add_delay_parameter_draws\(\)](#) or [delay_parameter_draws\(\)](#), or to the tidybayes functions [add_epred_draws\(\)](#) and [add_predicted_draws\(\)](#).

Value

A [tibble::tibble\(\)](#) of newdata ready to predict from.

See Also

Other newdata: [epidist_newdata.default\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [epidist_newdata.epidist_meta_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#)

Examples

```
prep_obs <- sierra_leone Ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_marginal_model()

# An intercept only model
epidist_newdata(prepare_obs)

# A row for each sex
epidist_newdata(prepare_obs, sex)
```

epidist_newdata.default

Default method for building newdata

Description

Default method for building newdata

Usage

```
## Default S3 method:
epidist_newdata(data, ...)
```

Arguments

data	An epidist data object, such as one returned by as_epidist_marginal_model() , as_epidist_latent_model() , as_epidist_naive_model() or as_epidist_meta_model() .
...	Variables to expand into a grid, passed to tidyr::expand() . Supply the variables used in the model formula, such as sex. Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.

Value

This method errors. It is called when data is not an epidist model data object.

See Also

Other newdata: [epidist_newdata\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [epidist_newdata.epidist_marginal_model\(\)](#), [epidist_newdata.epidist_meta_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#)

epidist_newdata.epidist_latent_model

Build newdata for the latent model

Description

The latent model uses the response, the observation time relative to the primary event, and the primary and secondary censoring windows. This method adds all of them. The response is set to NA because it is the quantity being predicted. The rest are set here.

Usage

```
## S3 method for class 'epidist_latent_model'
epidist_newdata(data, ..., pwindow = 0, swindow = 0, relative_obs_time = Inf)
```

Arguments

<code>data</code>	An epidist data object, such as one returned by <code>as_epidist_marginal_model()</code> , <code>as_epidist_latent_model()</code> , <code>as_epidist_naive_model()</code> or <code>as_epidist_meta_model()</code> .
<code>...</code>	Variables to expand into a grid, passed to <code>tidyr::expand()</code> . Supply the variables used in the model formula, such as <code>sex</code> . Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.
<code>pwindow</code>	Width of the primary event censoring window. Defaults to 0, which is no censoring.
<code>swindow</code>	Width of the secondary event censoring window. Defaults to 0, which is no censoring.
<code>relative_obs_time</code>	Observation time relative to the primary event. Defaults to <code>Inf</code> , which is no right truncation. <code>brms</code> warns about infinite values in the data when this is <code>Inf</code> . That warning is safe here, because prediction is done in R and never passes the value to Stan.

Details

The defaults give the delay distribution with no censoring and no truncation, which is a continuous probability density function. For a discrete probability mass function with daily censoring set `pwindow` and `swindow` to 1. For the delay distribution as it would be seen at a given time set `relative_obs_time` to that time relative to the primary event.

Value

A `tibble::tibble()` of newdata ready to predict from.

See Also

Other latent_model: `as_epidist_latent_model()`, `as_epidist_latent_model.epidist_aggregate_data()`, `as_epidist_latent_model.epidist_linelist_data()`, `epidist_family_model.epidist_latent_model()`, `epidist_formula_model.epidist_latent_model()`, `epidist_model_prior.epidist_latent_model()`, `is_epidist_latent_model()`, `new_epidist_latent_model()`

Other newdata: `epidist_newdata()`, `epidist_newdata.default()`, `epidist_newdata.epidist_marginal_model()`, `epidist_newdata.epidist_meta_model()`, `epidist_newdata.epidist_naive_model()`

Examples

```
prep_obs <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_latent_model()

# A row for each sex, with no censoring and no truncation
```

```
epidist_newdata(prepare_obs, sex)

# The same, with daily censoring
epidist_newdata(prepare_obs, sex, pwindow = 1, swindow = 1)
```

```
epidist_newdata.epidist_marginal_model
```

Build newdata for the marginal model

Description

The marginal model uses the response, the observation time relative to the primary event, the primary and secondary censoring windows, the upper bound of the delay and the minimum delay. This method adds all of them. The response and its upper bound are set to 0. `brms` needs the response in `newdata` and needs it to be a whole number for this model, but prediction ignores its value. The rest are set here.

Usage

```
## S3 method for class 'epidist_marginal_model'
epidist_newdata(
  data,
  ...,
  pwindow = 0,
  swindow = 0,
  relative_obs_time = Inf,
  delay_min = 0
)
```

Arguments

<code>data</code>	An epidist data object, such as one returned by as_epidist_marginal_model() , as_epidist_latent_model() , as_epidist_naive_model() or as_epidist_meta_model() .
<code>...</code>	Variables to expand into a grid, passed to tidyr::expand() . Supply the variables used in the model formula, such as <code>sex</code> . Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.
<code>pwindow</code>	Width of the primary event censoring window. Defaults to 0, which is no censoring.
<code>swindow</code>	Width of the secondary event censoring window. Defaults to 0, which is no censoring.
<code>relative_obs_time</code>	Observation time relative to the primary event. Defaults to <code>Inf</code> , which is no right truncation. <code>brms</code> warns about infinite values in the data when this is <code>Inf</code> . That warning is safe here, because prediction is done in R and never passes the value to Stan.

delay_min Minimum delay, the left truncation point. Defaults to 0, which is no left truncation.

Details

The defaults give the delay distribution with no censoring and no truncation, which is a continuous probability density function. For a discrete probability mass function with daily censoring set pwindow and swindow to 1. For the delay distribution as it would be seen at a given time set relative_obs_time to that time relative to the primary event.

Value

A `tibble::tibble()` of newdata ready to predict from.

See Also

Other marginal_model: `as_epidist_marginal_model()`, `as_epidist_marginal_model.epidist_aggregate_data()`, `as_epidist_marginal_model.epidist_linelist_data()`, `epidist_family_model.epidist_marginal_model()`, `epidist_formula_model.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`, `new_epidist_marginal_model()`

Other newdata: `epidist_newdata()`, `epidist_newdata.default()`, `epidist_newdata.epidist_latent_model()`, `epidist_newdata.epidist_meta_model()`, `epidist_newdata.epidist_naive_model()`

Examples

```
prep_obs <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_marginal_model()

# A row for each sex, with no censoring and no truncation
epidist_newdata(prepare_obs, sex)

# The same, with daily censoring
epidist_newdata(prepare_obs, sex, pwindow = 1, swindow = 1)
```

epidist_newdata.epidist_meta_model

Build newdata for the meta model

Description

The meta model holds individual level rows and summary rows in one data frame, telling them apart by the obs_type slot. Prediction is for the delay distribution itself, so this method builds an individual level row. It takes the same arguments as `epidist_newdata.epidist_marginal_model()`, and fills the slots a summary row would use with the values an individual row carries. The result is what `delay_parameter_draws()` and `add_summaries()` need, so there is no need to copy a summary row out of the model data and overwrite its covariates.

Usage

```
## S3 method for class 'epidist_meta_model'
epidist_newdata(
  data,
  ...,
  pwindow = 0,
  swindow = 0,
  relative_obs_time = Inf,
  delay_min = 0
)
```

Arguments

data	An epidist data object, such as one returned by as_epidist_marginal_model() , as_epidist_latent_model() , as_epidist_naive_model() or as_epidist_meta_model() .
...	Variables to expand into a grid, passed to tidyr::expand() . Supply the variables used in the model formula, such as sex. Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.
pwindow	Width of the primary event censoring window. Defaults to 0, which is no censoring.
swindow	Width of the secondary event censoring window. Defaults to 0, which is no censoring.
relative_obs_time	Observation time relative to the primary event. Defaults to Inf, which is no right truncation. brms warns about infinite values in the data when this is Inf. That warning is safe here, because prediction is done in R and never passes the value to Stan.
delay_min	Minimum delay, the left truncation point. Defaults to 0, which is no left truncation.

Details

A model with a study level term, such as $\mu \sim 1 + (1 \mid \text{study})$, needs either a study column in newdata or `re_formula = NA` when predicting. Expand study here to predict for each study, or leave it out and pass `re_formula = NA` for the population level delay. The primary event distribution of the individual level rows is a parameter of the model rather than a column, so it needs nothing here.

Value

A [tibble::tibble\(\)](#) of newdata ready to predict from.

See Also

Other meta_model: [as_epidist_meta_model\(\)](#), [as_epidist_meta_model.NULL\(\)](#), [as_epidist_meta_model.epidist_](#), [as_epidist_meta_model.epidist_estimates_data\(\)](#), [as_epidist_meta_model.epidist_linelist_data\(\)](#),

```
assert_epidist.epidist_meta_model(), epidist_family_model.epidist_meta_model(), epidist_formula_model.epidist_meta_leave_one_out(), epidist_model_prior.epidist_meta_model(), epidist_transform_data_model.is_epidist_meta_model(), new_epidist_meta_model())
```

Other newdata: `epidist_newdata()`, `epidist_newdata.default()`, `epidist_newdata.epidist_latent_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_newdata.epidist_naive_model()`

Examples

```
estimates <- as_epidist_estimates_data(
  data.frame(
    study = c("A", "A", "B"),
    type = c("mean", "sd", "mean"),
    value = c(7.5, 3.6, 6.4),
    n = c(120, 120, 80),
    relative_obs_time = c(20, 20, Inf),
    trunc_adjusted = c(FALSE, FALSE, TRUE),
    cens_adjusted = c(0, 0, 1)
  )
)
meta <- as_epidist_meta_model(estimates = estimates)

# The population level delay, with no censoring and no truncation
epidist_newdata(meta)

# A row for each study, with daily censoring
epidist_newdata(meta, study, pwindow = 1, swindow = 1)
```

```
epidist_newdata.epidist_naive_model
```

Build newdata for the naive model

Description

The naive model accounts for neither censoring nor truncation, so the only variable it uses beyond those in the model formula is the response. This method adds it, set to NA because it is the quantity being predicted.

Usage

```
## S3 method for class 'epidist_naive_model'
epidist_newdata(data, ...)
```

Arguments

data An epidist data object, such as one returned by `as_epidist_marginal_model()`, `as_epidist_latent_model()`, `as_epidist_naive_model()` or `as_epidist_meta_model()`.

... Variables to expand into a grid, passed to `tidyr::expand()`. Supply the variables used in the model formula, such as `sex`. Each combination of their unique values becomes a row. Supply no variables to get a single row, which is what an intercept only model needs. A variable expanded here keeps its expanded values, so naming it as an argument of the method as well is an error.

Value

A `tibble::tibble()` of newdata ready to predict from.

See Also

Other naive_model: `as_epidist_naive_model()`, `as_epidist_naive_model.epidist_aggregate_data()`, `as_epidist_naive_model.epidist_linelist_data()`, `epidist_formula_model.epidist_naive_model()`, `epidist_stancode.epidist_naive_model()`, `epidist_transform_data_model.epidist_naive_model()`, `is_epidist_naive_model()`, `new_epidist_naive_model()`

Other newdata: `epidist_newdata()`, `epidist_newdata.default()`, `epidist_newdata.epidist_latent_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_newdata.epidist_meta_model()`

Examples

```
prep_obs <- sierra_leone Ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_naive_model()

# A row for each sex
epidist_newdata(prepare_obs, sex)
```

epidist_prior

Define custom prior distributions for epidist models

Description

This function combines model specific prior distributions from `epidist_model_prior()`, family specific prior distributions from `epidist_family_prior()`, and user provided prior distributions into a single set of custom priors. Each element overwrites previous elements, such that user provided prior distributions have the highest priority. If a user prior distribution is provided which is not a parameter of the model, a warning will be shown.

