

# Package ‘simulist’

July 23, 2025

**Title** Simulate Disease Outbreak Line List and Contacts Data

**Version** 0.5.0

**Description** Tools to simulate realistic raw case data for an epidemic in the form of line lists and contacts using a branching process. Simulated outbreaks are parameterised with epidemiological parameters and can have age-structured populations, age-stratified hospitalisation and death risk and time-varying case fatality risk.

**License** MIT + file LICENSE

**URL** <https://github.com/epiverse-trace/simulist>,  
<https://epiverse-trace.github.io/simulist/>

**BugReports** <https://github.com/epiverse-trace/simulist/issues>

**Depends** R (>= 4.1.0)

**Imports** checkmate, english, epiparameter (>= 0.4.0), randomNames, rlang, stats

**Suggests** dplyr, epicontracts (>= 1.1.3), ggplot2, incidence2 (>= 2.3.0), knitr, rmarkdown, spelling, testthat (>= 3.0.0), tidyr

**VignetteBuilder** knitr

**Config/Needs/website** epiverse-trace/epiversetheme

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Joshua W. Lambert [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-5218-3046>),  
Carmen Tamayo [aut] (ORCID: <https://orcid.org/0000-0003-4184-2864>),  
Hugo Gruson [ctb, rev] (ORCID: <https://orcid.org/0000-0002-4094-1476>),  
Pratik R. Gupte [ctb, rev] (ORCID: <https://orcid.org/0000-0001-5294-7819>),  
Adam Kucharski [rev] (ORCID: <https://orcid.org/0000-0001-8814-9421>),

Chris Hartgerink [rev] (ORCID: <<https://orcid.org/0000-0003-1050-6809>>),  
Sebastian Funk [ctb] (ORCID: <<https://orcid.org/0000-0002-2842-3406>>),  
London School of Hygiene and Tropical Medicine, LSHTM [cph] (ROR:  
<<https://ror.org/00a0jsq62>>)

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

**Repository** CRAN

**Date/Publication** 2025-03-25 19:50:02 UTC

Contents

create_config . . . . .	2
messy_linelist . . . . .	3
sim_contacts . . . . .	6
sim_linelist . . . . .	8
sim_outbreak . . . . .	14
truncate_linelist . . . . .	19

<b>Index</b>	<b>22</b>
--------------	-----------

---

create_config	Create a list of configuration settings for some details of <a href="#">sim_linelist()</a>
---------------	--

---

Description

Create a list of configuration settings for some details of [sim\\_linelist\(\)](#)

Usage

create\_config(...)

Arguments

... <[dynamic-dots](#)> Named elements to replace default settings. Only if names match exactly are elements replaced, otherwise the function errors.  
Accepted arguments and their defaults are:  
last\_contact\_distribution A function to generate the time for last contact. Default parameterisation is a Poisson distribution with a  $\lambda$  of 3.  
first\_contact\_distribution A function to generate the time for the first contact. Default parameterisation is a Poisson distribution with a  $\lambda$  of 3.  
ct\_distribution A function to generate Ct values for each confirmed case. Default parameterisation is a Normal distribution with a mean ( $\mu$ ) of 25 and a standard deviation ( $\sigma$ ) of 2.  
network A character string, either "adjusted" (default) or "unadjusted".

`time_varying_death_risk` By default is NULL, but can also accept a function with two arguments, risk and time, to apply a time varying death risk of hospitalised and non-hospitalised cases in the outbreak simulation. See `vignette("time-varying-cfr", package = "simulist")`.

## Details

The `config` argument in `sim_linelist()` controls the small details around time windows around infections (time of first contact and last contact with infector), and the distribution of the Cycle threshold (Ct) value from a Real-time PCR or quantitative PCR (qPCR) for confirmed cases, the network effect in the simulation, and if there is a time-varying death risk.

These parameters do not warrant their own arguments in `sim_linelist()` as they rarely need to be changed from their default setting. Therefore it is not worth increasing the number of `sim_linelist()` arguments to accommodate these and the `config` argument keeps the function signature simpler and more readable.

The `last_contact_distribution` and `first_contact_distribution` can accept any function that generates positive integers (e.g. discrete probability distribution, `rpois()` or `rgeom()`). The `ct_distribution` can accept any function that generates real numbers (e.g. continuous or discrete probability distribution, `rnorm()`, `rlnorm()`).

The `network` option controls whether to sample contacts from a adjusted or unadjusted contact distribution. Adjusted (default) sampling uses  $q(n) \sim (n+1)p(n+1)$  where  $p(n)$  is the probability density function of a distribution, e.g., Poisson or Negative binomial. Unadjusted (`network = "unadjusted"`) instead samples contacts directly from a probability distribution  $p(n)$ .

