Package ‘epichains’

October 14, 2024

Title Simulating and Analysing Transmission Chain Statistics Using Branching Process Models

Version 0.1.1

Description Provides methods to simulate and analyse the size and length of branching processes with an arbitrary offspring distribution. These can be used, for example, to analyse the distribution of chain sizes or length of infectious disease outbreaks, as discussed in Farrington et al. (2003) <doi:10.1093/biostatistics/4.2.279>.

License MIT + file LICENSE

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

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

Depends R (>= 3.6.0)

Imports checkmate, stats, utils

Suggests bookdown, dplyr, epicontacts, ggplot2, knitr, lubridate, rmarkdown, spelling, testthat, truncdist

VignetteBuilder knitr

Config/Needs/website epiverse-trace/epiversetheme

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.2.9000

NeedsCompilation no

Author James M. Azam [aut, cre, cph] (<`https://orcid.org/0000-0001-5782-7330`>),
Sebastian Funk [aut, cph] (<`https://orcid.org/0000-0002-2842-3406`>),
Flavio Finger [aut] (<`https://orcid.org/0000-0002-8613-5170`>),
Zhian N. Kamvar [ctb] (<`https://orcid.org/0000-0003-1458-7108`>),
Hugo Gruson [ctb, rev] (<`https://orcid.org/0000-0002-4094-1476`>),
Karim Mané [rev] (<`https://orcid.org/0000-0002-9892-2999`>),

Pratik Gupte [rev] (<https://orcid.org/0000-0001-5294-7819>),
Joshua W. Lambert [rev] (<https://orcid.org/0000-0001-5218-3046>)

Maintainer James M. Azam <james.azam@lshtm.ac.uk>

Repository CRAN

Date/Publication 2024-10-14 09:20:11 UTC

Contents

`aggregate.epichains` *Aggregate cases in* <epichains> *objects by "generation" or "time", if present*

Description

This function provides a quick way to create a time series of cases over generation or time (if `generation_time` was specified) from simulated <epichains> objects.

Usage

```
## S3 method for class 'epichains'
aggregate(x, by = c("time", "generation"), ...)
```

Arguments

`x`	An <epichains> object.
`by`	The variable to aggregate by; A character string with options "time" and "generation".
`...`	Not used.

Value

A <`data.frame`> object of cases by `by`.

Author(s)

James M. Azam

Examples

```
set.seed(32)
chains <- simulate_chains(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  lambda = 2
)
chains

# Aggregate cases per time
cases_per_time <- aggregate(chains, by = "time")
head(cases_per_time)

# Aggregate cases per generation
cases_per_gen <- aggregate(chains, by = "generation")
head(cases_per_gen)
```

`covid19_sa` *COVID-19 Data Repository for South Africa*

Description

An aggregated subset of the COVID-19 Data Repository for South Africa created, maintained and hosted by Data Science for Social Impact research group, led by Dr. Vukosi Marivate.

Usage

```
covid19_sa
```

Format

`covid19_sa`**:**

A data frame with 19 rows and 2 columns:

date Date case was reported

cases Number of cases

Details

The data is originally provided as a linelist but has been subsetted and cleaned in `data-raw/covid19_sa.R`.

Source

`https://github.com/dsfsi/covid19za` Further details in `data-raw/covid19_sa.R`.

`dborel` *Density of the Borel distribution*

Description

Density of the Borel distribution

Usage

```
dborel(x, mu, log = FALSE)
```

Arguments

`x`	A numeric vector of quantiles.
`mu`	A non-negative number for the poisson mean.
`log`	Logical; if TRUE, probabilities p are given as log(p).

Value

A numeric vector of the borel probability density.

Author(s)

Sebastian Funk James M. Azam

Examples

```
set.seed(32)
dborel(1:5, 1)
```

`head.epichains` — `head` *and* `tail` *method for* `<epichains>` *class*

Description

`head` and `tail` method for `<epichains>` class

Usage

```
## S3 method for class 'epichains'
head(x, ...)

## S3 method for class 'epichains'
tail(x, ...)
```

Arguments

`x`	An `<epichains>` object.
`...`	Further arguments passed to or from other methods.

Details

This returns the top rows of an `<epichains>` object, starting from the first known infectors.

To view the full output, use `as.data.frame(<object_name>)`.

Value

An object of class `<data.frame>`.