Usage

```
epidist_prior(
  data,
  family,
  formula,
```

```

    prior,
    merge = TRUE,
    enforce_presence = FALSE
  )

```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>family</code>	A description of the response distribution and link function to be used in the model created using <code>epidist_family()</code> .
<code>formula</code>	A symbolic description of the model to be fitted created using <code>epidist_formula()</code> .
<code>prior</code>	One or more <code>brmsprior</code> objects created by <code>brms::set_prior()</code> or related functions. These priors are passed to <code>epidist_prior()</code> in the <code>prior</code> argument. Some models have default priors that are automatically added (see <code>epidist_model_prior()</code>). These can be merged with user-provided priors using the <code>merge_priors</code> argument.
<code>merge</code>	If <code>TRUE</code> then merge new priors with existing ones, if <code>FALSE</code> only use new priors. Defaults to <code>TRUE</code> . This may be useful if the built in approaches for merging priors are not flexible enough for a particular use case.
<code>enforce_presence</code>	If <code>TRUE</code> then only allow user priors that match existing default priors. If <code>FALSE</code> then allow user priors that are not present in the default set. Defaults to <code>FALSE</code> .

Details

Note that the matching of priors is imperfect as it does not use `brms`' internal prior matching functionality. For example, it cannot distinguish between a prior for all coefficients (`class = "b"`) and a prior for a specific coefficient (`class = "b"` and `coef` specified).

Some models add parameters which `brms` does not know about, such as the event windows of the latent model. Priors for these are written using the `parameter ~ distribution` syntax of `brms::set_prior()` and are passed to Stan unchanged. A prior written this way replaces any existing prior for the same parameter and is not checked against the parameters of the model. Note that the latent model requires a `uniform(0, 1)` prior on its event windows.

Value

A `brmsprior` object containing the combined custom prior distributions.

See Also

Other prior: `epidist_family_prior()`, `epidist_family_prior.default()`, `epidist_family_prior.gengamma()`, `epidist_family_prior.lognormal()`, `epidist_model_prior()`, `epidist_model_prior.default()`, `epidist_model_prior.epidist_meta_model()`

Examples

```
data <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model()
family <- epidist_family(data, family = lognormal())
formula <- epidist_formula(data, family = family, formula = mu ~ 1)
epidist_prior(data, family = family, formula = formula, prior = NULL)
```

epidist_stancode	<i>Define model specific Stan code</i>
------------------	--

Description

This function is used within [epidist\(\)](#) to create any custom Stan code which is injected into brms via the `stanvars` argument. It is unlikely that as a user you will need this function, but we export it nonetheless to be transparent about what exactly is happening inside of a call to [epidist\(\)](#).

Usage

```
epidist_stancode(data, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>...</code>	Additional arguments passed to <code>fn</code> method.

Value

A list of `stanvars` objects, or `NULL` when none are needed.

See Also

Other stan: [epidist_stancode.default\(\)](#)

Examples

```
data <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model()
family <- epidist_family(data, family = lognormal())
formula <- epidist_formula(data, family = family, formula = mu ~ 1)
epidist_stancode(data, family = family, formula = formula)
```

```
epidist_stancode.default
```

Default method for defining model specific Stan code

Description

Default method for defining model specific Stan code

Usage

```
## Default S3 method:
epidist_stancode(data, ...)
```

Arguments

<code>data</code>	An object with class corresponding to an implemented model.
<code>...</code>	Additional arguments passed to fn method.

Value

A list of stanvars objects, or NULL when none are needed.

See Also

Other stan: [epidist_stancode\(\)](#)

```
epidist_stancode.epidist_naive_model
```

Define the Stan code of the naive model

Description

The naive model uses the brms family as it is, so it only needs Stan code for a family brms does not have, such as [gengamma\(\)](#), whose density brms calls by name.

Usage

```
## S3 method for class 'epidist_naive_model'
epidist_stancode(
  data,
  family = epidist_family(data),
  formula = epidist_formula(data),
  ...
)
```

Arguments

data	An object with class corresponding to an implemented model.
family	The epidist family object specifying the distribution
formula	The model formula
...	Additional arguments passed to fn method.

Value

A list of stanvars objects, or NULL when none are needed.

See Also

Other naive_model: [as_epidist_naive_model\(\)](#), [as_epidist_naive_model.epidist_aggregate_data\(\)](#), [as_epidist_naive_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_naive_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#), [epidist_transform_data_model.epidist_naive_model\(\)](#), [is_epidist_naive_model\(\)](#), [new_epidist_naive_model\(\)](#)

epidist_strata	<i>Unique combinations of the predictors in a model</i>
----------------	---

Description

Returns one row of the model data for each unique combination of the variables used to predict the delay distribution parameters. Passing this to [delay_parameter_draws\(\)](#) gives one set of draws per combination rather than one per observation, which is the same result with far fewer draws.

Usage

```
epidist_strata(object, vars = NULL)
```

Arguments

object	A model fit with epidist() .
vars	A character vector of variables to take unique combinations of. If NULL, the default, the variables in the distributional parameter formulas are used.

Details

The variables are taken from the right hand side of each distributional parameter formula. The remaining columns are kept from the first row of the model data in which each combination occurs. This keeps the model variables that brms requires in newdata, such as the relative observation time and the censoring windows for the latent and marginal models. Those variables do not enter the distributional parameters, so the values kept do not change the draws.

A model with only an intercept has no predictors and so returns a single row. A continuous predictor has as many combinations as it has distinct values, so consider passing vars and a grid of your own instead.

Value

A tibble with one row per unique combination of vars, with the combination columns first.

See Also

Other postprocess: [add_summaries\(\)](#), [as_dist_spec.epidist_fit\(\)](#), [delay_parameter_draws\(\)](#), [delay_summary_draws\(\)](#), [epidist_delay_draws](#)

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

epidist_strata(fit)
```

epidist_transform_data_model

The model-specific parts of an epidist_transform_data() call

Description

The model-specific parts of an epidist_transform_data() call

Usage

```
epidist_transform_data_model(data, family, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
family	A description of the response distribution and link function to be used in the model created using epidist_family() .
formula	A formula object created using epidist_formula() .
...	Additional arguments passed to fn method.

Value

The data transformed ready for fitting.

See Also

Other transform_data: [epidist_transform_data\(\)](#), [epidist_transform_data_model.default\(\)](#)

epidist_transform_data_model.default

Default method for transforming data for a model

Description

Default method for transforming data for a model

Usage

```
## Default S3 method:
epidist_transform_data_model(data, family, formula, ...)
```

Arguments

data	An object with class corresponding to an implemented model.
family	A description of the response distribution and link function to be used in the model created using epidist_family() .
formula	A formula object created using epidist_formula() .
...	Additional arguments passed to fn method.

Value

The data transformed ready for fitting.

See Also

Other transform_data: [epidist_transform_data\(\)](#), [epidist_transform_data_model\(\)](#)

epidist_transform_data_model.epidist_marginal_model

Transform data for the marginal model

Description

This method transforms data into the format required by the marginal model by:

1. Identifying required columns for the marginal model
2. Summarising the data by counting unique combinations of these columns and any variables in the model formula using [.summarise_n_by_formula\(\)](#)
3. Converting the summarised data to a marginal model object using [new_epidist_marginal_model\(\)](#) and checking it with [assert_epidist\(\)](#)
4. Informing the user about any data aggregation that occurred using [.inform_data_summarised\(\)](#)

Usage

```
## S3 method for class 'epidist_marginal_model'
epidist_transform_data_model(data, family, formula, ...)
```

Arguments

data	The data to transform
family	The epidist family object specifying the distribution
formula	The model formula
...	Additional arguments passed to methods

Value

The data transformed ready for fitting.

See Also

Other marginal_model: [as_epidist_marginal_model\(\)](#), [as_epidist_marginal_model.epidist_aggregate_data\(\)](#), [as_epidist_marginal_model.epidist_linelist_data\(\)](#), [epidist_family_model.epidist_marginal_model\(\)](#), [epidist_formula_model.epidist_marginal_model\(\)](#), [epidist_newdata.epidist_marginal_model\(\)](#), [is_epidist_marginal_model\(\)](#), [new_epidist_marginal_model\(\)](#)

```
epidist_transform_data_model.epidist_meta_model
  Transform data for the meta model
```

Description

Individual level rows are summarised by counting unique combinations of the meta model columns and any variables in the model formula, exactly as [epidist_transform_data_model.epidist_marginal_model\(\)](#) does. Summary rows are passed through unchanged because each one is a distinct reported quantity.

Usage

```
## S3 method for class 'epidist_meta_model'
epidist_transform_data_model(data, family, formula, ...)
```

Arguments

data	The data to transform
family	The epidist family object specifying the distribution
formula	The model formula
...	Additional arguments passed to methods

Value

An `epidist_meta_model` object with the individual level rows summarised and the summary rows unchanged.

See Also

Other `meta_model`: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`, `as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`, `assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model.epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_`, `is_epidist_meta_model()`, `new_epidist_meta_model()`

`epidist_transform_data_model.epidist_naive_model`
Transform data for the naive model

Description

This method transforms data into the format required by the naive model by:

1. Identifying required columns for the naive model
2. Summarising the data by counting unique combinations of these columns and any variables in the model formula using `.summarise_n_by_formula()`
3. Converting the summarised data to a naive model object using `new_epidist_naive_model()` and checking it with `assert_epidist()`
4. Informing the user about any data aggregation that occurred using `.inform_data_summarised()`

Usage

```
## S3 method for class 'epidist_naive_model'
epidist_transform_data_model(data, family, formula, ...)
```

Arguments

<code>data</code>	The data to transform
<code>family</code>	The epidist family object specifying the distribution
<code>formula</code>	The model formula
<code>...</code>	Additional arguments passed to methods

Value

The data transformed ready for fitting.

See Also

Other naive_model: `as_epidist_naive_model()`, `as_epidist_naive_model.epidist_aggregate_data()`, `as_epidist_naive_model.epidist_linelist_data()`, `epidist_formula_model.epidist_naive_model()`, `epidist_newdata.epidist_naive_model()`, `epidist_stancode.epidist_naive_model()`, `is_epidist_naive_model`, `new_epidist_naive_model()`

gengamma

Generalised gamma delay distribution family

Description

A brms custom family for the generalised gamma distribution in the Prentice (1974) parameterisation of `flexsurv::dgengamma()`. `mu` and `sigma` are the location and scale of the log delay and `Q` is a positive shape. The Weibull ($Q = 1$) and gamma ($Q = \text{sigma}$) families are special cases and the lognormal is the limit as Q goes to zero. It needs the `flexsurv` package and works with every `epidist` model.

Usage

```
gengamma(link = "identity", link_sigma = "log", link_Q = "log")
```

Arguments

`link`, `link_sigma`, `link_Q`

The link functions of `mu`, `sigma` and `Q`.

Value

A brms custom family object.

References

Prentice, R. L. (1974). A log gamma model and its maximum likelihood estimation. *Biometrika*, 61(3), 539-544. doi:[10.1093/biomet/61.3.539](https://doi.org/10.1093/biomet/61.3.539)

See Also

Other family: `epidist_family()`, `epidist_family_model()`, `epidist_family_model.default()`, `epidist_family_param()`, `epidist_family_param.default()`, `epidist_family_param.gengamma()`

Examples

