# Package ‘epiparameter’

January 8, 2025

**Title** Classes and Helper Functions for Working with Epidemiological Parameters

**Version** 0.4.0

**Description** Classes and helper functions for loading, extracting, converting, manipulating, plotting and aggregating epidemiological parameters for infectious diseases. Epidemiological parameters extracted from the literature are loaded from the 'epiparameterDB' R package.

**License** MIT + file LICENSE

**URL** `https://github.com/epiverse-trace/epiparameter/`, `https://epiverse-trace.github.io/epiparameter/`

**BugReports** `https://github.com/epiverse-trace/epiparameter/issues`

**Depends** R (>= 4.1.0)

**Imports** cachem, checkmate, cli, distcrete, distributional, epiparameterDB, graphics, lifecycle, pillar, rlang, stats, tools, utils

**Suggests** bookdown, DT, ggplot2, knitr, RColorBrewer, rmarkdown, spelling, testthat (>= 3.0.0), vdiffr (>= 1.0.7), visNetwork

**VignetteBuilder** knitr

**Config/Needs/check** mrc-ide/epireview

**Config/Needs/website** epiverse-trace/epiversetheme, mrc-ide/epireview

**Config/potools/style** explicit

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**LazyData** true

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Joshua W. Lambert [aut, cre, cph]
(<https://orcid.org/0000-0001-5218-3046>),
Adam Kucharski [aut, cph] (<https://orcid.org/0000-0001-8814-9421>),
Carmen Tamayo [aut] (<https://orcid.org/0000-0003-4184-2864>),
Hugo Gruson [ctb, rev] (<https://orcid.org/0000-0002-4094-1476>),
Sebastian Funk [ctb] (<https://orcid.org/0000-0002-2842-3406>),
Pratik Gupte [rev] (<https://orcid.org/0000-0001-5294-7819>),
James M. Azam [rev] (<https://orcid.org/0000-0001-5782-7330>),
Chris Hartgerink [rev] (<https://orcid.org/0000-0003-1050-6809>),
Tim Taylor [rev] (<https://orcid.org/0000-0002-8587-7113>)

**Maintainer** Joshua W. Lambert <joshua.lambert@lshtm.ac.uk>

**Repository** CRAN

**Date/Publication** 2025-01-08 14:50:09 UTC

# Contents

aggregate.multi_epiparameter . . . 3
as.data.frame.epiparameter . . . 4
as.data.frame.multi_epiparameter . . . 5
as.function.epiparameter . . . 5
assert_epiparameter . . . 6
as_epiparameter . . . 7
as_epiparameter.data.frame . . . 8
c.epiparameter . . . 8
calc_disc_dist_quantile . . . 9
convert_params_to_summary_stats . . . 10
convert_summary_stats_to_params . . . 11
create_citation . . . 13
create_metadata . . . 14
create_method_assess . . . 15
create_prob_distribution . . . 17
create_region . . . 19
create_summary_stats . . . 19
create_uncertainty . . . 22
discretise . . . 23
epiparameter . . . 24
epiparameter_db . . . 27
epiparameter_distribution_functions . . . 30
epiparameter_options . . . 31
epireview_core_cols . . . 31
extract_param . . . 32
family.epiparameter . . . 34
format.epiparameter . . . 35
get_citation . . . 36
get_citation.epiparameter . . . 36
get_citation.multi_epiparameter . . . 37
get_parameters . . . 38

get_parameters.epiparameter . . . 38
is_continuous . . . 39
is_epiparameter . . . 40
is_epiparameter_params . . . 41
is_parameterised . . . 42
is_truncated . . . 43
lines.epiparameter . . . 44
mean.epiparameter . . . 45
parameter_tbl . . . 45
plot.epiparameter . . . 46
plot.multi_epiparameter . . . 47
print.epiparameter . . . 48
print.multi_epiparameter . . . 49
test_epiparameter . . . 50

**Index** **51**

---

`aggregate.multi_epiparameter` *Aggregate multiple* `<epiparameter>` *objects into a single* `<epiparameter>` *object.*

---

## Description

Combine a list of `<epiparameter>` objects into a single `<epiparameter>` with a mixture distribution [see `distributional::dist_mixture()`].

The aggregated `<epiparameter>` returned from `aggregate()` can then be used with the `density()`, `cdf()`, `quantile()` and `generate()` methods for the combined distributions.

## Usage

```
## S3 method for class 'multi_epiparameter'
aggregate(x, weighting = c("equal", "sample_size", "custom"), ..., weights)
```

## Arguments

| | |
|---|---|
| `x` | A `<multi_epiparameter>` object. |
| `weighting` | A character string with the type of weighting to use to create the mixture distribution. Options are: `"equal"` for equal weighting across distributions, `"sample_size"` for using the sample size in each `<epiparameter>` object to weight the distribution (the sample sizes are normalised), or `"custom"` allows a vector of weights to be passed to the `weights` argument for a custom weighting. |
| `...` | dots Not used, will warn if extra arguments are passed to function. |
| `weights` | A `numeric` vector of equal length the number of `<epiparameter>` objects passed to `x`. `weights` is only required if `weighting = "custom"`. |

### Details

The `aggregate()` method requires that all `<epiparameter>` objects are parameterised with `<distribution>` objects (from the **distributional** package). This means that unparameterised (see `is_parameterised()`) or discretised (see `discretise()`) distributions cannot be aggregated and the function will error.

### Value

An `<epiparameter>` object

### Examples

```
ebola_si <- epiparameter_db(epi_name = "serial interval", disease = "ebola")
aggregate(ebola_si)
```

---

## as.data.frame.epiparameter

`as.data.frame()` *method for* `<epiparameter>` *class*

---

### Description

`as.data.frame()` method for `<epiparameter>` class

### Usage

```
## S3 method for class 'epiparameter'
as.data.frame(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Details

The `<data.frame>` returned will contain some atomic columns (i.e. one object per row), and other columns that are lists (i.e. multiple objects per row). The list columns can contain lists or S3 objects (e.g. `<bibentry>` object in the `citation` column).

### Value

A `<data.frame>` with a single row.

### Examples

```
ep <- epiparameter_db(single_epiparameter = TRUE)
as.data.frame(ep)
```

## as.data.frame.multi_epiparameter

`as.data.frame()` *method for* `<multi_epiparameter>` *class*

### Description

`as.data.frame()` method for `<multi_epiparameter>` class

### Usage

```
## S3 method for class 'multi_epiparameter'
as.data.frame(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `<multi_epiparameter>` object. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Details

The `<data.frame>` returned will contain some atomic columns (i.e. one object per row), and other columns that are lists (i.e. multiple objects per row). The list columns can contain lists or S3 objects (e.g. `<bibentry>` object in the `citation` column).

### Value

A `<data.frame>` with as many rows as length of input list.

### Examples

```
db <- epiparameter_db()
as.data.frame(db)
```

## as.function.epiparameter

`as.function()` *method for* `<epiparameter>` *class*

### Description

Converts an `<epiparameter>` object to a distribution function (see epiparameter_distribution_functions), either probability density/mass function, (`density`), cumulative distribution function (`cdf`), random number generator (`generate`), or quantile (`quantile`).