## Value

A list of settings for `sim_linelist()`

## Examples

```
# example with default configuration
create_config()

# example with customised Ct distribution
create_config(
  ct_distribution = function(x) rlnorm(n = x, meanlog = 2, sdlog = 1)
)
```

---

`messy_linelist`

*Create messy line list data*

---

## Description

Take line list output from `sim_linelist()` and replace elements of the `<data.frame>` with missing values (e.g. NA), introduce spelling mistakes and inconsistencies, as well as coerce date types.

**Usage**

```
messy_linelist(linelist, ...)
```

**Arguments**

**linelist** Line list <data.frame> output from [sim\\_linelist\(\)](#).

**...** [<dynamic-dots>](#) Named elements to replace default settings. Only if names match exactly are elements replaced, otherwise the function errors.

Accepted arguments and their defaults are:

**prop\_missing** A numeric between 0 and 1 for the proportion of missing values introduced. Default is 0.1 (10%).

**missing\_value** A single atomic R object used to represent missing values. Default is NA.

**prop\_spelling\_mistakes** A numeric between 0 and 1 used to specify the proportion of spelling mistakes in character columns. Default is 0.1 (10%).

**inconsistent\_sex** A logical boolean to specify whether the \$sex column uses "m" and "f", or inconsistently uses "m", "f", "M", "F", "male", "female", "Male" or "Female". Default is TRUE so sexes are sampled from the options.

**sex\_as\_numeric** A logical boolean used to specify whether the values in the \$sex column should be encoded as numeric values (0 and 1). Default is FALSE. sex\_as\_numeric cannot be TRUE if inconsistent\_sex = TRUE.

**numeric\_as\_char** A logical boolean used to specify whether numeric columns should be coerced to character. Default is TRUE.

**date\_as\_char** A logical boolean used to specify whether Date columns should be coerced to character. Default is TRUE.

**inconsistent\_dates** A logical boolean used to specify whether the values in Date columns are inconsistently formatted (e.g. "%Y-%m-%d", "%Y/%m/%d", "%d-%m-%Y", or "%d %B %Y"). Default is FALSE.

**prop\_int\_as\_word** A numeric between 0 and 1 for the proportion of elements in integer columns should that are coerced to words (see [english::words\(\)](#)). Default is 0.5 (50%).

**prop\_duplicate\_row** A numeric between 0 and 1 for the proportion of rows to duplicate. Default is 0.01 (1%). If prop\_duplicate\_row > 0 then it is guaranteed that at least one row will be duplicated.

**inconsistent\_id** A logical boolean used to specify whether the \$id column has inconsistent formatting by appending random prefixes and suffixes to a random sample (~10%) of IDs. Default is FALSE, so IDs are numbers (numeric, characters or words depending on prop\_int\_as\_word and numeric\_as\_char).

**Details**

By default messy\_linelist():

- Introduces 10% of values missing, i.e. converts to NA.

- Introduces spelling mistakes in 10% of character columns.
- Introduce inconsistency in the reporting of \$sex.
- Converts numeric columns (double & integer) to character.
- Converts Date columns to character.
- Converts 50% of integers to (English) words.
- Duplicates 1% of rows.

Setting `missing_value` to something other than NA will likely cause type coercion in the line list `<data.frame>` columns, most likely to character.

When setting `sex_as_numeric` to TRUE, male is set to 0 and female to 1. Only one of `inconsistent_sex` or `sex_as_numeric` can be TRUE, otherwise the function will error.

If `numeric_as_char` = TRUE and `sex_as_numeric` = TRUE then the sex encoded as 0 or 1 is converted to character. If `prop_spelling_mistake` > 0 and `numeric_as_char` = TRUE the columns that are converted from numeric to character do not have spelling mistakes introduced, because they are numeric characters stored as character strings. If `prop_spelling_mistake` > 0 and `date_as_char` = TRUE spelling mistakes are not introduced into dates.

The Date columns can be converted into an inconsistent format by setting `inconsistent_dates` = TRUE and it requires `date_as_char` = TRUE, if the latter is FALSE the function will error.

If `numeric_as_char` = FALSE and `prop_int_as_word` > 0 then the integer columns are converted to character string (either character numbers or words) but the other numeric columns are not coerced. Spelling mistakes are not introduced into integers converted to words when `prop_spelling_mistakes` > 0 and `prop_int_as_word` > 0.

Rows are duplicated after other *messy* modifications so the duplicated row contains identical *messy* elements.

## Value

A messy line list `<data.frame>`.