```
gengamma()
```

`is_epidist_aggregate_data`*Check if data has the epidist_aggregate_data class*

Description

Check if data has the epidist_aggregate_data class

Usage

```
is_epidist_aggregate_data(data)
```

Arguments

data	The data to convert
------	---------------------

Value

A logical, TRUE if data inherits from epidist_aggregate_data and FALSE otherwise.

See Also

Other aggregate_data: [as_epidist_aggregate_data\(\)](#), [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.default\(\)](#), [as_epidist_aggregate_data.epidist_linelist_data\(\)](#), [assert_epidist.epidist_aggregate_data\(\)](#), [new_epidist_aggregate_data\(\)](#)

`is_epidist_data`*Check if data has the epidist_data class*

Description

All epidist data objects carry this class in addition to their specific class. See [epidist_data](#) for the methods it provides.

Usage

```
is_epidist_data(data)
```

Arguments

data	An object.
------	------------

Value

A logical, TRUE if data inherits from epidist_data and FALSE otherwise.

See Also

Other epidist_data: [epidist_data](#)

Examples

```
sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  is_epidist_data()
```

```
is_epidist_estimates_data
```

Check if data has the epidist_estimates_data class

Description

Check if data has the epidist_estimates_data class

Usage

```
is_epidist_estimates_data(data)
```

Arguments

data The data to convert

Value

A logical, TRUE if data inherits from epidist_estimates_data and FALSE otherwise.

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [as_epidist_estimates_data.list\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [new_epidist_estimates_data\(\)](#)

Examples

```
is_epidist_estimates_data(data.frame())
is_epidist_estimates_data(new_epidist_estimates_data(data.frame()))
```

`is_epidist_latent_model`*Check if data has the epidist_latent_model class*

Description

Check if data has the epidist_latent_model class

Usage

```
is_epidist_latent_model(data)
```

Arguments

data	An object
------	-----------

Value

A logical, TRUE if data inherits from epidist_latent_model and FALSE otherwise.

See Also

Other latent_model: [as_epidist_latent_model\(\)](#), [as_epidist_latent_model.epidist_aggregate_data\(\)](#), [as_epidist_latent_model.epidist_linelist_data\(\)](#), [epidist_family_model.epidist_latent_model\(\)](#), [epidist_formula_model.epidist_latent_model\(\)](#), [epidist_model_prior.epidist_latent_model\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [new_epidist_latent_model\(\)](#)

`is_epidist_linelist_data`*Check if data has the epidist_linelist_data class*

Description

Check if data has the epidist_linelist_data class

Usage

```
is_epidist_linelist_data(data)
```

Arguments

data	The data to convert
------	---------------------

Value

A logical, TRUE if data inherits from epidist_linelist_data and FALSE otherwise.

Value

A logical, TRUE if data inherits from epidist_meta_model and FALSE otherwise.

See Also

Other meta_model: `as_epidist_meta_model()`, `as_epidist_meta_model.NULL()`, `as_epidist_meta_model.epidist_`
`as_epidist_meta_model.epidist_estimates_data()`, `as_epidist_meta_model.epidist_linelist_data()`,
`assert_epidist.epidist_meta_model()`, `epidist_family_model.epidist_meta_model()`, `epidist_formula_model`
`epidist_meta_leave_one_out()`, `epidist_model_prior.epidist_meta_model()`, `epidist_newdata.epidist_meta_`
`epidist_transform_data_model.epidist_meta_model()`, `new_epidist_meta_model()`

Examples

```
is_epidist_meta_model(data.frame())
is_epidist_meta_model(new_epidist_meta_model(data.frame()))
```

```
is_epidist_multivariate
```

Check if an object has the epidist_multivariate class

Description

Check if an object has the epidist_multivariate class

Usage

```
is_epidist_multivariate(data)
```

Arguments

data An object.

Value

A logical.

See Also

Other multivariate: `as_epidist_multivariate()`, `as_epidist_multivariate.data.frame()`,
`as_epidist_multivariate.matrix()`, `assert_epidist.epidist_multivariate()`, `new_epidist_multivariate()`,
`print.epidist_multivariate()`, `vcov.epidist_multivariate()`

Examples

```
is_epidist_multivariate(1)
```

```
is_epidist_naive_model
```

Check if data has the epidist_naive_model class

Description

Check if data has the epidist_naive_model class

Usage

```
is_epidist_naive_model(data)
```

Arguments

data An object.

Value

A logical, TRUE if data inherits from epidist_naive_model and FALSE otherwise.

See Also

Other naive_model: [as_epidist_naive_model\(\)](#), [as_epidist_naive_model.epidist_aggregate_data\(\)](#), [as_epidist_naive_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_naive_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#), [epidist_stancode.epidist_naive_model\(\)](#), [epidist_transform_data.new_epidist_naive_model\(\)](#)

```
new_epidist_aggregate_data
```

Class constructor for epidist_aggregate_data objects

Description

Class constructor for epidist_aggregate_data objects

Usage

```
new_epidist_aggregate_data(data)
```

Arguments

data A data.frame to convert

Value

An object of class epidist_aggregate_data

See Also

Other aggregate_data: [as_epidist_aggregate_data\(\)](#), [as_epidist_aggregate_data.data.frame\(\)](#), [as_epidist_aggregate_data.default\(\)](#), [as_epidist_aggregate_data.epidist_linelist_data\(\)](#), [assert_epidist.epidist_aggregate_data\(\)](#), [is_epidist_aggregate_data\(\)](#)

Examples

```
df <- new_epidist_aggregate_data(data.frame())
class(df)
```

new_epidist_estimates_data

Class constructor for epidist_estimates_data objects

Description

Class constructor for epidist_estimates_data objects

Usage

```
new_epidist_estimates_data(data)
```

Arguments

data A data.frame to convert

Value

An object of class epidist_estimates_data

See Also

Other estimates_data: [as_epidist_estimates_data\(\)](#), [as_epidist_estimates_data.data.frame\(\)](#), [as_epidist_estimates_data.epidist_estimates_data\(\)](#), [as_epidist_estimates_data.epidist_multivariate\(\)](#), [as_epidist_estimates_data.list\(\)](#), [assert_epidist.epidist_estimates_data\(\)](#), [epidist_estimates_epireview\(\)](#), [epidist_estimates_parameters\(\)](#), [epidist_estimates_summaries\(\)](#), [is_epidist_estimates_data\(\)](#)

Examples

```
df <- new_epidist_estimates_data(data.frame())
class(df)
```

```
new_epidist_latent_model
```

Class constructor for epidist_latent_model objects

Description

Class constructor for epidist_latent_model objects

Usage

```
new_epidist_latent_model(data, primary = .primary_choices(), ...)
```

Arguments

data	An object to be set with the class epidist_latent_model
primary	The distribution of the primary event within its censoring window. "uniform", the default, assumes it is equally likely at any point. "expgrowth" tilts it, with the growth rate estimated as the pgrowth distributional parameter.
...	Additional arguments passed to methods.

Value

An object of class epidist_latent_model

See Also

Other latent_model: [as_epidist_latent_model\(\)](#), [as_epidist_latent_model.epidist_aggregate_data\(\)](#), [as_epidist_latent_model.epidist_linelist_data\(\)](#), [epidist_family_model.epidist_latent_model\(\)](#), [epidist_formula_model.epidist_latent_model\(\)](#), [epidist_model_prior.epidist_latent_model\(\)](#), [epidist_newdata.epidist_latent_model\(\)](#), [is_epidist_latent_model\(\)](#)

```
new_epidist_linelist_data
```

Class constructor for epidist_linelist_data objects

Description

Class constructor for epidist_linelist_data objects

Usage

```
new_epidist_linelist_data(data)
```

Arguments

data	A data.frame to convert
------	-------------------------

Value

An object of class `epidist_linelist_data`

See Also

Other `linelist_data`: `as_epidist_linelist_data()`, `as_epidist_linelist_data.data.frame()`, `as_epidist_linelist_data.default()`, `as_epidist_linelist_data.epidist_aggregate_data()`, `assert_epidist.epidist_linelist_data()`, `is_epidist_linelist_data()`

`new_epidist_marginal_model`

Class constructor for `epidist_marginal_model` objects

Description

Class constructor for `epidist_marginal_model` objects

Usage

```
new_epidist_marginal_model(data, primary = .primary_choices())
```

Arguments

<code>data</code>	A <code>data.frame</code> to convert
<code>primary</code>	The primary event distribution, "uniform" or "expgrowth".

Value

An object of class `epidist_marginal_model`

See Also

Other `marginal_model`: `as_epidist_marginal_model()`, `as_epidist_marginal_model.epidist_aggregate_data()`, `as_epidist_marginal_model.epidist_linelist_data()`, `epidist_family_model.epidist_marginal_model()`, `epidist_formula_model.epidist_marginal_model()`, `epidist_newdata.epidist_marginal_model()`, `epidist_transform_data_model.epidist_marginal_model()`, `is_epidist_marginal_model()`

<code>new_epidist_meta_model</code>
<i>Class constructor for epidist_meta_model objects</i>

Description

Class constructor for epidist_meta_model objects

Usage

```
new_epidist_meta_model(data, primary = .primary_choices())
```

Arguments

<code>data</code>	A data.frame to convert
<code>primary</code>	The primary event distribution of the individual level rows, "uniform" or "expgrowth". Summary rows use their growth_rate metadata instead, or the pgrowth parameter where their growth_known slot is 0.

Value

An object of class epidist_meta_model

See Also

Other meta_model: [as_epidist_meta_model\(\)](#), [as_epidist_meta_model.NULL\(\)](#), [as_epidist_meta_model.epidist_](#)
[as_epidist_meta_model.epidist_estimates_data\(\)](#), [as_epidist_meta_model.epidist_linelist_data\(\)](#),
[assert_epidist.epidist_meta_model\(\)](#), [epidist_family_model.epidist_meta_model\(\)](#), [epidist_formula_model](#)
[epidist_meta_leave_one_out\(\)](#), [epidist_model_prior.epidist_meta_model\(\)](#), [epidist_newdata.epidist_meta_](#)
[epidist_transform_data_model.epidist_meta_model\(\)](#), [is_epidist_meta_model\(\)](#)

<code>new_epidist_multivariate</code>
<i>Class constructor for epidist_multivariate objects</i>

Description

Use this where a study published a mean vector and a covariance matrix directly, rather than draws they can be computed from. Converting the object with a family, which needs the draws themselves, is then unavailable.

Usage

```
new_epidist_multivariate(  
  value,  
  vcov,  
  params,  
  index = 1,  
  n_draws = NA_integer_,  
  draws = NULL  
)
```

Arguments

value	A named numeric vector of the mean of each element.
vcov	The covariance matrix of value.
params	A character vector of the parameter names, in order.
index	The trajectory points, in order. Defaults to a single point.
n_draws	The number of draws value and vcov were computed from, or NA where they were published directly.
draws	A matrix of draws with one row per draw and one column per element of value, or NULL.

Value

An object of class `epidist_multivariate`.

See Also

Other multivariate: `as_epidist_multivariate()`, `as_epidist_multivariate.data.frame()`, `as_epidist_multivariate.matrix()`, `assert_epidist.epidist_multivariate()`, `is_epidist_multivariate()`, `print.epidist_multivariate()`, `vcov.epidist_multivariate()`

Examples

```
new_epidist_multivariate(  
  value = c(mean = 7.5, sd = 3.6),  
  vcov = matrix(c(0.09, 0.02, 0.02, 0.04), nrow = 2),  
  params = c("mean", "sd")  
)
```