### Usage

```
## S3 method for class 'epiparameter'
as.function(x, func_type = c("density", "cdf", "generate", "quantile"), ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `func_type` | A single `character` string specifying which distribution to convert `<epiparameter>` object into. Default is `"density"`. Other options are `"cdf"`, `"generate"`, or `"quantile"`. |
| `...` | dots Extra arguments to be passed to the method. |

### Details

The function returned takes a single required argument `x`.

### Value

A function object.

### Examples

```
ep <- epiparameter_db(single_epiparameter = TRUE)
# by default it will convert to a density function
f <- as.function(ep)
# use function
f(10)

f <- as.function(ep, func_type = "cdf")
f(10)
```

---

## `assert_epiparameter` *Assert an object is a valid* `<epiparameter>` *object*

---

### Description

Assert an object is a valid `<epiparameter>` object

### Usage

```
assert_epiparameter(x)
```

### Arguments

| | |
|---|---|
| `x` | An R object. |

### Value

Invisibly returns an `<epiparameter>`. Called for side-effects (errors when invalid `<epiparameter>` object is provided).

## Examples

```
ep <- epiparameter_db(single_epiparameter = TRUE)
assert_epiparameter(ep)

# example with invalid <epiparameter>
ep$disease <- NULL
try(assert_epiparameter(ep))
```

---

## as_epiparameter *Convert to an* <epiparameter> *object*

---

### Description

Convert from an R object to an <epiparameter> object. If conversion is not possible the function will error.

### Usage

```
as_epiparameter(x, ...)
```

### Arguments

| | |
|---|---|
| x | an object used to select a method. |
| ... | dots Extra arguments to be passed to the method. |

### Details

To create the full citation the information from the article table from the **epireview** package of the corresponding entry will need to be passed to function via the ... argument. The argument should be called article, as it will be matched by name by $.

To specify a probability distribution pass a character string to the function via the ... argument. The argument should be called prob_distribution. For example, to specify a gamma distribution: as_epiparameter(x, prob_distribution = "gamma").

**Warning**: distributions specified via the prob_dist argument will overwrite the probability distribution specified in the x argument. For example, if the probability distribution is given in an **epireview** entry and the prob_dist argument is specified then the function may error or return an unparameterised <epiparameter> if the parameterisation becomes incompatible.

Valid probability distributions are: "gamma", "lnorm", "weibull", "nbinom", "geom", "pois", "norm", "exp".

### Value

An <epiparameter> object or list of <epiparameter> objects.

## as_epiparameter.data.frame — *Convert* `<data.frame>` *to an* `<epiparameter>` *object*

### Description

Convert the tabular information in `<data.frame>` to an `<epiparameter>`. If the information in the `<data.frame>` cannot be converted into an `<epiparameter>` the function will error.

### Usage

```
## S3 method for class 'data.frame'
as_epiparameter(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | A `<data.frame>`. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Value

An `<epiparameter>` object or list of `<epiparameter>` objects.

### Examples

```
ep <- epiparameter_db(single_epiparameter = TRUE)
df <- as.data.frame(ep)
ep <- as_epiparameter(df)
```

## c.epiparameter — `c()` *method for* `<epiparameter>` *class*

### Description

`c()` method for `<epiparameter>` class

### Usage

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

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

### Arguments

| | |
|---|---|
| `...` | dots Objects to be concatenated. |

## Value

An <epiparameter> or list of <epiparameter> objects.

## Examples

```
db <- epiparameter_db()

# combine two <epiparameter> objects into a list
c(db[[1]], db[[2]])

# combine a list of <epiparameter> objects and a single <epiparameter> object
c(db, db[[1]])
```

---

# calc_disc_dist_quantile

*Calculate the quantiles of a probability distribution based on the vector of probabilities and time data (e.g. time since infection)*

---

## Description

This function can be used in cases where the data on a fitted distribution is not openly available and the summary statistics of the distribution are not reported so the data are scraped from the plot and the quantiles are needed in order use the `extract_param()` function.

## Usage

```
calc_disc_dist_quantile(prob, days, quantile)
```

## Arguments

| | |
|---|---|
| `prob` | A `numeric` vector of probabilities. |
| `days` | A `numeric` vector of days. |
| `quantile` | A single `numeric` or vector of `numerics` specifying which quantiles to extract from the distribution. |

## Value

A named vector of quantiles.

## Examples

```
prob <- dgamma(seq(0, 10, length.out = 21), shape = 2, scale = 2)
days <- seq(0, 10, 0.5)
quantiles <- c(0.025, 0.975)
calc_disc_dist_quantile(prob = prob, days = days, quantile = quantiles)
```

## convert_params_to_summary_stats

*Convert the parameter(s) of a distribution to summary statistics*

### Description

Convert the parameters for a range of distributions to a number of summary statistics. All summary statistics are calculated analytically given the parameters.

### Usage

```
convert_params_to_summary_stats(x, ...)

## S3 method for class 'character'
convert_params_to_summary_stats(
  x = c("lnorm", "gamma", "weibull", "nbinom", "geom"),
  ...
)

## S3 method for class 'epiparameter'
convert_params_to_summary_stats(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An R object. |
| `...` | <`dynamic-dots`> `Numeric` named parameter(s) used to convert to summary statistics. An example is the `meanlog` and `sdlog` parameters of the lognormal (`lnorm`) distribution. |

### Details

The distribution names and parameter names follow the style of distributions in R, for example the lognormal distribution is `lnorm`, and its parameters are `meanlog` and `sdlog`.

### Value

A list of eight elements including: mean, median, mode, variance (`var`), standard deviation (`sd`), coefficient of variation (`cv`), skewness, and excess kurtosis (`ex_kurtosis`).

### See Also

`convert_summary_stats_to_params()`

## Examples

```
# example using characters
convert_params_to_summary_stats("lnorm", meanlog = 1, sdlog = 2)
convert_params_to_summary_stats("gamma", shape = 1, scale = 1)
convert_params_to_summary_stats("nbinom", prob = 0.5, dispersion = 2)

# example using <epiparameter>
epiparameter <- epiparameter_db(single_epiparameter = TRUE)
convert_params_to_summary_stats(epiparameter)

# example using <epiparameter> and specifying parameters
epiparameter <- epiparameter_db(
  disease = "Influenza",
  author = "Virlogeux",
  subset = prob_dist == "weibull"
)
convert_params_to_summary_stats(epiparameter[[2]], shape = 1, scale = 1)
```

---

## convert_summary_stats_to_params

*Convert the summary statistics of a distribution to parameters*

---

### Description

Convert the summary statistics for a range of distributions to the distribution's parameters. Most summary statistics are calculated analytically given the parameters. An exception is the Weibull distribution which uses a root finding numerical method.

### Usage

```
convert_summary_stats_to_params(x, ...)

## S3 method for class 'character'
convert_summary_stats_to_params(
  x = c("lnorm", "gamma", "weibull", "nbinom", "geom"),
  ...
)

## S3 method for class 'epiparameter'
convert_summary_stats_to_params(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An R object. |
| `...` | <`dynamic-dots`> `Numeric` named summary statistics used to convert to parameter(s). An example is the `mean` and `sd` summary statistics for the lognormal (`lnorm`) distribution. |

## Details

Summary statistics should be named accordingly (case-sensitive):

- mean: `mean`
- median: `median`
- mode: `mode`
- variance: `var`
- standard deviation: `sd`
- coefficient of variation: `cv`
- skewness: `skewness`
- excess kurtosis: `ex_kurtosis`

**Note**: Not all combinations of summary statistics can be converted into distribution parameters. In this case the function will error stating that the parameters cannot be calculated from the given input.

The distribution names and parameter names follow the style of distributions in R, for example the lognormal distribution is `lnorm`, and its parameters are `meanlog` and `sdlog`.

## Value

A list of either one or two elements (depending on how many parameters the distribution has).

## See Also

`convert_params_to_summary_stats()`

## Examples

```
# examples using characters
convert_summary_stats_to_params("lnorm", mean = 1, sd = 1)
convert_summary_stats_to_params("weibull", mean = 2, var = 2)
convert_summary_stats_to_params("geom", mean = 2)

# examples using <epiparameter>
epiparameter <- epiparameter_db(single_epiparameter = TRUE)
convert_summary_stats_to_params(epiparameter)

# example using <epiparameter> and specifying summary stats
epiparameter$summary_stats <- list()
convert_summary_stats_to_params(epiparameter, mean = 10, sd = 2)
```

## `create_citation` *Create a citation for an* `<epiparameter>` *object*

### Description

A helper function when creating an `<epiparameter>` object to create a citation list with sensible defaults, type checking and arguments to help remember which citation information is accepted in the list.