## Examples

```
linelist <- sim_linelist()
messy_linelist <- messy_linelist(linelist)

# increasing proportion of missingness to 30% with a missing value of -99
messy_linelist <- messy_linelist(
  linelist,
  prop_missing = 0.3,
  missing_value = -99
)

# increasing proportion of spelling mistakes to 50%
messy_linelist <- messy_linelist(linelist, prop_spelling_mistakes = 0.5)

# encode `$sex` as `numeric`
messy_linelist <- messy_linelist(
  linelist,
  sex_as_numeric = TRUE,
```

```

    inconsistent_sex = FALSE
  )

  # inconsistently formatted dates
  messy_linelist <- messy_linelist(linelist, inconsistent_dates = TRUE)

```

---

sim\_contacts

*Simulate contacts for an infectious disease outbreak*


---

## Description

Simulate contacts for an infectious disease outbreak

## Usage

```

sim_contacts(
  contact_distribution = function(x) stats::dpois(x = x, lambda = 2),
  infectious_period = function(x) stats::rlnorm(n = x, meanlog = 2, sdlog = 0.5),
  prob_infection = 0.5,
  outbreak_start_date = as.Date("2023-01-01"),
  anonymise = FALSE,
  outbreak_size = c(10, 10000),
  population_age = c(1, 90),
  contact_tracing_status_probs = c(under_followup = 0.7, lost_to_followup = 0.2, unknown
    = 0.1),
  config = create_config()
)

```

## Arguments

contact\_distribution

A function or an <epiparameter> object to generate the number of contacts per infection.

The function can be defined or anonymous. The function must have a single argument in the form of an integer vector with elements representing the number of contacts, and return a numeric vector where each element corresponds to the probability of observing the number of contacts in the vector passed to the function. The index of the numeric vector returned is offset by one to the corresponding probability of observing the number of contacts, i.e. the first element of the output vector is the probability of observing zero contacts, the second element is the probability of observing one contact, etc.

An <epiparameter> can be provided. This will be converted into a probability mass function internally.

The default is an anonymous function with a Poisson probability mass function ([dpois\(\)](#)) with a mean ( $\lambda$ ) of 2 contacts per infection.

infectious_period	<p>A function or an <code>&lt;epiparameter&gt;</code> object for the infectious period. This defines the duration from becoming infectious to no longer infectious. In the simulation, individuals are assumed to become infectious immediately after being infected (the latency period is assumed to be zero). The time intervals between an infected individual and their contacts are assumed to be uniformly distributed within the infectious period. Infectious periods must be strictly positive.</p> <p>The function can be defined or anonymous. The function must return a vector of randomly generated real numbers representing sampled infectious periods. The function must have a single argument, the number of random infectious periods to generate.</p> <p>An <code>&lt;epiparameter&gt;</code> can be provided. This will be converted into random number generator internally.</p> <p>The default is an anonymous function with a lognormal distribution random number generator (<code>rlnorm()</code>) with <code>meanlog = 2</code> and <code>sdlog = 0.5</code>.</p>
prob_infection	A single numeric for the probability of a secondary contact being infected by an infected primary contact.
outbreak_start_date	A date for the start of the outbreak.
anonymise	A logical boolean for whether case names should be anonymised. Default is <code>FALSE</code> .
outbreak_size	A numeric vector of length 2 defining the minimum and the maximum number of infected individuals for the simulated outbreak. Default is <code>c(10, 1e4)</code> , so the minimum outbreak size is 10 infected individuals, and the maximum outbreak size is 10,000 infected individuals. Either number can be changed to increase or decrease the maximum or minimum outbreak size to allow simulating larger or smaller outbreaks. If the minimum outbreak size cannot be reached after running the simulation for many iterations (internally) then the function errors, whereas if the maximum outbreak size is exceeded the function returns the data early and a warning stating how many cases and contacts are returned.
population_age	Either a numeric vector with two elements or a <code>&lt;data.frame&gt;</code> with age structure in the population. Use a numeric vector to specify the age range of the population, the first element is the lower bound for the age range, and the second is the upper bound for the age range (both inclusive, i.e. <code>[lower, upper]</code> ). The <code>&lt;data.frame&gt;</code> with age groups and the proportion of the population in that group. See details and examples for more information.
contact_tracing_status_probs	A named numeric vector with the probability of each contact tracing status. The names of the vector must be <code>"under_followup"</code> , <code>"lost_to_followup"</code> , <code>"unknown"</code> . Values of each contact tracing status must sum to one.
config	A list of settings to adjust the randomly sampled delays and <code>Ct</code> values. See <code>create_config()</code> for more information.

## Value

A contacts `<data.frame>`

**Author(s)**

Joshua W. Lambert, Carmen Tamayo

**Examples**

```
# quickly simulate contact tracing data using the function defaults
contacts <- sim_contacts()
head(contacts)

# to simulate more realistic contact tracing data load epiparameters from
# {epiparameter}
library(epiparameter)
contact_distribution <- epiparameter(
  disease = "COVID-19",
  epi_name = "contact distribution",
  prob_distribution = create_prob_distribution(
    prob_distribution = "pois",
    prob_distribution_params = c(mean = 2)
  )
)

infectious_period <- epiparameter(
  disease = "COVID-19",
  epi_name = "infectious period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)

contacts <- sim_contacts(
  contact_distribution = contact_distribution,
  infectious_period = infectious_period,
  prob_infection = 0.5
)
```

---

sim\_linelist

*Simulate a line list*


---

**Description**

The line list is simulated using a branching process and parameterised with epidemiological parameters.