```
new_epidist_naive_model
```

Class constructor for epidist_naive_model objects

Description

Class constructor for epidist_naive_model objects

Usage

```
new_epidist_naive_model(data)
```

Arguments

data An object to be set with the class epidist_naive_model.

Value

An object of class epidist_naive_model.

See Also

Other naive_model: [as_epidist_naive_model\(\)](#), [as_epidist_naive_model.epidist_aggregate_data\(\)](#), [as_epidist_naive_model.epidist_linelist_data\(\)](#), [epidist_formula_model.epidist_naive_model\(\)](#), [epidist_newdata.epidist_naive_model\(\)](#), [epidist_stancode.epidist_naive_model\(\)](#), [epidist_transform_data](#), [is_epidist_naive_model\(\)](#)

```
plot.epidist_delay_draws
```

Plot posterior draws of the delay distribution

Description

Plots the draws returned by [delay_parameter_draws\(\)](#), [delay_summary_draws\(\)](#) and [add_summaries\(\)](#). The default, type = "parameters", draws the posterior density of each distributional parameter, and of any summary column [add_summaries\(\)](#) added, in its own panel. type = "delay" draws the delay distribution the draws imply over a grid of delays, as the posterior median density with a ribbon between two quantiles, or as one line per draw when ndraws is given.

Usage

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

## S3 method for class 'epidist_delay_draws'
autoplot(
  object,
  type = c("parameters", "delay"),
  by = NULL,
  pars = NULL,
  true_values = NULL,
  ndraws = NULL,
  probs = c(0.05, 0.95),
  max_delay = NULL,
  family = NULL,
  ...
)
```

Arguments

x, object	An epidist_delay_draws object, as returned by delay_parameter_draws() , delay_summary_draws() or add_summaries() .
...	Passed from plot() to autoplot(). Unused otherwise.
type	Either "parameters", the default, to plot the posterior density of each parameter, or "delay" to plot the delay distribution the draws imply.
by	A character vector of columns of object that define the strata to colour by. If NULL, the default, the variables recorded on object are used. See the details.
pars	A character vector of the columns to plot when type = "parameters". If NULL, the default, the distributional parameters of the family are plotted, along with the mean, sd and quantile columns present.
true_values	A named numeric vector of true parameter values to mark with dashed vertical lines when type = "parameters". The names must be among the parameters plotted.
ndraws	The number of draws per stratum to plot the delay distribution of when type = "delay", one line each, sampled at random. If NULL, the default, the posterior median density is drawn with a ribbon between the probs quantiles instead.
probs	A numeric vector of two probabilities giving the quantiles the ribbon spans when type = "delay". Defaults to c(0.05, 0.95).
max_delay	The largest delay to evaluate the delay distribution at when type = "delay". If NULL, the default, the posterior median of the 99% quantile of the delay distribution is used.
family	A model fit with epidist() , a brms family, or the name of one, giving the delay distribution. If NULL, the default, the family recorded on object is used.

Details

The strata of the draws are coloured. By default they are the unique combinations of the variables in the distributional parameter formulas, which `delay_parameter_draws()` records, and each .row when the draws have several rows of newdata but no such variables. Pass by to stratify by other columns of the draws. The columns in by are kept in the plot data, so the plot can be faceted by them.

The delay distribution is evaluated with the density of the family for the lognormal, gamma and Weibull families. For any other family, delays are simulated from each draw as in `add_summaries()` and their density is estimated with `stats::density()`. Either way the density is evaluated at every draw, so thin the draws with the `ndraws` argument of `delay_parameter_draws()`, or build newdata with `epidist_strata()`, when there are many.

`plot()` and `autoplot()` are the same function. Both need `ggplot2`.

The plot is drawn with `ggplot2::theme_minimal()` and the colour blind friendly palette the package documentation uses. Add a theme or a scale of your own to the returned plot to override either.

Value

A `ggplot` object.

See Also

`delay_parameter_draws()` and `delay_summary_draws()` for the draws, and `plot_events()` to plot the data.

Other plot: `plot_delays()`, `plot_delays.default()`, `plot_delays.epidist_fit()`, `plot_delays.epidist_linelist`, `plot_delays.list()`, `plot_events()`

Examples

```
draws <- data.frame(
  mu = rnorm(200, 1.8, 0.05),
  sigma = exp(rnorm(200, log(0.5), 0.05))
) |>
  add_summaries(family = "lognormal", probs = 0.5)
plot(draws, true_values = c(mu = 1.8, sigma = 0.5))
plot(draws, type = "delay")
plot(draws, type = "delay", ndraws = 50)
```

plot_delays

Plot the observed delay distribution

Description

Bins the delay between the lower bounds of the primary and secondary event windows of each case, the delay as it was observed, and draws the proportion of cases in each bin as a column. Pass a named list of datasets to compare their observed delays, a reference distribution to draw the delay distribution they are a sample of, or a model fitted with `epidist()` to draw the delays it predicts over the delays it was fitted to.

Usage

```
plot_delays(x, ...)
```

Arguments

<code>x</code>	An <code>epidist_linelist_data</code> or <code>epidist_aggregate_data</code> object, a named list of them to compare, or a model fitted with <code>epidist()</code> .
<code>...</code>	Passed to the method.

Details

The observed delay is `stime_lwr - ptime_lwr`, the delay between the days the events were reported on when the data is censored daily. It is the response the naive model fits, so the plot shows the data that model sees rather than the delay distribution itself. Censoring and truncation both bias it, which is what the reference distribution makes visible.

Counts are weighted by the `n` column when the data has one, so aggregate data gives the same plot as the linelist it was aggregated from.

Proportions are within each stratum, so datasets of different sizes can be compared. The density is the proportion divided by binwidth, which puts the columns on the scale of the reference distribution.

The columns named by `by` are kept in the plot data, so the plot can be faceted by them.

Value

A `ggplot` object.

See Also

`plot_events()` to plot the event windows the delays come from, and `plot.epidist_delay_draws()` to plot a fitted delay distribution.

Other plot: `plot.epidist_delay_draws()`, `plot_delays.default()`, `plot_delays.epidist_fit()`, `plot_delays.epidist_linelist_data()`, `plot_delays.list()`, `plot_events()`

Examples

```
linelist <- sierra_leone Ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  )
plot_delays(linelist)
plot_delays(linelist, reference = c(mu = 1.8, sigma = 1))
plot_delays(list(All = linelist, Early = head(linelist, 500)))
```

plot_delays.default	<i>Default method for plotting observed delays</i>
---------------------	--

Description

Default method for plotting observed delays

Usage

```
## Default S3 method:
plot_delays(x, ...)
```

Arguments

x	An epidist_linelist_data or epidist_aggregate_data object, a named list of them to compare, or a model fitted with epidist() .
...	Passed to the method.

Value

This method errors. It is called when x is neither delay data nor a fitted model.

See Also

Other plot: [plot.epidist_delay_draws\(\)](#), [plot_delays\(\)](#), [plot_delays.epidist_fit\(\)](#), [plot_delays.epidist_linelist_data\(\)](#), [plot_delays.list\(\)](#), [plot_events\(\)](#)

plot_delays.epidist_fit	<i>Plot the delays a fitted model predicts over the delays it was fitted to</i>
-------------------------	---

Description

Draws the observed delays of the data the model was fitted to as columns, as [plot_delays.epidist_linelist_data\(\)](#) does, and the delays the model predicts for the same cases over them, as the posterior median proportion in each bin with a ribbon between two quantiles.

Usage

```
## S3 method for class 'epidist_fit'
plot_delays(
  x,
  by = NULL,
  binwidth = 1,
  ndraws = 100,
  probs = c(0.05, 0.95),
  ...
)
```

Arguments

<code>x</code>	An <code>epidist_linelist_data</code> or <code>epidist_aggregate_data</code> object, a named list of them to compare, or a model fitted with <code>epidist()</code> .
<code>by</code>	A character vector of columns of the model data that define the strata to colour by. If <code>NULL</code> , the default, the variables in the distributional parameter formulas are used, as <code>epidist_strata()</code> does.
<code>binwidth</code>	The width of the delay bins, on the scale of the event times. Defaults to 1, the daily censoring the data usually has.
<code>ndraws</code>	The number of posterior draws to predict from, sampled at random. Defaults to 100, which is enough for the median and the quantiles of a binned distribution and bounds the size of the prediction. Use <code>NULL</code> to predict from every draw.
<code>probs</code>	A numeric vector of two probabilities giving the quantiles the ribbon spans. Defaults to <code>c(0.05, 0.95)</code> .
<code>...</code>	Passed to the method.

Details

The predictions come from `brms::posterior_predict()`, so they are of the delay as it was observed, under the censoring and truncation of each case. That makes the plot a posterior predictive check of the observed delays rather than a picture of the delay distribution itself, which `plot.epidist_delay_draws()` draws.

Both the columns and the predictions are proportions within each stratum, so a bin holds the share of cases with that delay. Bins where neither the data nor the predictive interval puts any mass are dropped, which trims the tail of the predictive distribution.

Value

A ggplot object.

See Also

Other plot: `plot.epidist_delay_draws()`, `plot_delays()`, `plot_delays.default()`, `plot_delays.epidist_linelist`, `plot_delays.list()`, `plot_events()`

Examples

```
fit <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  ) |>
  as_epidist_aggregate_data() |>
  as_epidist_marginal_model() |>
  epidist(chains = 2, cores = 2, refresh = ifelse(interactive(), 250, 0))

plot_delays(fit)
```

plot_delays.epidist_linelist_data

Plot the observed delays of a dataset

Description

Plot the observed delays of a dataset

Usage

```
## S3 method for class 'epidist_linelist_data'
plot_delays(
  x,
  by = NULL,
  binwidth = 1,
  delay_min = NULL,
  reference = NULL,
  family = "lognormal",
  ...
)
```

Arguments

x	An epidist_linelist_data or epidist_aggregate_data object, a named list of them to compare, or a model fitted with epidist() .
by	A string naming a column of x to stratify the delays by.
binwidth	The width of the delay bins, on the scale of the event times. Defaults to 1, the daily censoring the data usually has.
delay_min	The minimum delay to mark with a dashed vertical line, as a number on the scale of the event times. If NULL, the default, the delay_min column of x is used when it has one and no line is drawn otherwise.
reference	A named numeric vector of the distributional parameters of a delay distribution to draw over the columns, as c(mu = 1.8, sigma = 0.5) for a lognormal delay. If NULL, the default, no distribution is drawn.
family	A brms family, or the name of one, giving the delay distribution reference holds the parameters of. Defaults to "lognormal".
...	Passed to the method.

Value

A ggplot object.

See Also

Other plot: [plot.epidist_delay_draws\(\)](#), [plot_delays\(\)](#), [plot_delays.default\(\)](#), [plot_delays.epidist_fit\(\)](#), [plot_delays.list\(\)](#), [plot_events\(\)](#)

plot_delays.list	<i>Compare the observed delays of several datasets</i>
------------------	--

Description

The names of x label the strata.