### Usage

```
create_citation(
  author = utils::person(),
  year = NA_integer_,
  title = NA_character_,
  journal = NA_character_,
  doi = NA_character_,
  pmid = NA_integer_
)
```

### Arguments

| | |
|---|---|
| `author` | Either a `<person>`, a `character` string, or a vector or list of `characters` in the case of multiple authors. Specify the full name (`"<given name>" "<family name>"`). When using `characters` make sure the name can be converted to a `<person>` (see `as.person()`). Use white space separation between names. Multiple names can be stored within a single `<person>` (see `person()`). |
| `year` | A `numeric` of the year of publication. |
| `title` | A `character` string with the title of the article that published the epidemiological parameters. |
| `journal` | A `character` string with the name of the journal that published the article that published the epidemiological parameters. This can also be a pre-print server, e.g., medRxiv. |
| `doi` | A `character` string of the Digital Object Identifier (DOI) assigned to papers which are unique to each paper. |
| `pmid` | A `character` string with the PubMed unique identifier number (PMID) assigned to papers to give them a unique identifier within PubMed. |

### Details

This function acts as a wrapper around `bibentry()` to create citations for sources reporting epidemiological parameters.

### Value

A `<bibentry>` object of the citation

### Examples

```
create_citation(
  author = person(given = "John", family = "Smith"),
  year = 2002,
  title = "COVID-19 incubation period",
  journal = "Epi Journal",
  doi = "10.19832/j.1366-9516.2012.09147.x"
)
```

---

## `create_metadata` *Specify metadata associated with data set*

---

### Description

A helper function when creating an `<epiparameter>` object to create a metadata list with sensible defaults, type checking and arguments to help remember metadata list structure (element names).

### Usage

```
create_metadata(
  units = NA_character_,
  sample_size = NA_integer_,
  region = NA_character_,
  transmission_mode = NA_character_,
  vector = NA_character_,
  extrinsic = FALSE,
  inference_method = NA_character_
)
```

### Arguments

| | |
|---|---|
| `units` | A `character` for the units of the epidemiological parameter. |
| `sample_size` | The sample of the data used to fit the delay distribution. This is usually the number of people with data on a primary and possibly secondary event of interest. In cases where the sample size is not stated NA can be used. |
| `region` | The geographical location the data was collected. This can either be given at sub-national, national, continental. Multiple nested regions can be given and are comma separated. When the region is not specified NA can be given. |
| `transmission_mode` | A character string specifying how the pathogen is transmitted. This information is used to determine whether the epidemiological parameters are from a vector-borne disease (i.e. is transmitted between humans through an intermediate vector), this is specified by `transmission_mode = "vector_borne"`. |
| `vector` | The name of the vector transmitting the vector-borne disease. This can be a common name, or a latin binomial name of a specific vector species. Both the common name and taxonomic name can be given with one given in parentheses. When a disease is not vector-borne NA should be given. |

| | |
|---|---|
| `extrinsic` | A boolean value defining whether the data entry is an extrinsic delay distribution, such as the extrinsic incubation period. This field is required because intrinsic and extrinsic delay distributions are stored as separate entries in the database and can be linked. When the disease is not vector-borne FALSE should be given. See Details for explanation of extrinsic distribution. |
| `inference_method` | The type of inference used to fit the delay distribution to the data. Abbreviations of model fitting techniques can be specified as long as they are non-ambiguous. This field is only used to determine whether the uncertainty intervals possibly specified in the other fields are: confidence intervals (in the case of maximum likelihood), or credible intervals (in the case of bayesian inference). Uncertainty bounds for another types of inference methods, or if the inference method is unstated are assumed to be confidence intervals. When the inference method is unknown or a disease does not have a probability distribution NA can be given. |

## Details

In vector-borne diseases the transmissibility of a disease is dependent on both the time taken for a host (i.e. human) to become infectious, but also on the time it takes the vector to become infectious. Therefore, the extrinsic delay, in which the vector has been infected by is not yet infectious can have a role in the spread of a disease.

## Value

A named list containing information on the sample size of the study, geography, whether the disease is vector-borne and if so whether it is the intrinsic or extrinsic distribution as well as method of distribution parameter estimation.

## Examples

```
# it will automatically populate the fields with defaults if left empty
create_metadata()

# supplying each field
create_metadata(
  units = "days",
  sample_size = 10,
  region = "UK",
  transmission_mode = "vector_borne",
  vector = "mosquito",
  extrinsic = FALSE,
  inference_method = "MLE"
)
```

---

# `create_method_assess` *Specify methodological aspects of distribution fitting*

---

## Description

A helper function when creating an `<epiparameter>` object to create a method assessment list with sensible defaults, type checking and arguments to help remember which method assessments can be accepted in the list.

## Usage

```
create_method_assess(
  censored = NA,
  right_truncated = NA,
  phase_bias_adjusted = NA
)
```

## Arguments

| | |
|---|---|
| `censored` | A boolean `logical` whether the study used single or double interval censoring in the methods to infer the delay distribution |
| `right_truncated` | A boolean `logical` whether the study used right- truncation in the methods to infer the delay distribution |
| `phase_bias_adjusted` | A boolean `logical` whether the study adjusted for phase bias in the methods to infer the delay distribution |

## Details

Currently, the method assessment focuses on common methodological aspects of delay distributions (e.g. incubation period, serial interval, etc.), and does not currently take into account methodological aspects which may be important when fitting offspring distributions to data on disease (super)spreading.

## Value

A named list with three elements

## Examples

```
create_method_assess(
  censored = FALSE,
  right_truncated = FALSE,
  phase_bias_adjusted = FALSE
)
```

`create_prob_distribution`
*Create a distribution object*

## Description

Creates an S3 class holding the distribution and parameters from the probability distribution name, its parameters and distribution truncation and discretisation.

The class holding the distribution depends on whether it is a discretised distribution. For continuous and discrete distributions S3 classes from the **distributional** package are used, for discretised continuous distributions the an S3 class from the **distcrete** package is used.

For details on the properties of the distribution classes from each respective package see their documentation (either `?distributional` or `?distcrete`)

## Usage

```
create_prob_distribution(
  prob_distribution,
  prob_distribution_params,
  discretise = FALSE,
  truncation = NA,
  ...
)
```

## Arguments

| | |
|---|---|
| `prob_distribution` | A `character` string specifying the probability distribution. This should match the R naming convention of probability distributions (e.g. lognormal is `lnorm`, negative binomial is `nbinom`, and geometric is `geom`). |
| `prob_distribution_params` | A named vector of probability distribution parameters. |
| `discretise` | A boolean `logical` whether the distribution is discretised. Default is `FALSE` which assumes a continuous probability distribution. |
| `truncation` | A `numeric` specifying the truncation point if the inferred distribution was truncated, `NA` if not or unknown. |
| `...` | dots Extra arguments to be passed to **distributional** or **distcrete** functions that construct the S3 distribution objects. To see which arguments can be adjusted for discretised distributions see `distcrete::distcrete()`, for other distributions see the `?distributional` documentation and find the specific distribution constructor function, e.g. for the Gamma distribution see `distributional::dist_gamma()`. |

## Details

Truncation is enabled only for continuous distributions as there is no truncation implemented in **distcrete**.

By default the discretisation of continuous distributions uses a discretisation interval (`interval`) of 1. If the unit of the distribution is days, then this will be discretised by day. The endpoint weighting (`w`) for the discretisation is 1. `w` can be `[0,1]`. For more information please see `distcrete::distcrete()`.

## Value

An S3 class containing the probability distribution or a `character` string if the parameters of the probability distribution are unknown.