Author(s)

James M. Azam

Examples

```
set.seed(32)
chains_pois_offspring <- simulate_chains(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  lambda = 2
)
head(chains_pois_offspring)
set.seed(32)
chains_pois_offspring <- simulate_chains(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  lambda = 2
)
tail(chains_pois_offspring)
```

`likelihood` *Estimate the log-likelihood/likelihood for observed branching processes*

Description

Estimate the log-likelihood/likelihood for observed branching processes

Usage

```
likelihood(
  chains,
  statistic = c("size", "length"),
  offspring_dist,
  nsim_obs,
  obs_prob = 1,
  stat_threshold = Inf,
  log = TRUE,
  exclude = NULL,
  individual = FALSE,
  ...
)
```

Arguments

`chains`	Vector of chain summaries (sizes/lengths). Can be a `<numeric>` vector or an object of class `<epichains>` or `<epichains_summary>` (obtained from `simulate_chains()` or `simulate_chain_stats()`). See examples below.
`statistic`	The chain statistic to track as the stopping criteria for each chain being simulated when `stat_threshold` is not `Inf`; A `<string>`. It can be one of: • "size": the total number of cases produced by a chain before it goes extinct. • "length": the total number of generations reached by a chain before it goes extinct.
`offspring_dist`	Offspring distribution: a `<function>` like the ones provided by R to generate random numbers from given distributions (e.g., `rpois` for Poisson). More specifically, the function needs to accept at least one argument, `n`, which is the number of random numbers to generate. It can accept further arguments, which will be passed on to the random number generating functions. Examples that can be provided here are `rpois` for Poisson distributed offspring, `rnbinom` for negative binomial offspring, or custom functions.

`nsim_obs`	Number of simulations to be used to approximate the log-likelihood/likelihood if `obs_prob` < 1 (imperfect observation). If `obs_prob` == 1, this argument is ignored.
`obs_prob`	Observation probability. A number (probability) between 0 and 1. A value greater than 0 but less 1 implies imperfect observation and simulations will be used to approximate the (log)likelihood. This will also require specifying `nsim_obs`. In the simulation, the observation process is assumed to be constant.
`stat_threshold`	A stopping criterion for individual chain simulations; a positive number coercible to integer. When any chain's cumulative statistic reaches or surpasses `stat_threshold`, that chain ends. It also serves as a censoring limit so that results above the specified value, are set to `Inf`. Defaults to `Inf`. NOTE: For objects of class `<epichains>` or `<epichains_summary>`, the `stat_threshold` used in the simulation is extracted and used here so if this argument is specified, it is ignored and a warning is thrown.
`log`	Should the log-likelihoods be transformed to likelihoods? Logical. Defaults to `TRUE`.
`exclude`	A vector of indices of the sizes/lengths to exclude from the log-likelihood calculation.
`individual`	Logical; If TRUE, a vector of individual log-likelihood/likelihood contributions will be returned rather than the sum/product.
`...`	any parameters to pass to `simulate_chain_stats`

Value

If `log = TRUE`

- A joint log-likelihood (sum of individual log-likelihoods), if `individual == FALSE` (default) and `obs_prob == 1` (default), or
- A list of individual log-likelihoods, if `individual == TRUE` and `obs_prob == 1` (default), or
- A list of individual log-likelihoods (same length as `nsim_obs`), if `individual == TRUE` and `0 <= obs_prob < 1`, or
- A vector of joint log-likelihoods (same length as `nsim_obs`), if individual == FALSE and `0 <= obs_prob < 1` (imperfect observation).

If `log = FALSE`, the same structure of outputs as above are returned, except that likelihoods, instead of log-likelihoods, are calculated in all cases. Moreover, the joint likelihoods are the product, instead of the sum, of the individual likelihoods.

Author(s)

Sebastian Funk, James M. Azam

Examples

```
# example of observed chain sizes
set.seed(121)
# randomly generate 20 chains of size 1 to 10
```

```
chain_sizes <- sample(1:10, 20, replace = TRUE)
likelihood(
  chains = chain_sizes,
  statistic = "size",
  offspring_dist = rpois,
  nsim_obs = 100,
  lambda = 0.5
)
# Example using an <epichains> object
set.seed(32)
epichains_obj_eg <- simulate_chains(
  n_chains = 10,
  pop = 100,
  percent_immune = 0,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  mu = 2,
  size = 0.2
)

epichains_obj_eg_lik <- likelihood(
  chains = epichains_obj_eg,
  statistic = "size",
  offspring_dist = rnbinom,
  mu = 2,
  size = 0.2,
  stat_threshold = 10
)
epichains_obj_eg_lik

# Example using a <epichains_summary> object
set.seed(32)
epichains_summary_eg <- simulate_chain_stats(
  n_chains = 10,
  pop = 100,
  percent_immune = 0,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  mu = 2,
  size = 0.2
)
epichains_summary_eg_lik <- likelihood(
  chains = epichains_summary_eg,
  statistic = "size",
  offspring_dist = rnbinom,
  mu = 2,
  size = 0.2
)
epichains_summary_eg_lik
```