Usage

```
## S3 method for class 'list'
plot_delays(
  x,
  by = NULL,
  binwidth = 1,
  delay_min = NULL,
  reference = NULL,
  family = "lognormal",
  ...
)
```

Arguments

x	An epidist_linelist_data or epidist_aggregate_data object, a named list of them to compare, or a model fitted with epidist() .
by	A string naming a column of x to stratify the delays by.
binwidth	The width of the delay bins, on the scale of the event times. Defaults to 1, the daily censoring the data usually has.
delay_min	The minimum delay to mark with a dashed vertical line, as a number on the scale of the event times. If NULL, the default, the delay_min column of x is used when it has one and no line is drawn otherwise.
reference	A named numeric vector of the distributional parameters of a delay distribution to draw over the columns, as c(mu = 1.8, sigma = 0.5) for a lognormal delay. If NULL, the default, no distribution is drawn.
family	A brms family, or the name of one, giving the delay distribution reference holds the parameters of. Defaults to "lognormal".
...	Passed to the method.

Value

A ggplot object.

See Also

Other plot: [plot.epidist_delay_draws\(\)](#), [plot_delays\(\)](#), [plot_delays.default\(\)](#), [plot_delays.epidist_fit\(\)](#), [plot_delays.epidist_linelist_data\(\)](#), [plot_events\(\)](#)

plot_events

Plot the primary and secondary event windows of each case

Description

Draws one row per case, ordered by primary event time, with the primary and secondary event windows as horizontal segments joined by a line for the delay between them. A dashed vertical line marks the observation time when one is given. This is the plot the vignettes use to show how censoring and truncation obscure the delays.

Usage

```
plot_events(data, obs_time = NULL, by = NULL, n = 200)
```

Arguments

data	An <code>epidist_linelist_data</code> object.
obs_time	The observation time to mark with a dashed vertical line, as a date or a number on the scale of the event times. If <code>NULL</code> , the default, no line is drawn.
by	A string naming a column of data to colour the cases by. If <code>NULL</code> , the default, the primary and secondary event windows are coloured differently instead.
n	The maximum number of cases to draw. Defaults to 200. Use <code>Inf</code> to draw every case.

Details

Dates are used when data has the date columns that `as_epidist_linelist_data.data.frame()` keeps, `pdate_lwr` and so on, and the numeric time columns otherwise. `obs_time` must be on the same scale.

Cases are ordered by the lower bound of their primary event window and numbered in that order, so the vertical axis shows the growth of the outbreak. When there are more than `n` cases, `n` evenly spaced cases in that order are drawn. This keeps the shape of the outbreak without over-plotting.

The plot is drawn with `ggplot2::theme_minimal()` and the colour blind friendly palette the package documentation uses. Add a theme or a scale of your own to the returned plot to override either. The column named `by` is kept in the plot data, so the plot can be faceted by it.

Value

A `ggplot` object.

See Also

Other plot: `plot.epidist_delay_draws()`, `plot_delays()`, `plot_delays.default()`, `plot_delays.epidist_fit()`, `plot_delays.epidist_linelist_data()`, `plot_delays.list()`

Examples

```
linelist <- sierra_leone_ebola_data |>
  as_epidist_linelist_data(
    pdate_lwr = "date_of_symptom_onset",
    sdate_lwr = "date_of_sample_tested"
  )
plot_events(linelist, n = 100)
plot_events(linelist, obs_time = as.Date("2014-10-01"), by = "district")
```

```
print.epidist_multivariate
```

Print an epidist_multivariate object

Description

Print an epidist_multivariate object

Usage

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

Arguments

x	An epidist_multivariate object.
...	Not used.

Value

The input, invisibly.

See Also

Other multivariate: [as_epidist_multivariate\(\)](#), [as_epidist_multivariate.data.frame\(\)](#), [as_epidist_multivariate.matrix\(\)](#), [assert_epidist.epidist_multivariate\(\)](#), [is_epidist_multivariate\(\)](#), [new_epidist_multivariate\(\)](#), [vcov.epidist_multivariate\(\)](#)

Examples

```
print(new_epidist_multivariate(
  value = c(mean = 7.5, sd = 3.6),
  vcov = matrix(c(0.09, 0.02, 0.02, 0.04), nrow = 2),
  params = c("mean", "sd")
))
```

```
sierra_leone_ebola_data
```

Ebola linelist data from Fang et al. (2016)

Description

Linelist data for the Ebola virus collected in Sierra Leone. If you use this data in your work, please cite the corresponding paper.

Usage

```
sierra_leone_ebola_data
```

Format

A tibble with 8,358 rows and 7 columns:

id Unique identification number for the case

age Age as numeric

sex Sex as character, either "Female", "Male" or NA

date_of_symptom_onset The date symptoms began

date_of_sample_tested The date the sample was tested

district The district (ADM2)

chiefdom The chiefdom (ADM3)

Source

[doi:10.1073/pnas.1518587113](https://doi.org/10.1073/pnas.1518587113)

```
simulate_dates
```

Convert simulated event times to dates

Description

Takes the continuous event times produced by `simulate_gillespie()` and `simulate_secondary()` and returns the dates an analyst would actually receive. Event times are floored to their reporting window, so each event is known only by the window it fell in, and are then offset from `outbreak_start_date`.

Usage

```
simulate_dates(
  data,
  outbreak_start_date = as.Date("2024-01-01"),
  primary_window = 1,
  secondary_window = primary_window,
  obs_time = NULL,
  keep_times = FALSE
)
```

Arguments

<code>data</code>	A <code>data.frame</code> with numeric <code>ptime</code> and <code>stime</code> columns, as returned by simulate_secondary() .
<code>outbreak_start_date</code>	The date the outbreak started, corresponding to time zero.
<code>primary_window</code>	Width of the primary event reporting window in days. Either a single value used for every observation, or one value per row of data. The default of 1 gives daily reporting. Use 7 for weekly.
<code>secondary_window</code>	Width of the secondary event reporting window in days, in the same form as <code>primary_window</code> . Defaults to <code>primary_window</code> , so the two events share a reporting interval unless you say otherwise.
<code>obs_time</code>	Optional numeric observation time, in the same units as <code>ptime</code> and <code>stime</code> . When supplied an <code>obs_date</code> column is added. When <code>NULL</code> , the default, no observation date is added and as_epidist_linelist_data() falls back to the day after the last secondary event.
<code>keep_times</code>	Whether to keep the underlying numeric times. Useful when comparing estimates against the values used to simulate.

Details

The returned columns are named to match [as_epidist_linelist_data\(\)](#), so the output can be passed straight to it.

Value

A `data.frame` with `pdate_lwr`, `pdate_upr`, `sdate_lwr` and `sdate_upr` columns, and `obs_date` when `obs_time` is supplied.

See Also

Other simulate: [simulate_exponential_cases\(\)](#), [simulate_gillespie\(\)](#), [simulate_secondary\(\)](#), [simulate_study\(\)](#), [simulate_uniform_cases\(\)](#)

Examples

```
simulate_gillespie(seed = 1) |>
  simulate_secondary(meanlog = 1.8, sdlog = 0.5) |>
  simulate_dates(outbreak_start_date = as.Date("2024-02-01")) |>
  head()
```

```
simulate_exponential_cases
```

```
Simulate exponential cases
```

Description

This function simulates cases from an exponential distribution. The user may specify the rate parameter r , the sample size, and the upper bound of the survival time. If the rate parameter is 0, then this function defaults to the uniform distribution.

Usage

```
simulate_exponential_cases(r = 0.2, sample_size = 10000, seed = NULL, t = 30)
```

Arguments

<code>r</code>	The exponential growth rate parameter. Defaults to 0.2.
<code>sample_size</code>	The number of cases to simulate. Defaults to 10000.
<code>seed</code>	The random seed to be used in the simulation process. Defaults to NULL, which leaves the random number generator as it is.
<code>t</code>	Upper bound of the survival time. Defaults to 30.

Value

A data.frame with two columns: case (case number) and ptime (primary event time).

See Also

Other simulate: [simulate_dates\(\)](#), [simulate_gillespie\(\)](#), [simulate_secondary\(\)](#), [simulate_study\(\)](#), [simulate_uniform_cases\(\)](#)

Examples

```
simulate_exponential_cases(r = 0.2, sample_size = 5, seed = 101)
```

simulate_gillespie	<i>Simulate cases from a stochastic SIR model</i>
--------------------	---

Description

This function simulates cases from an stochastic SIR model. The user may specify the initial epidemic growth rate r , the rate of recovery gamma γ , the initial number of infected cases I_0 , and the total population size N .

Usage

```
simulate_gillespie(r = 0.2, gamma = 1/7, I0 = 50, N = 10000, seed = NULL)
```

Arguments

<code>r</code>	The initial epidemic growth rate. Defaults to 0.2.
<code>gamma</code>	The rate of recovery. Defaults to 1/7.
<code>I0</code>	The initial number of infected people. Defaults to 50.
<code>N</code>	The total population size. Defaults to 10000.
<code>seed</code>	The random seed to be used in the simulation process. Defaults to NULL, which leaves the random number generator as it is.

Value

A data.frame with two columns: case (case number) and ptime (primary event time).

See Also

Other simulate: [simulate_dates\(\)](#), [simulate_exponential_cases\(\)](#), [simulate_secondary\(\)](#), [simulate_study\(\)](#), [simulate_uniform_cases\(\)](#)

Examples

```
cases <- simulate_gillespie(  
  r = 0.2, gamma = 1 / 7, I0 = 5, N = 50, seed = 101  
)  
head(cases)
```

simulate_secondary	<i>Simulate secondary events based on a delay distribution</i>
--------------------	--

Description

This function simulates secondary events based on a given delay distribution. The input dataset should have the primary event times in a column named `ptime`.

Usage

```
simulate_secondary(data, dist = rlnorm, ...)
```

Arguments

<code>data</code>	A data frame with the primary event times.
<code>dist</code>	The delay distribution to be used. Defaults to <code>rlnorm()</code> .
<code>...</code>	Arguments to be passed to the delay distribution function.

Value

A `data.frame` that augments `data` with two new columns: `delay` (secondary event latency) and `stime` (the time of the secondary event).

See Also

Other simulate: [simulate_dates\(\)](#), [simulate_exponential_cases\(\)](#), [simulate_gillespie\(\)](#), [simulate_study\(\)](#), [simulate_uniform_cases\(\)](#)

Examples

```
simulate_uniform_cases(sample_size = 5) |>
  simulate_secondary(meanlog = 1.8, sdlog = 0.5)
```

simulate_study	<i>Simulate the summaries a published study would have reported</i>
----------------	---

Description

Applies the observation and estimation procedure of one study to a simulated line list and returns the summaries that study would have published, with the metadata [as_epidist_estimates_data\(\)](#) needs to forward model them. The line list must carry the exact event times that [simulate_dates\(\)](#) keeps when `keep_times = TRUE`, because the censoring adjustments a study may have used differ in how much of each event time they saw.