**Usage**

```
sim_linelist(
  contact_distribution = function(x) stats::dpois(x = x, lambda = 2),
  infectious_period = function(x) stats::rlnorm(n = x, meanlog = 2, sdlog = 0.5),
```



```

prob_infection = 0.5,
onset_to_hosp = function(x) stats::rlnorm(n = x, meanlog = 1.5, sdlog = 0.5),
onset_to_death = function(x) stats::rlnorm(n = x, meanlog = 2.5, sdlog = 0.5),
onset_to_recovery = NULL,
reporting_delay = NULL,
hosp_risk = 0.2,
hosp_death_risk = 0.5,
non_hosp_death_risk = 0.05,
outbreak_start_date = as.Date("2023-01-01"),
anonymise = FALSE,
outbreak_size = c(10, 10000),
population_age = c(1, 90),
case_type_probs = c(suspected = 0.2, probable = 0.3, confirmed = 0.5),
config = create_config()
)

```

## Arguments

### contact\_distribution

A function or an `<epiparameter>` object to generate the number of contacts per infection.

The function can be defined or anonymous. The function must have a single argument in the form of an integer vector with elements representing the number of contacts, and return a numeric vector where each element corresponds to the probability of observing the number of contacts in the vector passed to the function. The index of the numeric vector returned is offset by one to the corresponding probability of observing the number of contacts, i.e. the first element of the output vector is the probability of observing zero contacts, the second element is the probability of observing one contact, etc.

An `<epiparameter>` can be provided. This will be converted into a probability mass function internally.

The default is an anonymous function with a Poisson probability mass function (`dpois()`) with a mean ( $\lambda$ ) of 2 contacts per infection.

### infectious\_period

A function or an `<epiparameter>` object for the infectious period. This defines the duration from becoming infectious to no longer infectious. In the simulation, individuals are assumed to become infectious immediately after being infected (the latency period is assumed to be zero). The time intervals between an infected individual and their contacts are assumed to be uniformly distributed within the infectious period. Infectious periods must be strictly positive.

The function can be defined or anonymous. The function must return a vector of randomly generated real numbers representing sampled infectious periods. The function must have a single argument, the number of random infectious periods to generate.

An `<epiparameter>` can be provided. This will be converted into random number generator internally.

The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with  $\text{meanlog} = 2$  and  $\text{sdlog} = 0.5$ .

- prob\_infection** A single numeric for the probability of a secondary contact being infected by an infected primary contact.
- onset\_to\_hosp** A function or an <epiparameter> object for the onset-to-hospitalisation delay distribution. `onset_to_hosp` can also be set to NULL to not simulate hospitalisation (admission) dates.
- The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-hospitalisation delay. The function must have a single argument.
- An <epiparameter> can be provided. This will be converted into a random number generator internally.
- The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with `meanlog = 1.5` and `sdlog = 0.5`.
- If `onset_to_hosp` is set to NULL then `hosp_risk` and `hosp_death_risk` will be automatically set to NULL if not manually specified.
- onset\_to\_death** A function or an <epiparameter> object for the onset-to-death delay distribution. `onset_to_death` can also be set to NULL to not simulate dates for individuals that died.
- The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-death delay. The function must have a single argument.
- An <epiparameter> can be provided. This will be converted into a random number generator internally.
- The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with `meanlog = 2.5` and `sdlog = 0.5`.
- If `onset_to_death` is set to NULL then `non_hosp_death_risk` and `hosp_death_risk` will be automatically set to NULL if not manually specified.
- For hospitalised cases, the function ensures the onset-to-death time is greater than the onset-to-hospitalisation time. After many (1000) attempts, if an onset-to-death time (from `onset_to_death`) cannot be sampled that is greater than an onset-to-hospitalisation time (from `onset_to_hosp`) then the function will error. Due to this conditional sampling, the onset-to-death times in the line list may not resemble the distributional form input into the function.
- onset\_to\_recovery** A function or an <epiparameter> object for the onset-to-recovery delay distribution. `onset_to_recovery` can also be NULL to not simulate dates for individuals that recovered.
- The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-recovery delay. The function must have a single argument.
- An <epiparameter> can be provided. This will be converted into a random number generator internally.
- The default is NULL so by default cases that recover get an NA in the `$date_outcome` line list column.
- For hospitalised cases, the function ensures the onset-to-recovery time is greater than the onset-to-hospitalisation time. After many (1000) attempts, if an onset-to-recovery time (from `onset_to_recovery`) cannot be sampled that is greater

than a onset-to-hospitalisation time (from `onset_to_hosp`) then the function will error. Due to this conditional sampling, the onset-to-recovery times in the line list may not resemble the distributional form input into the function.

#### reporting\_delay

A function for the reporting delay distribution or NULL. The (random) number generating function creates delays between the time of symptom onset (`$date_onset`) and the case being reported (`$date_reporting`).