## Examples

```
# example with continuous distribution without truncation
create_prob_distribution(
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 1, scale = 1),
  discretise = FALSE,
  truncation = NA
)

# example with continuous distribution with truncation
create_prob_distribution(
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 1, scale = 1),
  discretise = FALSE,
  truncation = 10
)

# example with discrete distribution
create_prob_distribution(
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 1, scale = 1),
  discretise = TRUE,
  truncation = NA
)

# example passing extra arguments to distcrete
create_prob_distribution(
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 1, scale = 1),
  discretise = TRUE,
  truncation = NA,
  w = 0.5
)
```

## `create_region` *Specify the geography of the data entry*

### Description

The geography of the data set can be a single geographical region at either continent, country, region or city level. By specifying the level of the geography the other fields may be deduced.

### Usage

```
create_region(
  continent = NA_character_,
  country = NA_character_,
  region = NA_character_,
  city = NA_character_
)
```

### Arguments

| | |
|---|---|
| `continent` | A `character` string specifying the continent. |
| `country` | A `character` string specifying the country. |
| `region` | A `character` string specifying the region. |
| `city` | A `character` string specifying the city. |

### Value

A named list.

### Examples

```
create_region(country = "UK")
```

## `create_summary_stats` *Specify reported summary statistics*

### Description

A helper function when creating an `<epiparameter>` object to create a summary statistics list with sensible defaults, type checking and arguments to help remember which summary statistics can be accepted in the list.

## Usage

```
create_summary_stats(
  mean = NA_real_,
  mean_ci_limits = c(NA_real_, NA_real_),
  mean_ci = NA_real_,
  sd = NA_real_,
  sd_ci_limits = c(NA_real_, NA_real_),
  sd_ci = NA_real_,
  median = NA_real_,
  median_ci_limits = c(NA_real_, NA_real_),
  median_ci = NA_real_,
  dispersion = NA_real_,
  dispersion_ci_limits = c(NA_real_, NA_real_),
  dispersion_ci = NA_real_,
  lower_range = NA_real_,
  upper_range = NA_real_,
  quantiles = NA_real_
)
```

## Arguments

| | |
|---|---|
| `mean` | A `numeric` of the mean (expectation) of the probability distribution. |
| `mean_ci_limits` | A `numeric` vector of length two of the confidence interval around the mean. |
| `mean_ci` | A `numeric` specifying the confidence interval width, e.g. `95` would be the 95% CI |
| `sd` | A `numeric` of the standard deviation of the probability distribution. |
| `sd_ci_limits` | A `numeric` vector of length 2 of the confidence interval around the standard deviation. |
| `sd_ci` | A `numeric` specifying the confidence interval width, e.g. `95` would be 95% confidence interval. |
| `median` | A `numeric` of the median of the probability distribution. |
| `median_ci_limits` | A `numeric` vector of length two of the confidence interval around the median. |
| `median_ci` | A `numeric` specifying the confidence interval width of the median. |
| `dispersion` | A `numeric` of the dispersion of the probability distribution. **Important** this is the dispersion for probability distributions that are not usually parameterised by a dispersion parameter, for example a lognormal distribution. If a probability distribution is usually parameterised with a dispersion parameter, e.g. negative binomial distribution, then this should be considered a parameter and not a summary statistic and should go in the `prob_distribution` argument when constructing an `<epiparameter>` object with `epiparameter()` (see `create_prob_distribution()`). |
| `dispersion_ci_limits` | A `numeric` vector of length 2 of the confidence interval around the dispersion. |
| `dispersion_ci` | A `numeric` specifying the confidence interval width, e.g. `95` would be 95% confidence interval. |

| | |
|---|---|
| `lower_range` | The lower range of the data, used to infer the parameters of the distribution when not provided. |
| `upper_range` | The upper range of the data, used to infer the parameters of the distribution when not provided. |
| `quantiles` | A `numeric` vector of the quantiles for the distribution. If quantiles are not provided a default empty vector with the 2.5th, 5th, 25th, 75th, 95th, 97.5th quantiles are supplied. |

## Value

A list of summary statistics. The output list has element names equal to the function arguments:

```
$mean
$mean_ci_limits
$mean_ci
$sd
$sd_ci_limits
$sd_ci
$median
$median_ci_limits
$median_ci
$dispersion
$dispersion_ci_limits
$dispersion_ci
$lower_range
$upper_range
$quantiles
```

## Examples

```
# mean and standard deviation
create_summary_stats(mean = 5, sd = 2)

# mean and standard deviation with uncertainty
create_summary_stats(
  mean = 4,
  mean_ci_limits = c(2.1, 5.7),
  mean_ci = 95,
  sd = 0.7,
  sd_ci_limits = c(0.3, 1.1),
  sd_ci = 95
)

# median and range
create_summary_stats(
  median = 5,
  lower_range = 1,
  upper_range = 13
)
```

## create_uncertainty *Specify distribution parameter uncertainty*

### Description

A helper function when creating uncertainty for the parameters of the distribution for the `<epiparameter>` object.

### Usage

```
create_uncertainty(ci_limits = NA_real_, ci, ci_type)
```

### Arguments

| | |
|---|---|
| `ci_limits` | A numeric vector of length two with the lower and upper bound of the confidence interval or credible interval. |
| `ci` | A numeric specifying the interval for the ci, e.g. 95 is 95% ci. |
| `ci_type` | A character string, either `"confidence interval"` or `"credible interval"`. |

### Value

List of three elements:

1. `$ci_limits` is the upper and lower bounds of the CI (either confidence interval or credible interval) (i.e. a two element numeric vector).
2. `$ci` the interval (e.g. 95 is 95% CI) given by a single numeric.
3. `$ci_type` a character string specifying the type of uncertainty (can be either `"confidence interval"` or `"credible interval"`).

### Examples

```
# example with uncertainty for a single parameter
create_uncertainty(
  ci_limits = c(1, 3),
  ci = 95,
  ci_type = "confidence interval"
)

# example for multiple parameters
# lengh of list should match number of parameters
list(
  shape = create_uncertainty(
    ci_limits = c(1, 3),
    ci = 95,
    ci_type = "confidence interval"
  ),
  scale = create_uncertainty(
    ci_limits = c(2, 4),
    ci = 95,
    ci_type = "confidence interval"
  )
)

# example with unknown uncertainty
# the function can be called without arguments
create_uncertainty()
# or give NA as the first argument
create_uncertainty(NA)
```

---

## discretise — *Discretises a continuous distribution in an* `<epiparameter>` *object*

---

### Description

Discretises a continuous distribution in an `<epiparameter>` object

### Usage

```
discretise(x, ...)

discretize(x, ...)

## S3 method for class 'epiparameter'
discretise(x, ...)

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

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Extra arguments to be passed to the method. |

### Details

Converts the S3 distribution object in an `<epiparameter>` from continuous (using an object from the `{distributional}` package) to a discretised distribution (using an object from the `{distcrete}` package).

### Value

An `<epiparameter>` object.

## Examples

```
ebola_incubation <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
discretise(ebola_incubation)
```

---

## epiparameter *Create an* <epiparameter> *object*

---

### Description

The <epiparameter> class is used to store epidemiological parameters for a single disease. These epidemiological parameters cover a variety of aspects including delay distributions (e.g. incubation periods and serial intervals, among others) and offspring distributions.

The <epiparameter> object is the functional unit provided by {epiparameter} to plug into epidemiological pipelines. Obtaining an <epiparameter> object can be achieved in two main ways:

1. The epidemiological distribution is stored in the {epiparameter} library and can be accessed by `epiparameter_db()`.
2. The alternative method is when you have information (e.g. disease and distribution parameter estimates) and would like to input this into an <epiparameter> object in order to work in existing analysis pipelines. This is where the `epiparameter()` function can be used to fill out each field for which information is known.

### Usage