Usage

```
simulate_study(
  data,
  study,
  report = c("moments", "quantiles", "multivariate", "mean_se"),
  probs = c(0.25, 0.5, 0.75),
  cens_adjusted = 0,
  trunc_adjusted = FALSE,
  trunc_design = c("cohort", "accrual"),
  relative_obs_time = Inf,
  delay_min = 0,
  growth_rate = 0,
  n = NULL,
  max_delay = NULL,
  ...
)
```

Arguments

data	An <code>epidist_linelist_data</code> object built from simulated event times, with the exact <code>ptime</code> and <code>stime</code> columns kept by <code>simulate_dates()</code> when <code>keep_times = TRUE</code> . Every case must use the same primary window and the same secondary window.
study	A string labelling the study.
report	What the study published. "moments" gives a mean and a standard deviation with the sample size, "quantiles" the quantiles at <code>probs</code> with the sample size, "multivariate" the mean and standard deviation with their bootstrap covariance, through <code>new_epidist_multivariate()</code> , and "mean_se" a mean with its standard error and no sample size.
probs	The probabilities of the quantiles a "quantiles" study reports.
cens_adjusted	The censoring adjustment code the study used, one of 0 to 4 as described above and in <code>as_epidist_estimates_data()</code> .
trunc_adjusted	Whether the study corrected for right truncation. If <code>FALSE</code> , the study's truncation is applied to the line list.
trunc_design	How the study stopped collecting, "cohort" or "accrual". Only used when <code>trunc_adjusted</code> is <code>FALSE</code> .
relative_obs_time	The study's observation time. For a "cohort" design this bounds each delay, and for an "accrual" design it is the length of the collection window from the start of the line list. Defaults to <code>Inf</code> , which is only allowed for a study that adjusted for right truncation.
delay_min	The smallest measured delay the study counted. Cases below it are dropped.
growth_rate	The growth rate the study is described by. Passed through as metadata and not used to select cases, so it should be the rate the line list was simulated with, or <code>NA</code> for a study that estimates it, see <code>as_epidist_estimates_data()</code> . A <code>growth_rate_sd</code> passes through ...

<code>n</code>	The number of cases the study sampled from those it could have seen. Defaults to NULL, meaning all of them.
<code>max_delay</code>	The grid cutoff passed through to <code>as_epidist_estimates_data()</code> , or NULL for its default.
<code>...</code>	Further columns, such as covariates, added to every row of the result.

Details

The study measures each delay as its censoring adjustment code says, drops the cases its truncation would have hidden from it, drops delays below `delay_min`, takes a sample of `n` cases and summarises them. Each step follows the estimand the meta model uses for that code, see `vignette("model")`.

The measured delay is, by `cens_adjusted` code:

- 0, the difference between the lower edges of the secondary and primary windows, which is the integer date difference of a daily line list.
- 1, the exact delay between the two events.
- 2, the exact secondary time less the lower edge of the primary window, the uniform single interval approximation.
- 3, the date difference of code 0 plus half a secondary window, midpoint imputation.
- 4, the exact secondary time less the midpoint of the primary window.

A study that did not adjust for right truncation sees a case only if its delay had completed by its observation time. Under a "cohort" design the delay the observation time bounds is the underlying one, so a study on the discrete grid (codes 0 and 3) keeps a case only if the whole window its delay fell in is below `relative_obs_time`, and a code 4 study keeps a case if its code 2 delay is. Under an "accrual" design the study stopped at a calendar date `relative_obs_time` after the start of the line list, and keeps every case whose secondary event fell before it.

Sampling `n` cases and the bootstrap behind `report = "multivariate"` use the random number generator, so set a seed for a reproducible study.

The advisory checks of `as_epidist_estimates_data()` do not run on the result. They run once on the combined object when several studies are passed to `as_epidist_estimates_data()` in a list.

Value

An `epidist_estimates_data` object with one row per reported summary.

See Also

Other simulate: `simulate_dates()`, `simulate_exponential_cases()`, `simulate_gillespie()`, `simulate_secondary()`, `simulate_uniform_cases()`

Examples

```
linelist <- simulate_gillespie(seed = 1) |>
  simulate_secondary(meanlog = 1.8, sdlog = 0.5) |>
  simulate_dates(keep_times = TRUE) |>
  as_epidist_linelist_data()
simulate_study(
  linelist, "naive snapshot",
  cens_adjusted = 0, trunc_adjusted = FALSE, relative_obs_time = 15
)
```

`simulate_uniform_cases`*Simulate cases from a uniform distribution*

Description

This function simulates cases from a uniform distribution, where the primary event times are uniformly distributed between 0 and t .

Usage

```
simulate_uniform_cases(sample_size = 1000, t = 60)
```

Arguments

<code>sample_size</code>	The number of cases to simulate.
<code>t</code>	Upper bound of the uniform distribution to generate primary event times.

Value

A data.frame with two columns: `case` (case number) and `ptime` (primary event time).

See Also

Other simulate: [simulate_dates\(\)](#), [simulate_exponential_cases\(\)](#), [simulate_gillespie\(\)](#), [simulate_secondary\(\)](#), [simulate_study\(\)](#)

Examples

```
simulate_uniform_cases(sample_size = 5, t = 10)
```

`vcov.epidist_multivariate`*The covariance matrix of an epidist_multivariate object*

Description

The covariance matrix of an epidist_multivariate object

Usage

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

Arguments

<code>object</code>	An epidist_multivariate object.
<code>...</code>	Not used.

Value

The covariance matrix.

See Also

Other multivariate: [as_epidist_multivariate\(\)](#), [as_epidist_multivariate.data.frame\(\)](#), [as_epidist_multivariate.matrix\(\)](#), [assert_epidist.epidist_multivariate\(\)](#), [is_epidist_multivariate\(\)](#), [new_epidist_multivariate\(\)](#), [print.epidist_multivariate\(\)](#)

Examples

```
vcov(new_epidist_multivariate(  
  value = c(mean = 7.5, sd = 3.6),  
  vcov = matrix(c(0.09, 0.02, 0.02, 0.04), nrow = 2),  
  params = c("mean", "sd")  
))
```

Index

- * **aggregate_data**
 - as_epidist_aggregate_data, [12](#)
 - as_epidist_aggregate_data.data.frame, [13](#)
 - as_epidist_aggregate_data.default, [14](#)
 - as_epidist_aggregate_data.epidist_linelist_data, [15](#)
 - assert_epidist.epidist_aggregate_data, [7](#)
 - is_epidist_aggregate_data, [109](#)
 - new_epidist_aggregate_data, [114](#)
- * **assert**
 - assert_epidist, [6](#)
- * **datasets**
 - sierra_leone Ebola_data, [130](#)
- * **data**
 - sierra_leone Ebola_data, [130](#)
- * **diagnostics**
 - epidist_diagnostics, [58](#)
- * **epidist_data**
 - epidist_data, [54](#)
 - is_epidist_data, [109](#)
- * **estimates_data**
 - as_epidist_estimates_data, [16](#)
 - as_epidist_estimates_data.data.frame, [17](#)
 - as_epidist_estimates_data.epidist_estimates_data, [23](#)
 - as_epidist_estimates_data.epidist_multivariate_data, [24](#)
 - as_epidist_estimates_data.list, [26](#)
 - assert_epidist.epidist_estimates_data, [8](#)
 - epidist_estimates_epireview, [59](#)
 - epidist_estimates_parameters, [62](#)
 - epidist_estimates_summaries, [64](#)
 - is_epidist_estimates_data, [110](#)
 - new_epidist_estimates_data, [115](#)
- * **family**
 - epidist_family, [66](#)
 - epidist_family_model, [67](#)
 - epidist_family_model.default, [67](#)
 - epidist_family_param.default, [70](#)
 - epidist_family_param.gengamma, [71](#)
 - epidist_family_param.gengamma, [108](#)
- * **fit**
 - epidist, [53](#)
- * **formula**
 - epidist_formula, [76](#)
 - epidist_formula_model, [77](#)
 - epidist_formula_model.default, [77](#)
- * **gen**
 - epidist_gen_log_lik, [81](#)
 - epidist_gen_meta_log_lik, [82](#)
 - epidist_gen_meta_predict, [83](#)
 - epidist_gen_posterior_epred, [84](#)
 - epidist_gen_posterior_predict, [85](#)
- * **latent_model**
 - as_epidist_latent_model, [27](#)
 - as_epidist_latent_model.epidist_aggregate_data, [28](#)
 - as_epidist_latent_model.epidist_linelist_data, [29](#)
 - epidist_family_model.epidist_latent_model, [68](#)
 - epidist_formula_model.epidist_latent_model, [78](#)
 - epidist_model_prior.epidist_latent_model, [89](#)
 - epidist_newdata.epidist_latent_model, [93](#)
 - is_epidist_latent_model, [111](#)
 - new_epidist_latent_model, [116](#)
- * **linelist_data**
 - as_epidist_linelist_data, [30](#)
 - as_epidist_linelist_data.data.frame, [31](#)

as_epidist_linelist_data.default, 32
 as_epidist_linelist_data.epidist_aggregate_data, 33
 assert_epidist.epidist_linelist_data, 8
 is_epidist_linelist_data, 111
 new_epidist_linelist_data, 116
 * **marginal_model**
 as_epidist_marginal_model, 34
 as_epidist_marginal_model.epidist_aggregate_data, 35
 as_epidist_marginal_model.epidist_linelist_data, 36
 epidist_family_model.epidist_marginal_model, 69
 epidist_formula_model.epidist_marginal_model, 79
 epidist_newdata.epidist_marginal_model, 95
 epidist_transform_data_model.epidist_marginal_model, 105
 is_epidist_marginal_model, 112
 new_epidist_marginal_model, 117
 * **meta_model**
 as_epidist_meta_model, 38
 as_epidist_meta_model.epidist_aggregate_data, 39
 as_epidist_meta_model.epidist_estimates_data, 41
 as_epidist_meta_model.epidist_linelist_data, 42
 as_epidist_meta_model.NULL, 44
 assert_epidist.epidist_meta_model, 9
 epidist_family_model.epidist_meta_model, 69
 epidist_formula_model.epidist_meta_model, 80
 epidist_meta_leave_one_out, 86
 epidist_model_prior.epidist_meta_model, 90
 epidist_newdata.epidist_meta_model, 96
 epidist_transform_data_model.epidist_meta_model, 106
 is_epidist_meta_model, 112
 new_epidist_meta_model, 118
 * **multivariate**
 as_epidist_multivariate, 45
 as_epidist_multivariate.data.frame, 46
 as_epidist_multivariate.matrix, 47
 assert_epidist.epidist_multivariate, 10
 is_epidist_multivariate, 113
 new_epidist_multivariate, 118
 print.epidist_multivariate, 129
 data.epidist_multivariate, 138
 * **naive_model**
 as_epidist_naive_model, 47
 as_epidist_naive_model.epidist_aggregate_data, 48
 as_epidist_naive_model.epidist_linelist_data, 49
 epidist_formula_model.epidist_naive_model, 81
 epidist_newdata.epidist_naive_model, 88
 epidist_stancode.epidist_naive_model, 102
 epidist_transform_data_model.epidist_naive_model, 107
 is_epidist_naive_model, 114
 new_epidist_naive_model, 120
 * **newdata**
 epidist_newdata, 91
 epidist_newdata.default, 93
 epidist_newdata.epidist_latent_model, 93
 epidist_newdata.epidist_marginal_model, 95
 epidist_newdata.epidist_meta_model, 96
 epidist_newdata.epidist_naive_model, 98
 * **plot**
 plot.epidist_delay_draws, 120
 plot_delays, 122
 plot_delays.default, 124
 plot_delays.epidist_fit, 124
 plot_delays.epidist_linelist_data, 126
 plot_delays.list, 127
 plot_events, 128
 * **postprocess**