The function can be defined or anonymous. The function must return a vector of numerics for the length of the reporting delay. The function must have a single argument.

The default is NULL so by default there is no reporting delay, and the `$date_reporting` line list column is identical to the `$date_onset` column.

#### hosp\_risk

Either a single numeric for the hospitalisation risk of everyone in the population, or a `<data.frame>` with age specific hospitalisation risks. Default is 20% hospitalisation (0.2) for the entire population. If the `onset_to_hosp` argument is set to NULL this argument will automatically be set to NULL if not specified or can be manually set to NULL. See details and examples for more information.

#### hosp\_death\_risk

Either a single numeric for the death risk for hospitalised individuals across the population, or a `<data.frame>` with age specific hospitalised death risks. Default is 50% death risk in hospitals (0.5) for the entire population. If the `onset_to_death` argument is set to NULL this argument will automatically be set to NULL if not specified or can be manually set to NULL. See details and examples for more information. The `hosp_death_risk` can vary through time if specified in the `time_varying_death_risk` element of config, see `vignette("time-varying-cfr", package = "simulist")` for more information.

#### non\_hosp\_death\_risk

Either a single numeric for the death risk for outside of hospitals across the population, or a `<data.frame>` with age specific death risks outside of hospitals. Default is 5% death risk outside of hospitals (0.05) for the entire population. If the `onset_to_death` argument is set to NULL this argument will automatically be set to NULL if not specified or can be manually set to NULL. See details and examples for more information. The `non_hosp_death_risk` can vary through time if specified in the `time_varying_death_risk` element of config, see `vignette("time-varying-cfr", package = "simulist")` for more information.

#### outbreak\_start\_date

A date for the start of the outbreak.

#### anonymise

A logical boolean for whether case names should be anonymised. Default is FALSE.

#### outbreak\_size

A numeric vector of length 2 defining the minimum and the maximum number of infected individuals for the simulated outbreak. Default is `c(10, 1e4)`, so the minimum outbreak size is 10 infected individuals, and the maximum outbreak size is 10,000 infected individuals. Either number can be changed to increase or decrease the maximum or minimum outbreak size to allow simulating larger or smaller outbreaks. If the minimum outbreak size cannot be reached after running the simulation for many iterations (internally) then the function errors,

- whereas if the maximum outbreak size is exceeded the function returns the data early and a warning stating how many cases and contacts are returned.
- population\_age** Either a numeric vector with two elements or a `<data.frame>` with age structure in the population. Use a numeric vector to specify the age range of the population, the first element is the lower bound for the age range, and the second is the upper bound for the age range (both inclusive, i.e. `[lower, upper]`). The `<data.frame>` with age groups and the proportion of the population in that group. See details and examples for more information.
- case\_type\_probs** A named numeric vector with the probability of each case type. The names of the vector must be "suspected", "probable", "confirmed". Values of each case type must sum to one.
- config** A list of settings to adjust the randomly sampled delays and Ct values. See [create\\_config\(\)](#) for more information.

## Details

For age-stratified hospitalised and death risks a `<data.frame>` will need to be passed to the `hosp_risk` and/or `hosp_death_risk` arguments. This `<data.frame>` should have two columns:

- **age\_limit**: a column with one numeric per cell for the lower bound (minimum) age of the age group (inclusive).
- **risk**: a column with one numeric per cell for the proportion (or probability) of hospitalisation for that age group. Should be between 0 and 1.

For an age-structured population, a `<data.frame>` with two columns:

- **age\_limit**: a column with one numeric per cell for the lower bound (minimum) age of the age group (inclusive), except the last element which is the upper bound (maximum) of the population.
- **proportion**: a column with the proportion of the population that are in that age group. Proportions must sum to one.

## Value

A line list `<data.frame>`

## Author(s)

Joshua W. Lambert, Carmen Tamayo

## Examples

```
# quickly simulate a line list using the function defaults
linelist <- sim_linelist()
head(linelist)

# to simulate a more realistic line list load epiparameters from
# {epiparameter}
library(epiparameter)
```

```

contact_distribution <- epiparameter(
  disease = "COVID-19",
  epi_name = "contact distribution",
  prob_distribution = create_prob_distribution(
    prob_distribution = "pois",
    prob_distribution_params = c(mean = 2)
  )
)

infectious_period <- epiparameter(
  disease = "COVID-19",
  epi_name = "infectious period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)

onset_to_hosp <- epiparameter(
  disease = "COVID-19",
  epi_name = "onset to hospitalisation",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 0.5)
  )
)

# get onset to death from {epiparameter} database
onset_to_death <- epiparameter_db(
  disease = "COVID-19",
  epi_name = "onset to death",
  single_epiparameter = TRUE
)

# example with single hospitalisation risk for entire population
linelist <- sim_linelist(
  contact_distribution = contact_distribution,
  infectious_period = infectious_period,
  prob_infection = 0.5,
  onset_to_hosp = onset_to_hosp,
  onset_to_death = onset_to_death,
  hosp_risk = 0.5
)
head(linelist)

# example with age-stratified hospitalisation risk
# 20% for over 80s
# 10% for under 5s
# 5% for the rest
age_dep_hosp_risk <- data.frame(
  age_limit = c(1, 5, 80),
  risk = c(0.1, 0.05, 0.2)
)
linelist <- sim_linelist(