```
epiparameter(
  disease,
  pathogen = NA_character_,
  epi_name,
 prob_distribution = create_prob_distribution(prob_distribution = NA_character_),
  uncertainty = create_uncertainty(),
  summary_stats = create_summary_stats(),
  citation = create_citation(),
  metadata = create_metadata(),
  method_assess = create_method_assess(),
  notes = NULL,
  auto_calc_params = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `disease` | A `character` string with name of the infectious disease. |
| `pathogen` | A `character` string with the name of the causative agent of disease, or `NA` if not known. |
| `epi_name` | A `character` string with the name of the epidemiological parameter type. |
| `prob_distribution` | An S3 class containing the probability distribution or a character string if the parameters of the probability distribution are unknown but the name of the distribution is known, or `NA` if the distribution name and parameters are unknown. Use `create_prob_distribution()` to create `prob_distribution`. |
| `uncertainty` | A list of named vectors with the uncertainty around the probability distribution parameters. If uncertainty around the parameter estimates is unknown use `create_uncertainty()` (which is the argument default) to create a list with the correct names with missing values. |
| `summary_stats` | A list of summary statistics, use `create_summary_stats()` to create list. This list can include summary statistics about the inferred distribution such as it's mean and standard deviation, quantiles of the distribution, or information about the data used to fit the distribution such as lower and upper range. The summary statistics can also include uncertainty around metrics such as confidence interval around mean and standard deviation. |
| `citation` | A `<bibentry>` with the citation of the source of the data or the paper that inferred the distribution parameters, use `create_citation()` to create citation. |
| `metadata` | A list of metadata, this can include: units, sample size, the transmission mode of the disease (e.g. is it vector-borne or directly transmitted), etc. It is assumed that the disease is not vector-borne and that the distribution is intrinsic (e.g. not an extrinsic delay distribution such as extrinsic incubation period) unless `transmission_mode = "vector_borne"` is contained in the metadata. Use `create_metadata()` to create metadata. |
| `method_assess` | A list of methodological aspects used when fitting the distribution, use `create_method_assess()` to create method assessment. |
| `notes` | A `character` string with any additional information about the data, inference method or disease. |
| `auto_calc_params` | A boolean `logical` determining whether to try and calculate the probability distribution parameters from summary statistics if distribution parameters are not provided. Default is `TRUE`. In the case when sufficient summary statistics are provided and the parameter(s) of the distribution are not, the `.calc_dist_params()` function is called to calculate the parameters and add them to the `epiparameter` object created. |
| `...` | dots Extra arguments to be passed to internal functions. This is most commonly used to pass arguments to `distcrete::distcrete()` that construct the discretised distribution S3 object. To see which arguments can be adjusted for discretised distributions see `distcrete::distcrete()`. |

## Details

Accepted <`epiparameter`> distribution parameterisations are:

- Gamma must be either 'shape' and 'scale' or 'shape' and 'rate'
- Weibull must be 'shape' and 'scale'
- Lognormal must be 'meanlog' and 'sdlog' or 'mu' and 'sigma'
- Negative Binomial must be either 'mean' and 'dispersion' or 'n' and 'p'
- Geometric must be either 'mean' or 'prob'
- Poisson must be 'mean'

## Value

An <`epiparameter`> object.

## Examples

```
# minimal input required for `epiparameter`
ebola_incubation <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)

# minimal input required for discrete `epiparameter`
ebola_incubation <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1),
    discretise = TRUE
  )
)

# example with more fields filled in
ebola_incubation <- epiparameter(
  disease = "ebola",
  pathogen = "ebola_virus",
  epi_name = "incubation",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1),
    discretise = FALSE,
    truncation = NA
  ),
  uncertainty = list(
    shape = create_uncertainty(),
    scale = create_uncertainty()
  ),
  summary_stats = create_summary_stats(mean = 2, sd = 1),
  citation = create_citation(
    author = person(given = "John", family = "Smith"),
    year = 2002,
    title = "COVID-19 incubation period",
    journal = "Epi Journal",
    doi = "10.19832/j.1366-9516.2012.09147.x"
  ),
  metadata = create_metadata(
    units = "days",
    sample_size = 10,
    region = "UK",
    transmission_mode = "natural_human_to_human",
    inference_method = "MLE"
  ),
  method_assess = create_method_assess(
    censored = TRUE
  ),
  notes = "No notes"
)
```

## epiparameter_db *Create* <epiparameter> *object(s) directly from the epiparameter library (database)*

### Description

Extract <epiparameter> object(s) directly from the library of epidemiological parameters. The epiparameter library of epidemiological parameters is compiled from primary literature sources. The list output from epiparameter_db() can be subset by the data it contains, for example by: disease, pathogen, epidemiological distribution, sample size, region, etc.

If a distribution from a specific study is required, the author argument can be specified.

Multiple entries (<epiparameter> objects) can be returned, use the arguments to subset entries and use single_epiparameter = TRUE to force a single <epiparameter> to be returned.

### Usage

```
epiparameter_db(
  disease = "all",
  pathogen = "all",
  epi_name = "all",
  author = NULL,
  subset = NULL,
  single_epiparameter = FALSE
)
```

## Arguments

| | |
|---|---|
| `disease` | A `character` string specifying the disease. |
| `pathogen` | A `character` string specifying the pathogen. |
| `epi_name` | A `character` string specifying the epidemiological parameter. See details for full list of epidemiological distributions. |
| `author` | A `character` string specifying the author of the study reporting the distribution. Only the first author will be matched. It is recommended to use the family name as first names may or may not be initialised. |
| `subset` | Either `NULL` or a valid R expressions that evaluates to logicals to subset the list of `<epiparameter>`, or a function that can be applied over a list of `<epiparameter>` objects. Subsetting (using `subset`) can be combined with the subsetting done with the `disease` and `epi_name` arguments (and `author` if specified). If left as `NULL` (default) no subsetting is carried out. The `subset` argument is similar to subsetting a `<data.frame>`, but the difference is that fixed comparisons and not vectorised comparisons are needed. For example `sample_size > 10` is a valid subset expression, but `sample_size == max(sample_size)`, which would be a valid subset expression for a `<data.frame>` does not work. The vectorised expression will often not error, but will likely return unexpected results. For the `sample_size == max(sample_size)` example it will always return `TRUE` (except for `NA`s) as it is a single numeric so will be equal to it's max value. The expression should be specified without using the data object name (e.g. `df$var`) and instead just `var` should be supplied. In other words, this argument uses non-standard evaluation, just as the `subset` argument in `subset()`, and is similar to `<data-masking>` used by the `dplyr` package. |
| `single_epiparameter` | A boolean `logical` determining whether a single `<epiparameter>` or multiple entries from the library can be returned if matched by the other arguments (`disease`, `epi_name`, `author`). This argument is used to prevent multiple sets of parameters being returned when only one is wanted. **Note**: If multiple entries match the arguments supplied and `single_epiparameter = TRUE` then the `<epiparameter>` that is parameterised (and accounts for truncation if available) and has the largest sample size will be returned (see `is_parameterised()`). If multiple entries are equal after this sorting the first entry will be returned. |

## Details

`disease`, `epi_name` and `author` are given as individual arguments as these are the most common variables to subset the parameter library by. The `subset` argument facilitates all other subsetting of rows to select the `<epiparameter>` object(s) desired. To subset based on multiple variables separate each expression with `&`.

List of epidemiological parameters:

- "all" (default, returns all entries in library)
- "incubation period"

- "onset to hospitalisation"
- "onset to death"
- "serial interval"
- "generation time"
- "offspring distribution"
- "hospitalisation to death"
- "hospitalisation to discharge"
- "notification to death"
- "notification to discharge"
- "onset to discharge"
- "onset to ventilation"

## Value

An `<epiparameter>` object or list of `<epiparameter>` objects.