- add_summaries, 5
- as_dist_spec.epidist_fit, 10
- delay_parameter_draws, 50
- delay_summary_draws, 51
- epidist_delay_draws, 56
- epidist_strata, 103
- * prior**
 - epidist_family_prior, 72
 - epidist_family_prior.default, 73
 - epidist_family_prior.gengamma, 74
 - epidist_family_prior.lognormal, 75
 - epidist_model_prior, 88
 - epidist_model_prior.default, 89
 - epidist_model_prior.epidist_meta_model, 90
 - epidist_prior, 99
- * simulate**
 - simulate_dates, 130
 - simulate_exponential_cases, 132
 - simulate_gillespie, 133
 - simulate_secondary, 134
 - simulate_study, 134
 - simulate_uniform_cases, 137
- * stan**
 - epidist_stancode, 101
 - epidist_stancode.default, 102
- * transform_data**
 - epidist_transform_data_model, 104
 - epidist_transform_data_model.default, 105
- .inform_data_summarised(), 105, 107
- .summarise_n_by_formula(), 105, 107
- [.epidist_data (epidist_data), 54
- [.epidist_delay_draws
 - (epidist_delay_draws), 56
- [<-.epidist_data (epidist_data), 54
- [[<-.epidist_data (epidist_data), 54
- \$<-.epidist_data (epidist_data), 54
- add_delay_parameter_draws
 - (delay_parameter_draws), 50
- add_delay_parameter_draws(), 92
- add_summaries, 5
- add_summaries(), 11, 46, 50–52, 56–58, 96, 104, 120–122
- as_dist_spec.epidist_fit, 10
- as_dist_spec.epidist_fit(), 6, 51, 52, 58, 104
- as_epidist_aggregate_data, 12
- as_epidist_aggregate_data(), 7, 14–16, 109, 115
- as_epidist_aggregate_data.data.frame, 13
- as_epidist_aggregate_data.data.frame(), 7, 13, 15, 16, 109, 115
- as_epidist_aggregate_data.default, 14
- as_epidist_aggregate_data.default(), 7, 13, 14, 16, 109, 115
- as_epidist_aggregate_data.epidist_linelist_data, 15
- as_epidist_aggregate_data.epidist_linelist_data(), 7, 13–15, 109, 115
- as_epidist_estimates_data, 16
- as_epidist_estimates_data(), 8, 22–26, 39, 45, 59, 61, 64, 65, 110, 115, 134–136
- as_epidist_estimates_data.data.frame, 17
- as_epidist_estimates_data.data.frame(), 8, 17, 24–26, 60–65, 110, 115
- as_epidist_estimates_data.epidist_estimates_data, 23
- as_epidist_estimates_data.epidist_estimates_data(), 8, 17, 23, 25, 26, 61, 64, 65, 110, 115
- as_epidist_estimates_data.epidist_multivariate, 24
- as_epidist_estimates_data.epidist_multivariate(), 8, 17, 23, 24, 26, 61, 64, 65, 110, 115
- as_epidist_estimates_data.list, 26
- as_epidist_estimates_data.list(), 8, 17, 23–25, 61, 64, 65, 110, 115
- as_epidist_latent_model, 27
- as_epidist_latent_model(), 28, 29, 36, 49, 68, 79, 90, 92–95, 97, 98, 111, 116
- as_epidist_latent_model.epidist_aggregate_data, 28
- as_epidist_latent_model.epidist_aggregate_data(), 27, 29, 68, 79, 90, 94, 111, 116
- as_epidist_latent_model.epidist_linelist_data, 29
- as_epidist_latent_model.epidist_linelist_data(), 27, 28, 68, 79, 90, 94, 111, 116
- as_epidist_linelist_data, 30
- as_epidist_linelist_data(), 9, 15, 29, 32, 33, 112, 117, 131
- as_epidist_linelist_data.data.frame, 31

- `as_epidist_linelist_data.data.frame()`,
9, 13, 30, 33, 112, 117, 128
- `as_epidist_linelist_data.default`, 32
- `as_epidist_linelist_data.default()`, 9,
14, 30–33, 112, 117
- `as_epidist_linelist_data.epidist_aggregate_data`,
33
- `as_epidist_linelist_data.epidist_aggregate_data()`,
9, 28, 30, 32, 33, 112, 117
- `as_epidist_marginal_model`, 34
- `as_epidist_marginal_model()`, 19, 36–38,
49, 69, 79, 92–98, 106, 112, 117
- `as_epidist_marginal_model.epidist_aggregate_data`,
35
- `as_epidist_marginal_model.epidist_aggregate_data()`,
34, 37, 69, 79, 96, 106, 112, 117
- `as_epidist_marginal_model.epidist_linelist_data`,
36
- `as_epidist_marginal_model.epidist_linelist_data()`,
34–36, 40, 42, 43, 69, 79, 96, 106,
112, 117
- `as_epidist_meta_model`, 38
- `as_epidist_meta_model()`, 9, 17–20, 22, 24,
40–45, 62, 64, 70, 80, 82, 83, 87,
91–95, 97, 98, 107, 113, 118
- `as_epidist_meta_model.epidist_aggregate_data`,
39
- `as_epidist_meta_model.epidist_aggregate_data()`,
9, 39, 42–44, 70, 80, 87, 91, 97, 107,
113, 118
- `as_epidist_meta_model.epidist_estimates_data`,
41
- `as_epidist_meta_model.epidist_estimates_data()`,
9, 39, 41, 43, 44, 70, 80, 87, 91, 97,
107, 113, 118
- `as_epidist_meta_model.epidist_linelist_data`,
42
- `as_epidist_meta_model.epidist_linelist_data()`,
9, 39–42, 44, 70, 80, 87, 91, 97, 107,
113, 118
- `as_epidist_meta_model.NULL`, 44
- `as_epidist_meta_model.NULL()`, 9, 39,
41–43, 70, 80, 87, 91, 97, 107, 113,
118
- `as_epidist_multivariate`, 45
- `as_epidist_multivariate()`, 10, 22, 46, 47,
63, 113, 119, 129, 138
- `as_epidist_multivariate.data.frame`, 46
- `as_epidist_multivariate.data.frame()`,
10, 45, 47, 113, 119, 129, 138
- `as_epidist_multivariate.matrix`, 47
- `as_epidist_multivariate.matrix()`, 10,
45, 46, 113, 119, 129, 138
- `as_epidist_naive_model`, 47
- `as_epidist_naive_model()`, 49, 50, 81,
92–95, 97–99, 103, 108, 114, 120
- `as_epidist_naive_model.epidist_aggregate_data`,
48
- `as_epidist_naive_model.epidist_aggregate_data()`,
48, 50, 81, 99, 103, 108, 114, 120
- `as_epidist_naive_model.epidist_linelist_data`,
49
- `as_epidist_naive_model.epidist_linelist_data()`,
48, 49, 81, 99, 103, 108, 114, 120
- `assert_epidist`, 6
- `assert_epidist()`, 105, 107
- `assert_epidist.epidist_aggregate_data`,
7
- `assert_epidist.epidist_aggregate_data()`,
13–16, 109, 115
- `assert_epidist.epidist_estimates_data`,
8
- `assert_epidist.epidist_estimates_data()`,
17, 23–26, 61, 64, 65, 110, 115
- `assert_epidist.epidist_linelist_data`,
8
- `assert_epidist.epidist_linelist_data()`,
30, 32, 33, 112, 117
- `assert_epidist.epidist_meta_model`, 9
- `assert_epidist.epidist_meta_model()`,
39, 41–44, 70, 80, 87, 91, 98, 107,
113, 118
- `assert_epidist.epidist_multivariate`,
10
- `assert_epidist.epidist_multivariate()`,
45–47, 113, 119, 129, 138
- `autoplot.epidist_delay_draws`
(`plot.epidist_delay_draws`), 120
- `base::rbind()`, 55
- `brms::brm()`, 54, 59, 87
- `brms::brmsfamily()`, 53, 66, 71–75, 82–85
- `brms::brmsformula`, 53, 73–81, 89, 90
- `brms::default_prior()`, 90
- `brms::log_lik()`, 82, 83
- `brms::lognormal()`, 53, 66, 71–75, 82–85
- `brms::make_stancode()`, 54

- `brms::make_standata()`, 54
- `brms::posterior_epred()`, 84, 85, 92
- `brms::posterior_predict()`, 84–86, 92, 125
- `brms::prepare_predictions()`, 11, 50, 52, 86
- `brms::set_prior()`, 54, 100
- `brms::update.brmsfit()`, 87
- `brms::weibull()`, 66
-
- `delay_parameter_draws`, 50
- `delay_parameter_draws()`, 5, 6, 11, 46, 52, 58, 92, 96, 103, 104, 120–122
- `delay_summary_draws`, 51
- `delay_summary_draws()`, 6, 11, 51, 58, 86, 87, 104, 120–122
- `distspec::bound_dist()`, 11
- `distspec::Gamma()`, 11
- `distspec::LogNormal()`, 11
- `distspec::Normal()`, 11
- `distspec::Weibull()`, 11
- `dplyr::bind_cols()`, 56
- `dplyr::bind_rows()`, 57
- `dplyr::dplyr_col_modify()`, 55–57
- `dplyr::dplyr_reconstruct()`, 55–57
- `dplyr::dplyr_row_slice()`, 55–57
- `dplyr::filter()`, 56
- `dplyr::group_by()`, 55–57
- `dplyr::mutate()`, 56, 57, 61
- `dplyr::select()`, 56
- `dplyr::summarise()`, 56, 57
- `dplyr::ungroup()`, 56, 57
- `dplyr_col_modify.epidist_data`
(`epidist_data`), 54
- `dplyr_col_modify.epidist_delay_draws`
(`epidist_delay_draws`), 56
- `dplyr_reconstruct.epidist_data`
(`epidist_data`), 54
- `dplyr_reconstruct.epidist_delay_draws`
(`epidist_delay_draws`), 56
- `dplyr_row_slice.epidist_data`
(`epidist_data`), 54
- `dplyr_row_slice.epidist_delay_draws`
(`epidist_delay_draws`), 56
-
- `epidist`, 53
- `epidist()`, 5, 11, 27, 34, 37, 38, 47, 49, 50, 52, 66, 76, 86, 101, 103, 121–127
- `epidist_data`, 54, 61, 109, 110
-
- `epidist_delay_draws`, 6, 11, 51, 52, 56, 104
- `epidist_diagnostics`, 58
- `epidist_estimates_epireview`, 59
- `epidist_estimates_epireview()`, 8, 17, 23–26, 64, 65, 110, 115
- `epidist_estimates_parameters`, 62
- `epidist_estimates_parameters()`, 8, 17, 21, 23–26, 60, 61, 65, 110, 115
- `epidist_estimates_summaries`, 64
- `epidist_estimates_summaries()`, 8, 17, 21, 23–26, 61, 64, 110, 115
- `epidist_family`, 66
- `epidist_family()`, 67, 68, 71, 72, 76, 100, 104, 105, 108
- `epidist_family_model`, 67
- `epidist_family_model()`, 66, 68, 71, 72, 108
- `epidist_family_model.default`, 67
- `epidist_family_model.default()`, 66, 67, 71, 72, 108
- `epidist_family_model.epidist_latent_model`, 68
- `epidist_family_model.epidist_latent_model()`, 27–29, 79, 90, 94, 111, 116
- `epidist_family_model.epidist_marginal_model`, 69
- `epidist_family_model.epidist_marginal_model()`, 34, 36, 37, 79, 96, 106, 112, 117
- `epidist_family_model.epidist_meta_model`, 69
- `epidist_family_model.epidist_meta_model()`, 9, 39, 41–44, 80, 87, 91, 98, 107, 113, 118
- `epidist_family_param()`, 66–68, 71, 72, 108
- `epidist_family_param.default`, 70
- `epidist_family_param.default()`, 66–68, 72, 108
- `epidist_family_param.gengamma`, 71
- `epidist_family_param.gengamma()`, 66–68, 71, 108
- `epidist_family_prior`, 72
- `epidist_family_prior()`, 73–75, 88–91, 99, 100
- `epidist_family_prior.default`, 73
- `epidist_family_prior.default()`, 72, 74, 75, 88, 89, 91, 100
- `epidist_family_prior.gengamma`, 74