```

```

    contact_distribution = contact_distribution,
    infectious_period = infectious_period,
    prob_infection = 0.5,
    onset_to_hosp = onset_to_hosp,
    onset_to_death = onset_to_death,
    hosp_risk = age_dep_hosp_risk
  )
  head(linelist)

```

---

sim\_outbreak

---

*Simulate a line list and a contacts table*


---

## Description

The line list and contacts are simulated using a branching process and parameterised with epidemiological parameters.

## Usage

```

sim_outbreak(
  contact_distribution = function(x) stats::dpois(x = x, lambda = 2),
  infectious_period = function(x) stats::rlnorm(n = x, meanlog = 2, sdlog = 0.5),
  prob_infection = 0.5,
  onset_to_hosp = function(x) stats::rlnorm(n = x, meanlog = 1.5, sdlog = 0.5),
  onset_to_death = function(x) stats::rlnorm(n = x, meanlog = 2.5, sdlog = 0.5),
  onset_to_recovery = NULL,
  reporting_delay = NULL,
  hosp_risk = 0.2,
  hosp_death_risk = 0.5,
  non_hosp_death_risk = 0.05,
  outbreak_start_date = as.Date("2023-01-01"),
  anonymise = FALSE,
  outbreak_size = c(10, 10000),
  population_age = c(1, 90),
  case_type_probs = c(suspected = 0.2, probable = 0.3, confirmed = 0.5),
  contact_tracing_status_probs = c(under_followup = 0.7, lost_to_followup = 0.2, unknown
    = 0.1),
  config = create_config()
)

```

## Arguments

contact\_distribution

A function or an <epiparameter> object to generate the number of contacts per infection.

The function can be defined or anonymous. The function must have a single argument in the form of an integer vector with elements representing the number of contacts, and return a numeric vector where each element corresponds to

the probability of observing the number of contacts in the vector passed to the function. The index of the numeric vector returned is offset by one to the corresponding probability of observing the number of contacts, i.e. the first element of the output vector is the probability of observing zero contacts, the second element is the probability of observing one contact, etc.

An `<epiparameter>` can be provided. This will be converted into a probability mass function internally.

The default is an anonymous function with a Poisson probability mass function (`dpois()`) with a mean ( $\lambda$ ) of 2 contacts per infection.

`infectious_period`

A function or an `<epiparameter>` object for the infectious period. This defines the duration from becoming infectious to no longer infectious. In the simulation, individuals are assumed to become infectious immediately after being infected (the latency period is assumed to be zero). The time intervals between an infected individual and their contacts are assumed to be uniformly distributed within the infectious period. Infectious periods must be strictly positive.

The function can be defined or anonymous. The function must return a vector of randomly generated real numbers representing sampled infectious periods. The function must have a single argument, the number of random infectious periods to generate.

An `<epiparameter>` can be provided. This will be converted into random number generator internally.

The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with `meanlog = 2` and `sdlog = 0.5`.

`prob_infection` A single numeric for the probability of a secondary contact being infected by an infected primary contact.

`onset_to_hosp` A function or an `<epiparameter>` object for the onset-to-hospitalisation delay distribution. `onset_to_hosp` can also be set to `NULL` to not simulate hospitalisation (admission) dates.

The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-hospitalisation delay. The function must have a single argument.

An `<epiparameter>` can be provided. This will be converted into a random number generator internally.

The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with `meanlog = 1.5` and `sdlog = 0.5`.

If `onset_to_hosp` is set to `NULL` then `hosp_risk` and `hosp_death_risk` will be automatically set to `NULL` if not manually specified.

`onset_to_death` A function or an `<epiparameter>` object for the onset-to-death delay distribution. `onset_to_death` can also be set to `NULL` to not simulate dates for individuals that died.

The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-death delay. The function must have a single argument.

An `<epiparameter>` can be provided. This will be converted into a random number generator internally.

The default is an anonymous function with a lognormal distribution random number generator (`rlnorm()`) with `meanlog = 2.5` and `sdlog = 0.5`.

If `onset_to_death` is set to `NULL` then `non_hosp_death_risk` and `hosp_death_risk` will be automatically set to `NULL` if not manually specified.

For hospitalised cases, the function ensures the onset-to-death time is greater than the onset-to-hospitalisation time. After many (1000) attempts, if an onset-to-death time (from `onset_to_death`) cannot be sampled that is greater than an onset-to-hospitalisation time (from `onset_to_hosp`) then the function will error. Due to this conditional sampling, the onset-to-death times in the line list may not resemble the distributional form input into the function.