## Examples

```
epiparameter_db(disease = "influenza", epi_name = "serial_interval")

# example using custom subsetting
eparam <- epiparameter_db(
  disease = "SARS",
  epi_name = "offspring_distribution",
  subset = sample_size > 40
)

# example using functional subsetting
eparam <- epiparameter_db(
  disease = "COVID-19",
  epi_name = "incubation_period",
  subset = is_parameterised
)

# example forcing a single <epiparameter> to be returned
eparam <- epiparameter_db(
  disease = "SARS",
  epi_name = "offspring_distribution",
  single_epiparameter = TRUE
)
```

## epiparameter_distribution_functions

*PDF, CDF, PMF, quantiles and random number generation for* `<epiparameter>` *objects*

### Description

The `<epiparameter>` object holds a probability distribution which can either be a continuous or discrete distribution. These are the density, cumulative distribution, quantile and random number generation functions. These operate on any distribution that can be included in an `<epiparameter>` object.

### Usage

```
## S3 method for class 'epiparameter'
density(x, at, ...)

## S3 method for class 'epiparameter'
cdf(x, q, ..., log = FALSE)

## S3 method for class 'epiparameter'
quantile(x, p, ...)

## S3 method for class 'epiparameter'
generate(x, times, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `at` | The quantiles to evaluate at. |
| `...` | dots Extra arguments to be passed to the method. |
| `q` | The quantiles to evaluate at. |
| `log` | If `TRUE`, probabilities will be given as log probabilities. |
| `p` | The probabilities to evaluate at. |
| `times` | The number of random samples. |

### Value

`numeric` vector.

### Examples

```
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)

# example of each distribution method for an `epiparameter` object
stats::density(ep, at = 1)
distributional::cdf(ep, q = 1)
stats::quantile(ep, p = 0.2)
distributional::generate(ep, times = 10)
```

## `epiparameter_options` *Package options*

### Description

Options to modify the printing of **epiparameter** objects. Currently options are only used to modify the printing of the `<multi_epiparameter>` class.

### Usage

```
epiparameter_options
```

### Format

An object of class `list` of length 2.

### Details

Options are set with `options()` and retrieved with `getOption()`.

If options are changed the **epiparameter** package will need to be reloaded for new options to be taken into account. Options can be set in the `.Rprofile` to persist across R sessions.

## `epireview_core_cols` *A vector of `character` strings with the core column names of the epidemiological parameter data exported by the **epireview** R package.*

### Description

A vector of `character` strings with the core column names of the epidemiological parameter data exported by the **epireview** R package.

### Usage

```
epireview_core_cols
```

## Format

`epireview_core_cols`:

A `character` vector with 58 elements.

The data is taken as the intersection of the column names of each of the disease parameter tables in the **epireview** R package.

## Source

https://github.com/mrc-ide/epireview

---

`extract_param` *Calculate the parameters of a parametric probability distribution from reported values of percentiles, or median and range*

---

## Description

Summary data of distributions, as provided by reports and meta-analyses, can be used to extract the parameters of a chosen distribution. Currently available distributions are: lognormal, gamma, Weibull and normal. Extracting from a lognormal returns the meanlog and sdlog parameters, extracting from the gamma and Weibull returns the shape and scale parameters, and extracting from the normal returns the mean and sd parameters.

## Usage

```
extract_param(
  type = c("percentiles", "range"),
  values,
  distribution = c("lnorm", "gamma", "weibull", "norm"),
  percentiles,
  samples,
  control = list(max_iter = 1000, tolerance = 1e-05)
)
```

## Arguments

| | |
|---|---|
| `type` | A `character` defining whether summary statistics based around `percentiles` (default) or `range`. |
| `values` | A `vector`. If `type = percentiles`: `c(percentile_1, percentile_2)`; and if `type = range`: `c(median, min, max)`. |
| `distribution` | A `character` specifying distribution to use. Default is `lnorm`; also takes `gamma`, `weibull` and `norm`. |
| `percentiles` | A `vector` with two elements specifying the percentiles defined in `values` if using `type = "percentiles"`. Percentiles should be specified between 0 and 1. For example 2.5th and 97.5th percentile is given as `c(0.025, 0.975)`. |
| `samples` | A `numeric` specifying the sample size if using `type = "range"`. |

| | |
|---|---|
| `control` | A named list containing options for the optimisation. List element `$max_iter` is a `numeric` specifying the maximum number of times the parameter extraction will run optimisation before returning result early. This prevents overly long optimisation loops if optimisation is unstable and does not converge over multiple iterations. Default is 1000 iterations. List element `$tolerance` is passed to `.check_optim_conv()` for tolerance on parameter convergence over iterations of optimisation. Elements of in the control list are not passed to `optim()`. |

## Details

For `gamma`, `lnorm` and `weibull`, `extract_param()` works only for strictly positive values at the percentiles of a distribution or the median and range of data (numerics supplied to the `values` argument). This means that negative values at the lower percentile or lower range will not work with this function although they may present themselves in epidemiological data (e.g. negative serial interval). For the `norm` distribution negative values are allowed.

## Value

A named `numeric` vector with the parameter values of the distribution. If the `distribution` = `lnorm` then the parameters returned are the meanlog and sdlog; if the `distribution` = `gamma` or `distribution` = `weibull` then the parameters returned are the shape and scale; if `distribution` = `norm` then the parameters returned are mean and sd.

## Author(s)

Adam Kucharski, Joshua W. Lambert

## Examples

```
# set seed to control for stochasticity
set.seed(1)

# extract parameters of a lognormal distribution from the 75 percentiles
extract_param(
  type = "percentiles",
  values = c(6, 13),
  distribution = "lnorm",
  percentiles = c(0.125, 0.875)
)

# extract parameters of a gamma distribution from median and range
extract_param(
  type = "range",
  values = c(10, 3, 18),
  distribution = "gamma",
  samples = 20
)
```

## family.epiparameter — *Family method for the* `<epiparameter>` *class*

### Description

The `family()` function is used to extract the distribution names from objects from `{distributional}` and `{distcrete}`. This method provides the same interface for `<epiparameter>` objects to give consistent output irrespective of the internal distribution class.

### Usage

```
## S3 method for class 'epiparameter'
family(object, ..., base_dist = FALSE)
```

### Arguments

| | |
|---|---|
| `object` | An `<epiparameter>` object. |
| `...` | further arguments passed to methods. |
| `base_dist` | A boolean `logical` for whether to return the name of a transformed distribution (e.g. `"mixture"` or `"truncated"`) or the underlying distribution type (e.g. `"gamma"` or `"lnorm"`). Default is `FALSE`. |

### Value

A character string with the name of the distribution, or `NA` when the `<epiparameter>` object is unparameterised.

### Examples

```
# example with continuous distribution
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
family(ep)

# example with discretised distribution
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 1),
    discretise = TRUE
  )
)
family(ep)
```

## format.epiparameter *Format method for* <epiparameter> *class*

### Description

Format method for <epiparameter> class

### Usage

```
## S3 method for class 'epiparameter'
format(x, ...)
```

### Arguments

| | |
|---|---|
| x | An <epiparameter> object. |
| ... | dots Extra arguments to be passed to the method. |

### Value

Invisibly returns an <epiparameter>. Called for printing side-effects.

### Examples

```
epiparameter <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
format(epiparameter)
```

## get_citation *Get citation from an object*

### Description

Extract the citation stored in an R object.

### Usage

```
get_citation(x, ...)
```

### Arguments

| | |
|---|---|
| x | an object used to select a method. |
| ... | dots Extra arguments to be passed to the method. |

### Value

A `<bibentry>` object or list of `<bibentry>` objects.

## get_citation.epiparameter *Get citation from an `<epiparameter>` object*

### Description

Extract the citation stored in an `<epiparameter>` object.

### Usage

```
## S3 method for class 'epiparameter'
get_citation(x, ...)
```

### Arguments

| | |
|---|---|
| x | An `<epiparameter>` object. |
| ... | dots Not used, extra arguments supplied will cause a warning. |

### Value

A `<bibentry>` object.

### Examples

```
# example with <epiparameter>
ep <- epiparameter_db(single_epiparameter = TRUE)
get_citation(ep)

# example returning bibtex format
ep <- epiparameter_db(disease = "COVID-19", single_epiparameter = TRUE)
cit <- get_citation(ep)
format(cit, style = "bibtex")
```

---

## get_citation.multi_epiparameter

*Get citation from a list of* `<epiparameter>` *objects*

---

### Description

Extract the citation stored in a list of `<epiparameter>` objects.

### Usage

```
## S3 method for class 'multi_epiparameter'
get_citation(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Value

A `<bibentry>` object containing multiple references. The length of output `<bibentry>` is equal to the length of the list of `<epiparameter>` objects supplied.