- `epidist_family_prior.gengamma()`, [72, 73, 75, 88, 89, 91, 100](#)
- `epidist_family_prior.lognormal`, [75](#)
- `epidist_family_prior.lognormal()`, [72–74, 88, 89, 91, 100](#)
- `epidist_formula`, [76](#)
- `epidist_formula()`, [77, 78, 100, 104, 105](#)
- `epidist_formula_model`, [77](#)
- `epidist_formula_model()`, [76, 78](#)
- `epidist_formula_model.default`, [77](#)
- `epidist_formula_model.default()`, [76, 77](#)
- `epidist_formula_model.epidist_latent_model`, [78](#)
- `epidist_formula_model.epidist_latent_model()`, [27–29, 68, 90, 94, 111, 116](#)
- `epidist_formula_model.epidist_marginal_model`, [79](#)
- `epidist_formula_model.epidist_marginal_model()`, [34, 36, 37, 69, 96, 106, 112, 117](#)
- `epidist_formula_model.epidist_meta_model`, [80](#)
- `epidist_formula_model.epidist_meta_model()`, [9, 39, 41–44, 70, 87, 91, 98, 107, 113, 118](#)
- `epidist_formula_model.epidist_naive_model`, [81](#)
- `epidist_formula_model.epidist_naive_model()`, [48–50, 99, 103, 108, 114, 120](#)
- `epidist_gen_log_lik`, [81](#)
- `epidist_gen_log_lik()`, [82–86](#)
- `epidist_gen_meta_log_lik`, [82](#)
- `epidist_gen_meta_log_lik()`, [39, 82, 84–86](#)
- `epidist_gen_meta_predict`, [83](#)
- `epidist_gen_meta_predict()`, [82, 83, 85, 86](#)
- `epidist_gen_posterior_epred`, [84](#)
- `epidist_gen_posterior_epred()`, [82–84, 86](#)
- `epidist_gen_posterior_predict`, [85](#)
- `epidist_gen_posterior_predict()`, [82–85](#)
- `epidist_meta_leave_one_out`, [86](#)
- `epidist_meta_leave_one_out()`, [9, 39, 41–44, 70, 80, 91, 98, 107, 113, 118](#)
- `epidist_model_prior`, [88](#)
- `epidist_model_prior()`, [39, 54, 72–75, 80, 89, 91, 99, 100](#)
- `epidist_model_prior.default`, [89](#)
- `epidist_model_prior.default()`, [72–75, 88, 91, 100](#)
- `epidist_model_prior.epidist_latent_model`, [89](#)
- `epidist_model_prior.epidist_latent_model()`, [27–29, 68, 79, 94, 111, 116](#)
- `epidist_model_prior.epidist_meta_model`, [90](#)
- `epidist_model_prior.epidist_meta_model()`, [9, 39, 41–44, 70, 72–75, 80, 87–89, 98, 100, 107, 113, 118](#)
- `epidist_newdata`, [91](#)
- `epidist_newdata()`, [11, 86, 87, 93, 94, 96, 98, 99](#)
- `epidist_newdata.default`, [93](#)
- `epidist_newdata.default()`, [92, 94, 96, 98, 99](#)
- `epidist_newdata.epidist_latent_model`, [93](#)
- `epidist_newdata.epidist_latent_model()`, [27–29, 68, 79, 90, 92, 93, 96, 98, 99, 111, 116](#)
- `epidist_newdata.epidist_marginal_model`, [95](#)
- `epidist_newdata.epidist_marginal_model()`, [34, 36, 37, 69, 79, 92–94, 96, 98, 99, 106, 112, 117](#)
- `epidist_newdata.epidist_meta_model`, [96](#)
- `epidist_newdata.epidist_meta_model()`, [9, 39, 41–44, 70, 80, 87, 91–94, 96, 99, 107, 113, 118](#)
- `epidist_newdata.epidist_naive_model`, [98](#)
- `epidist_newdata.epidist_naive_model()`, [48–50, 81, 92–94, 96, 98, 103, 108, 114, 120](#)
- `epidist_prior`, [99](#)
- `epidist_prior()`, [54, 72–75, 88–91, 100](#)
- `epidist_stancode`, [101](#)
- `epidist_stancode()`, [102](#)
- `epidist_stancode.default`, [102](#)
- `epidist_stancode.default()`, [101](#)
- `epidist_stancode.epidist_naive_model`, [102](#)
- `epidist_stancode.epidist_naive_model()`, [48–50, 81, 99, 108, 114, 120](#)
- `epidist_strata`, [103](#)
- `epidist_strata()`, [5, 6, 11, 50–52, 58, 122,](#)

- [125](#)
- `epidist_transform_data()`, [104](#), [105](#)
- `epidist_transform_data_model`, [104](#)
- `epidist_transform_data_model()`, [105](#)
- `epidist_transform_data_model.default`, [105](#)
- `epidist_transform_data_model.default()`, [104](#)
- `epidist_transform_data_model.epidist_marginal_model`, [105](#)
- `epidist_transform_data_model.epidist_marginal_model()`, [34](#), [36](#), [37](#), [69](#), [79](#), [96](#), [106](#), [112](#), [117](#)
- `epidist_transform_data_model.epidist_meta_model`, [106](#)
- `epidist_transform_data_model.epidist_meta_model()`, [9](#), [39](#), [41–44](#), [70](#), [80](#), [87](#), [91](#), [98](#), [113](#), [118](#)
- `epidist_transform_data_model.epidist_naive_model`, [107](#)
- `epidist_transform_data_model.epidist_naive_model()`, [48–50](#), [81](#), [99](#), [103](#), [114](#), [120](#)
- `gengamma`, [108](#)
- `gengamma()`, [5](#), [66–68](#), [71](#), [72](#), [102](#)
- `ggplot2::theme_minimal()`, [122](#), [128](#)
- `group_by.epidist_data(epidist_data)`, [54](#)
- `group_by.epidist_delay_draws(epidist_delay_draws)`, [56](#)
- `is_epidist_aggregate_data`, [109](#)
- `is_epidist_aggregate_data()`, [7](#), [13–16](#), [115](#)
- `is_epidist_data`, [109](#)
- `is_epidist_data()`, [56](#)
- `is_epidist_estimates_data`, [110](#)
- `is_epidist_estimates_data()`, [8](#), [17](#), [23–26](#), [61](#), [64](#), [65](#), [115](#)
- `is_epidist_latent_model`, [111](#)
- `is_epidist_latent_model()`, [27–29](#), [68](#), [79](#), [90](#), [94](#), [116](#)
- `is_epidist_linelist_data`, [111](#)
- `is_epidist_linelist_data()`, [9](#), [30](#), [32](#), [33](#), [117](#)
- `is_epidist_marginal_model`, [112](#)
- `is_epidist_marginal_model()`, [34](#), [36](#), [37](#), [69](#), [79](#), [96](#), [106](#), [117](#)
- `is_epidist_meta_model`, [112](#)
- `is_epidist_meta_model()`, [9](#), [39](#), [41–44](#), [70](#), [80](#), [87](#), [91](#), [98](#), [107](#), [118](#)
- `is_epidist_multivariate`, [113](#)
- `is_epidist_multivariate()`, [10](#), [45–47](#), [119](#), [129](#), [138](#)
- `is_epidist_naive_model`, [114](#)
- `is_epidist_naive_model()`, [48–50](#), [81](#), [99](#), [103](#), [108](#), [120](#)
- `loo::loo()`, [39](#), [83](#)
- `names<- .epidist_data(epidist_data)`, [54](#)
- `new_epidist_delay_draws(epidist_delay_draws)`, [56](#)
- `new_epidist_aggregate_data`, [114](#)
- `new_epidist_aggregate_data()`, [7](#), [13–16](#), [109](#)
- `new_epidist_estimates_data`, [115](#)
- `new_epidist_estimates_data()`, [8](#), [17](#), [23–26](#), [61](#), [64](#), [65](#), [110](#)
- `new_epidist_latent_model`, [116](#)
- `new_epidist_latent_model()`, [27–29](#), [68](#), [79](#), [90](#), [94](#), [111](#)
- `new_epidist_linelist_data`, [116](#)
- `new_epidist_linelist_data()`, [9](#), [30](#), [32](#), [33](#), [112](#)
- `new_epidist_marginal_model`, [117](#)
- `new_epidist_marginal_model()`, [34](#), [36](#), [37](#), [69](#), [79](#), [96](#), [105](#), [106](#), [112](#)
- `new_epidist_meta_model`, [118](#)
- `new_epidist_meta_model()`, [9](#), [39](#), [41–44](#), [70](#), [80](#), [87](#), [91](#), [98](#), [107](#), [113](#)
- `new_epidist_multivariate`, [118](#)
- `new_epidist_multivariate()`, [10](#), [45–47](#), [113](#), [129](#), [135](#), [138](#)
- `new_epidist_naive_model`, [120](#)
- `new_epidist_naive_model()`, [48–50](#), [81](#), [99](#), [103](#), [107](#), [108](#), [114](#)
- `plot()`, [6](#), [51](#), [57](#)
- `plot.epidist_delay_draws`, [120](#)
- `plot.epidist_delay_draws()`, [51](#), [56](#), [123–128](#)
- `plot_delays`, [122](#)
- `plot_delays()`, [122](#), [124–128](#)
- `plot_delays.default`, [124](#)
- `plot_delays.default()`, [122](#), [123](#), [125–128](#)
- `plot_delays.epidist_fit`, [124](#)
- `plot_delays.epidist_fit()`, [122–124](#), [126–128](#)
- `plot_delays.epidist_linelist_data`, [126](#)

plot_delays.epidist_linelist_data(),
 122–125, 127, 128
 plot_delays.list, 127
 plot_delays.list(), 122–126, 128
 plot_events, 128
 plot_events(), 122–127
 posterior::quantile2(), 5
 primarycensored::dpcens(), 35, 37, 40, 43,
 81
 primarycensored::rpcens(), 85, 86
 print.epidist_multivariate, 129
 print.epidist_multivariate(), 10, 45–47,
 113, 119, 138

 rbind.epidist_data(epidist_data), 54
 rlnorm(), 134

 sierra_leone_ebola_data, 130
 simulate_dates, 130
 simulate_dates(), 132–137
 simulate_exponential_cases, 132
 simulate_exponential_cases(), 131, 133,
 134, 136, 137
 simulate_gillespie, 133
 simulate_gillespie(), 130–132, 134, 136,
 137
 simulate_secondary, 134
 simulate_secondary(), 130–133, 136, 137
 simulate_study, 134
 simulate_study(), 22, 131–134, 137
 simulate_uniform_cases, 137
 simulate_uniform_cases(), 131–134, 136
 stats::density(), 122
 stats::formula, 53, 73–81, 89, 90

 tibble::tibble(), 92, 94, 96, 97, 99
 tidybayes::add_epred_draws(), 50
 tidyr::expand(), 91–95, 97, 99

 ungroup.epidist_data(epidist_data), 54
 ungroup.epidist_delay_draws
 (epidist_delay_draws), 56

 vcov.epidist_multivariate, 138
 vcov.epidist_multivariate(), 10, 45–47,
 113, 119, 129