#### `onset_to_recovery`

A function or an `<epiparameter>` object for the onset-to-recovery delay distribution. `onset_to_recovery` can also be `NULL` to not simulate dates for individuals that recovered.

The function can be defined or anonymous. The function must return a vector of numerics for the length of the onset-to-recovery delay. The function must have a single argument.

An `<epiparameter>` can be provided. This will be converted into a random number generator internally.

The default is `NULL` so by default cases that recover get an NA in the `$date_outcome` line list column.

For hospitalised cases, the function ensures the onset-to-recovery time is greater than the onset-to-hospitalisation time. After many (1000) attempts, if an onset-to-recovery time (from `onset_to_recovery`) cannot be sampled that is greater than an onset-to-hospitalisation time (from `onset_to_hosp`) then the function will error. Due to this conditional sampling, the onset-to-recovery times in the line list may not resemble the distributional form input into the function.

#### `reporting_delay`

A function for the reporting delay distribution or `NULL`. The (random) number generating function creates delays between the time of symptom onset (`$date_onset`) and the case being reported (`$date_reporting`).

The function can be defined or anonymous. The function must return a vector of numerics for the length of the reporting delay. The function must have a single argument.

The default is `NULL` so by default there is no reporting delay, and the `$date_reporting` line list column is identical to the `$date_onset` column.

#### `hosp_risk`

Either a single numeric for the hospitalisation risk of everyone in the population, or a `<data.frame>` with age specific hospitalisation risks. Default is 20% hospitalisation (0.2) for the entire population. If the `onset_to_hosp` argument is set to `NULL` this argument will automatically be set to `NULL` if not specified or can be manually set to `NULL`. See details and examples for more information.

#### `hosp_death_risk`

Either a single numeric for the death risk for hospitalised individuals across the population, or a `<data.frame>` with age specific hospitalised death risks. Default is 50% death risk in hospitals (0.5) for the entire population. If the `onset_to_death` argument is set to `NULL` this argument will automatically be set to `NULL` if not specified or can be manually set to `NULL`. See details and examples



for more information. The `hosp_death_risk` can vary through time if specified in the `time_varying_death_risk` element of config, see `vignette("time-varying-cfr", package = "simulist")` for more information.

`non_hosp_death_risk`

Either a single numeric for the death risk for outside of hospitals across the population, or a `<data.frame>` with age specific death risks outside of hospitals. Default is 5% death risk outside of hospitals (0.05) for the entire population. If the `onset_to_death` argument is set to `NULL` this argument will automatically be set to `NULL` if not specified or can be manually set to `NULL`. See details and examples for more information. The `non_hosp_death_risk` can vary through time if specified in the `time_varying_death_risk` element of config, see `vignette("time-varying-cfr", package = "simulist")` for more information.

`outbreak_start_date`

A date for the start of the outbreak.

`anonymise`

A logical boolean for whether case names should be anonymised. Default is `FALSE`.

`outbreak_size`

A numeric vector of length 2 defining the minimum and the maximum number of infected individuals for the simulated outbreak. Default is `c(10, 1e4)`, so the minimum outbreak size is 10 infected individuals, and the maximum outbreak size is 10,000 infected individuals. Either number can be changed to increase or decrease the maximum or minimum outbreak size to allow simulating larger or smaller outbreaks. If the minimum outbreak size cannot be reached after running the simulation for many iterations (internally) then the function errors, whereas if the maximum outbreak size is exceeded the function returns the data early and a warning stating how many cases and contacts are returned.

`population_age`

Either a numeric vector with two elements or a `<data.frame>` with age structure in the population. Use a numeric vector to specify the age range of the population, the first element is the lower bound for the age range, and the second is the upper bound for the age range (both inclusive, i.e. [lower, upper]). The `<data.frame>` with age groups and the proportion of the population in that group. See details and examples for more information.

`case_type_probs`

A named numeric vector with the probability of each case type. The names of the vector must be "suspected", "probable", "confirmed". Values of each case type must sum to one.

`contact_tracing_status_probs`

A named numeric vector with the probability of each contact tracing status. The names of the vector must be "under\_followup", "lost\_to\_followup", "unknown". Values of each contact tracing status must sum to one.

`config`

A list of settings to adjust the randomly sampled delays and Ct values. See `create_config()` for more information.

## Details

For age-stratified hospitalised and death risks a `<data.frame>` will need to be passed to the `hosp_risk` and/or `hosp_death_risk` arguments. This `<data.frame>` should have two columns:

- `age_limit`: a column with one numeric per cell for the lower bound (minimum) age of the age group (inclusive).
- `risk`: a column with one numeric per cell for the proportion (or probability) of hospitalisation for that age group. Should be between 0 and 1.