### Examples

```
# example with list of <epiparameter>
db <- epiparameter_db()
get_citation(db)
```

## `get_parameters` *Get parameters from an object*

### Description

Extract the parameters stored in an R object.

### Usage

```
get_parameters(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | an object used to select a method. |
| `...` | dots Extra arguments to be passed to the method. |

### Value

A named vector of parameters or `NA` when the `<epiparameter>` object is unparameterised, or a list where each element is a named vectors or `NA`.

## `get_parameters.epiparameter` *Get parameters from an `<epiparameter>` object*

### Description

Extract the parameters of the distribution stored in an `<epiparameter>` object.

### Usage

```
## S3 method for class 'epiparameter'
get_parameters(x, ...)

## S3 method for class 'multi_epiparameter'
get_parameters(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Details

The `<epiparameter>` object can be unparameterised in which it lacks a probability distribution or parameters of a probability distribution, in this case the `get_parameters.epiparameter()` method will return `NA`.

## Value

A named vector of parameters or `NA` when the `<epiparameter>` object is unparameterised.

## Examples

```
ep <- epiparameter_db(
  disease = "COVID-19",
  epi_name = "serial interval",
  single_epiparameter = TRUE
)
get_parameters(ep)
```

---

`is_continuous` *Check if distribution in* `<epiparameter>` *is continuous*

---

## Description

Check if distribution in `<epiparameter>` is continuous

## Usage

```
is_continuous(x)
```

## Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |

## Details

The `<epiparameter>` class can hold `<distribution>` and `<distcrete>` probability distribution objects from the **distributional** package and the **distcrete** package, respectively. `<distribution>` objects can be continuous or discrete distributions (e.g. gamma or negative binomial), and all `<distcrete>` objects are discrete.

## Value

A boolean `logical`.

## Examples

```
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 1)
  )
)
is_continuous(ep)
is_continuous(discretise(ep))

ep <- epiparameter(
  disease = "ebola",
  epi_name = "offspring distribution",
  prob_distribution = create_prob_distribution(
    prob_distribution = "nbinom",
    prob_distribution_params = c(mean = 2, dispersion = 0.5)
  )
)
is_continuous(ep)
```

---

## `is_epiparameter` *Check object is an* `<epiparameter>`

---

### Description

Check object is an `<epiparameter>`

### Usage

```
is_epiparameter(x)
```

### Arguments

| | |
|---|---|
| `x` | An R object. |

### Value

A boolean logical, `TRUE` if the object is an `<epiparameter>` and `FALSE` if not.

### Examples

```
ep <- epiparameter(
  disease = "ebola",
  epi_name = "serial_interval",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)

is_epiparameter(ep)

false_ep <- list(
  disease = "ebola",
  epi_name = "serial_interval",
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 1, scale = 1)
)
```

```
is_epiparameter(false_ep)
```

## is_epiparameter_params *Check whether the vector of parameters for the probability distribution are in the set of possible parameters used in the epiparameter package*

### Description

Check whether the vector of parameters for the probability distribution are in the set of possible parameters used in the epiparameter package

### Usage

```
is_epiparameter_params(prob_distribution, prob_distribution_params)
```

### Arguments

| | |
|---|---|
| prob_distribution | A `character` string specifying the probability distribution. This should match the R naming convention of probability distributions (e.g. lognormal is `lnorm`, negative binomial is `nbinom`, and geometric is `geom`). |
| prob_distribution_params | A named vector of probability distribution parameters. |

### Details

This check for valid parameters is independent of whether the distribution is truncated or discretised.

### Value

A boolean `logical`.

### Examples

```
is_epiparameter_params(
  prob_distribution = "gamma",
  prob_distribution_params = c(shape = 2, scale = 1)
)
```

is_parameterised *Check if* <epiparameter> *or list of* <epiparameter> *objects contains a distribution and distribution parameters*

## Description

Check if <epiparameter> or list of <epiparameter> objects contains a distribution and distribution parameters

## Usage

```
is_parameterised(x, ...)

is_parameterized(x, ...)

## S3 method for class 'epiparameter'
is_parameterised(x, ...)

## S3 method for class 'multi_epiparameter'
is_parameterised(x, ...)
```

## Arguments

| | |
|---|---|
| x | An <epiparameter> or list of <epiparameter> objects. |
| ... | dots Extra arguments to be passed to the method. |

## Value

A single boolean logical for <epiparameter> or vector of logicals equal in length to the list of <epiparameter> objects input. If the <epiparameter> object is missing either a probability distribution or parameters for the probability distribution returns FALSE, otherwise it returns TRUE.

## Examples

```
# parameterised <epiparameter>
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
is_parameterised(ep)

# unparameterised <epiparameter>
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation"
)
is_parameterised(ep)

# list of <epiparameter>
db <- epiparameter_db()
is_parameterised(db)
```

---

## `is_truncated` *Check if distribution in* `<epiparameter>` *is truncated*

---

### Description

Check if distribution in `<epiparameter>` is truncated

### Usage

```
is_truncated(x)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |

### Details

The `<epiparameter>` class can hold probability distribution objects from the `{distributional}` package or the `{distcrete}` package, however, only distribution objects from `{distributional}` can be truncated. If a `<epiparameter>` object has a `<distcrete>` object `is_truncated` will return `FALSE` by default.

### Value

A boolean `logical`.

### Examples

```
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 1)
  )
)
is_truncated(ep)

ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 1),
    truncation = 10
  )
)
is_truncated(ep)
```

## lines.epiparameter    lines() *method for* <epiparameter> *class*

### Description

`lines()` method for `<epiparameter>` class

### Usage

```
## S3 method for class 'epiparameter'
lines(x, cumulative = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `cumulative` | A boolean `logical`, default is `FALSE`. `cumulative = TRUE` plots the cumulative distribution function (CDF). |
| `...` | further arguments passed to or from other methods. |

### Value

These functions are invoked for their side effect of drawing on the active graphics device.

### Examples

```
ebola_si <- epiparameter_db(disease = "Ebola", epi_name = "serial")
plot(ebola_si[[1]])
lines(ebola_si[[2]])
```

## `mean.epiparameter` *Mean method for* `<epiparameter>` *class*

### Description

Mean method for `<epiparameter>` class

### Usage

```
## S3 method for class 'epiparameter'
mean(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Not used, extra arguments supplied will cause a warning. |

### Value

A `numeric` mean of a distribution or `NA`.

### Examples

```
ep <- epiparameter_db(
  disease = "COVID-19",
  epi_name = "incubation period",
  single_epiparameter = TRUE
)
mean(ep)
```

## `parameter_tbl` *Table of epidemiological distributions*

### Description

This function subsets the epidemiological parameter library to return only the chosen epidemiological distribution. The results are returned as a data frame containing the disease, epidemiological distribution, probability distribution, author of the study, and the year of publication.

### Usage

```
parameter_tbl(
  multi_epiparameter,
  disease = "all",
  pathogen = "all",
  epi_name = "all"
)
```

## Arguments

| | |
|---|---|
| `multi_epiparameter` | Either an `<epiparameter>` object or a list of `<epiparameter>` objects. |
| `disease` | A `character` string with name of the infectious disease. |
| `pathogen` | A `character` string with the name of the causative agent of disease, or `NA` if not known. |
| `epi_name` | A `character` string with the name of the epidemiological parameter type. |

## Value

A `<parameter_tbl>` object which is a subclass of `<data.frame>`.

## Author(s)

Joshua W. Lambert, Adam Kucharski

## Examples