For an age-structured population, a `<data.frame>` with two columns:

- `age_limit`: a column with one numeric per cell for the lower bound (minimum) age of the age group (inclusive), except the last element which is the upper bound (maximum) of the population.
- `proportion`: a column with the proportion of the population that are in that age group. Proportions must sum to one.

### Value

A list with two elements:

1. A line list `<data.frame>`
2. A contacts `<data.frame>`

### Author(s)

Joshua W. Lambert

### Examples

```
# quickly simulate an outbreak using the function defaults
outbreak <- sim_outbreak()
head(outbreak$linelist)
head(outbreak$contacts)

# to simulate a more realistic outbreak load epiparameters from
# {epiparameter}
library(epiparameter)
contact_distribution <- epiparameter(
  disease = "COVID-19",
  epi_name = "contact distribution",
  prob_distribution = create_prob_distribution(
    prob_distribution = "pois",
    prob_distribution_params = c(mean = 2)
  )
)

infectious_period <- epiparameter(
  disease = "COVID-19",
  epi_name = "infectious period",
  prob_distribution = create_prob_distribution(
    prob_distribution = "gamma",
    prob_distribution_params = c(shape = 1, scale = 1)
  )
)
```

```

onset_to_hosp <- epiparameter(
  disease = "COVID-19",
  epi_name = "onset to hospitalisation",
  prob_distribution = create_prob_distribution(
    prob_distribution = "lnorm",
    prob_distribution_params = c(meanlog = 1, sdlog = 0.5)
  )
)

# get onset to death from {epiparameter} database
onset_to_death <- epiparameter_db(
  disease = "COVID-19",
  epi_name = "onset to death",
  single_epiparameter = TRUE
)

outbreak <- sim_outbreak(
  contact_distribution = contact_distribution,
  infectious_period = infectious_period,
  prob_infection = 0.5,
  onset_to_hosp = onset_to_hosp,
  onset_to_death = onset_to_death
)

```

---

truncate_linelist	<i>Adjust or subset a line list to account for right truncation</i>
-------------------	---

---

## Description

Adjust or subset the line list <data.frame> by removing cases that have not been reported by the truncation time and setting hospitalisation admission or outcome dates that are after the truncation point to NA.

This is to replicate real-time outbreak data where recent cases or outcomes are not yet observed or reported (right truncation). It implies an assumption that symptom onsets are reported with a delay but hospitalisations are reported instantly.

## Usage

```

truncate_linelist(
  linelist,
  truncation_day = 14,
  unit = c("days", "weeks", "months", "years"),
  direction = c("backwards", "forwards")
)

```

**Arguments**

linelist	Line list <data.frame> output from <code>sim_linelist()</code> .
truncation_day	A single numeric specifying the number of days (default), weeks, months or years before the end of the outbreak (default) or since the start of the outbreak (see direction argument) to truncate the line list at. By default it is 14 days before the end of the outbreak.  Alternatively, truncation_day can accept a <Date> and this is used as the truncation_day and the unit and direction is ignored.
unit	A character string, either "days" (default), "weeks", "months", or "years", specifying the units of the truncation_day argument.  Years are assumed to be 365.25 days and months are assumed to be 365.25 / 12 days (same as <b>lubridate</b> ).
direction	A character string, either "backwards" (default) or "forwards". direction = backwards defines the truncation_day as the time before the end of the outbreak. direction = forwards defines the truncation_day as the time since the start of the outbreak.

**Details**

The day on which the line list is truncated is the same for all individuals in the line list, and is specified by the truncation\_day and unit arguments.

**Value**

A line list <data.frame>.

**Examples**

```
set.seed(1)
linelist <- sim_linelist()
linelist_trunc <- truncate_linelist(linelist)

# set truncation point 3 weeks before the end of outbreak
linelist_trunc <- truncate_linelist(
  linelist,
  truncation_day = 3,
  unit = "weeks"
)

# set truncation point to 2 months since the start of outbreak
linelist_trunc <- truncate_linelist(
  linelist,
  truncation_day = 2,
  unit = "months",
  direction = "forwards"
)

# set truncation point to 2023-03-01
linelist_trunc <- truncate_linelist(
```

```
    linelist,  
    truncation_day = as.Date("2023-03-01")  
)
```

# Index

`create_config`, [2](#)  
`create_config()`, [7](#), [12](#), [17](#)  
  
`dpois()`, [6](#), [9](#), [15](#)  
  
`english::words()`, [4](#)  
  
`messy_linelist`, [3](#)  
  
`rgeom()`, [3](#)  
`rlnorm()`, [3](#), [7](#), [9](#), [10](#), [15](#), [16](#)  
`rnorm()`, [3](#)  
`rpois()`, [3](#)  
  
`sim_contacts`, [6](#)  
`sim_linelist`, [8](#)  
`sim_linelist()`, [2–4](#), [20](#)  
`sim_outbreak`, [14](#)  
  
`truncate_linelist`, [19](#)