```
epiparameter_list <- epiparameter_db(disease = "COVID-19")
parameter_tbl(multi_epiparameter = epiparameter_list)

# example filtering an existing list to incubation periods
epiparameter_list <- epiparameter_db(disease = "COVID-19")
parameter_tbl(
  multi_epiparameter = epiparameter_list,
  epi_name = "incubation period"
)
```

---

`plot.epiparameter` *Plot method for* `<epiparameter>` *class*

---

## Description

Plot an `<epiparameter>` object by displaying the either the probability mass function (PMF), (in the case of discrete distributions) or probability density function (PDF) (in the case of continuous distributions), or the cumulative distribution function (CDF).

## Usage

```
## S3 method for class 'epiparameter'
plot(x, cumulative = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `cumulative` | A boolean `logical`, default is `FALSE`. `cumulative = TRUE` plots the cumulative distribution function (CDF). |
| `...` | further arguments passed to or from other methods. |

### Details

By default if the `xlim` argument is not specified the distribution is plotted between day 0 and the 99th quantile of the distribution. Alternatively, a `numeric` vector of length 2 with the first and last day to plot on the x-axis can be supplied to `xlim` (through ...).

### Value

These functions are invoked for their side effect of drawing on the active graphics device.

### Author(s)

Joshua W. Lambert

### Examples

```
# plot continuous epiparameter
ep <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 2, scale = 1)
  )
)
plot(ep)

# plot different day range (x-axis)
plot(ep, xlim = c(0, 10))

# plot CDF
plot(ep, cumulative = TRUE)

# plot discrete epiparameter
ep <- discretise(ep)
plot(ep)
```

---

## plot.multi_epiparameter

`plot()` *method for* `<multi_epiparameter>` *class*

---

### Description

Plots a list of `<epiparameter>` objects by overlaying their distributions.

### Usage

```
## S3 method for class 'multi_epiparameter'
plot(x, cumulative = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `<multi_epiparameter>` object. |
| `cumulative` | A boolean `logical`, default is `FALSE`. `cumulative = TRUE` plots the cumulative distribution function (CDF). |
| `...` | further arguments passed to or from other methods. |

## Details

Unparameterised and discrete `<epiparameter>` objects are not plotted (see `is_parameterised()` and `is_continuous()`).

## Value

These functions are invoked for their side effect of drawing on the active graphics device.

## Author(s)

Joshua W. Lambert

## Examples

```
ebola_si <- epiparameter_db(disease = "Ebola", epi_name = "serial")
plot(ebola_si)
```

---

`print.epiparameter` *Print method for* `<epiparameter>` *class*

---

## Description

Print method for `<epiparameter>` class

## Usage

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

## Arguments

| | |
|---|---|
| `x` | An `<epiparameter>` object. |
| `...` | dots Extra arguments to be passed to the method. |

## Value

Invisibly returns an `<epiparameter>`. Called for side-effects.

## Examples

```
epiparameter <- epiparameter(
  disease = "ebola",
  epi_name = "incubation_period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
epiparameter
```

---

## print.multi_epiparameter

*Print method for* `<multi_epiparameter>` *class*

---

### Description

Print method for `<multi_epiparameter>` class

### Usage

```
## S3 method for class 'multi_epiparameter'
print(x, ..., n = NULL)
```

### Arguments

| | |
|---|---|
| `x` | A `<multi_epiparameter>` object. |
| `...` | dots Extra arguments to be passed to the method. |
| `n` | A `numeric` specifying how many `<epiparameter>` objects to print. This argument is passed to `head()` for `list` printing. Default is `NULL` and the number of elements to print is controlled by package `options()`. |

### Value

Invisibly returns a `<multi_epiparameter>`. Called for side-effects.

### Examples

```
# entire database
db <- epiparameter_db()
db

# a single disease
db <- epiparameter_db(disease = "Ebola")
db

# a single epi parameter
db <- epiparameter_db(epi_name = "offspring distribution")
db
```

`test_epiparameter` *Test whether an object is a valid* `<epiparameter>` *object*

## Description

Test whether an object is a valid `<epiparameter>` object

## Usage

```
test_epiparameter(x)
```

## Arguments

| | |
|---|---|
| `x` | An R object. |

## Value

A boolean `logical` whether the object is a valid `<epiparameter>` object (prints message when invalid `<epiparameter>` object is provided).

## Examples

```
ep <- epiparameter_db(single_epiparameter = TRUE)
test_epiparameter(ep)

# example with invalid <epiparameter>
ep$disease <- NULL
test_epiparameter(ep)
```

# Index

∗ **datasets**
  epiparameter_options, 31
  epireview_core_cols, 31
∗ **epiparameter_distribution_functions**
  epiparameter_distribution_functions, 30
∗ **extract**
  extract_param, 32
..., *47*
.calc_dist_params(), *25*
.check_optim_conv(), *33*

aggregate.multi_epiparameter, 3
as.data.frame(), *4, 5*
as.data.frame.epiparameter, 4
as.data.frame.multi_epiparameter, 5
as.function(), *5*
as.function.epiparameter, 5
as.person(), *13*
as_epiparameter, 7
as_epiparameter.data.frame, 8
assert_epiparameter, 6

bibentry(), *13*

c(), *8*
c.epiparameter, 8
c.multi_epiparameter *(c.epiparameter)*, 8
calc_disc_dist_quantile, 9
cdf.epiparameter *(epiparameter_distribution_functions)*, 30
convert_params_to_summary_stats, 10
convert_params_to_summary_stats(), *12*
convert_summary_stats_to_params, 11
convert_summary_stats_to_params(), *10*
create_citation, 13
create_citation(), *25*
create_metadata, 14
create_metadata(), *25*
create_method_assess, 15
create_method_assess(), *25*
create_prob_distribution, 17
create_prob_distribution(), *20, 25*
create_region, 19
create_summary_stats, 19
create_summary_stats(), *25*
create_uncertainty, 22
create_uncertainty(), *25*

density.epiparameter *(epiparameter_distribution_functions)*, 30
discretise, 23
discretise(), *4*
discretize *(discretise)*, 23
distcrete::distcrete(), *17, 18, 25*
distributional::dist_gamma(), *17*
distributional::dist_mixture(), *3*
dots, *3–8, 17, 23, 25, 30, 35–38, 42, 45, 48, 49*

epiparameter, 24
epiparameter(), *20*
epiparameter_db, 27
epiparameter_db(), *24, 27*
epiparameter_distribution_functions, *5*, 30
epiparameter_options, 31
epireview_core_cols, 31
extract_param, 32
extract_param(), *9*

family(), *34*
family.epiparameter, 34
format.epiparameter, 35
function, *6*

generate.epiparameter *(epiparameter_distribution_functions)*, 30

get_citation, 36
get_citation.epiparameter, 36
get_citation.multi_epiparameter, 37
get_parameters, 38
get_parameters.epiparameter, 38
get_parameters.multi_epiparameter
(get_parameters.epiparameter),
38
getOption(), *31*

head(), *49*

is_continuous, 39
is_continuous(), *48*
is_epiparameter, 40
is_epiparameter_params, 41
is_parameterised, 42
is_parameterised(), *4*, *28*, *48*
is_parameterized (is_parameterised), 42
is_truncated, 43

lines(), *44*
lines.epiparameter, 44

mean.epiparameter, 45

optim(), *33*
options(), *31*, *49*

parameter_tbl, 45
person(), *13*
plot(), *47*
plot.epiparameter, 46
plot.multi_epiparameter, 47
print.epiparameter, 48
print.multi_epiparameter, 49

quantile.epiparameter
(epiparameter_distribution_functions),
30

subset(), *28*

test_epiparameter, 50