`print.epichains` *Print an* `<epichains>` *object*

Description

Print an `<epichains>` object

Usage

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

Arguments

`x`	An `<epichains>` object.
`...`	Other parameters passed to `print()`.

Value

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

Author(s)

James M. Azam

Examples

```
# Using a Poisson offspring distribution and simulating from an infinite
# population up to chain size 10.
set.seed(32)
chains_pois_offspring <- simulate_chains(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  lambda = 2
)
chains_pois_offspring # Print the object
```

print.epichains_summary

Print an `<epichains_summary>` *object*

Description

Prints a summary of the `<epichains_summary>` object. In particular, it prints the number of chains simulated, and the range of the statistic, represented as the maximum (`max_stat`) and minimum (`min_stat`). If the minimum or maximum is infinite, it is represented as `>= stat_threshold` where `stat_threshold` is the value of the censoring limit. See `?epichains_summary()` for the definition of `stat_threshold`.

Usage

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

Arguments

`x`	An `<epichains_summary>` object.
`...`	Not used.

Value

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

Author(s)

James M. Azam

Examples

```
# Using a Poisson offspring distribution and simulating from an infinite
# population up to chain size 10.
set.seed(32)
chain_summary_print_eg <- simulate_chain_stats(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  lambda = 2
)
chain_summary_print_eg # Print the object
```

`rborel` *Generate random numbers from the Borel distribution*

Description

Random numbers are generated by simulating from a Poisson branching process

Usage

```
rborel(n, mu, censor_at = Inf)
```

Arguments

`n`	Number of random variates to generate.
`mu`	A non-negative number for the poisson mean.
`censor_at`	A stopping criterion; `<numeric>`. Defaults to `Inf`. A value above which the simulation ends and the random number is set to `Inf` (as a form of censoring). `rborel()` simulates chain sizes using `simulate_chain_stats()` with a Poisson offspring distribution, so if `mu` >= 1, the simulation could proceed unendingly. This parameter is used to prevent this.

Value

A numeric vector of random numbers.

Author(s)

Sebastian Funk James M. Azam

Examples

```
set.seed(32)
rborel(5, 1)
```

`rgborel` *Generate random numbers from a Gamma-Borel mixture distribution*

Description

Generate random numbers from a Gamma-Borel mixture distribution

Usage

```
rgborel(n, size, prob, mu, censor_at = Inf)
```

Arguments

`n`	Number of random variates to generate.
`size`	The dispersion parameter (often called `k` in ecological applications); A positive number.
`prob`	Probability of success (in the parameterisation with `prob`, see also `NegBinomial`); A number between 0 and 1.
`mu`	Mean; A positive number.
`censor_at`	A stopping criterion; `<numeric>`. Defaults to `Inf`. A value above which the simulation ends and the random number is set to `Inf` (as a form of censoring). `rborel()` simulates chain sizes using `simulate_chain_stats()` with a Poisson offspring distribution, so if `mu` >= 1, the simulation could proceed unendingly. This parameter is used to prevent this.

Value

Numeric vector of random numbers

Author(s)

Sebastian Funk James M. Azam

Examples

```
set.seed(32)
rgborel(n = 5, size = 0.3, mu = 1, censor_at = 5)
```

`simulate_chains` *Simulate transmission chains*

Description

It generates independent transmission chains starting with a single case per chain, using a simple branching process model (See details for definition of "chains" and assumptions). Offspring for each chain are generated with an offspring distribution, and an optional generation time distribution function.

The individual chain simulations are controlled by customisable stopping criteria, including a threshold chain size or length, and a generation time cut off. The function also optionally accepts population related inputs such as the population size (defaults to Inf) and percentage of the population initially immune (defaults to 0).

Usage

```
simulate_chains(
  n_chains,
  statistic = c("size", "length"),
  offspring_dist,
  ...,
  stat_threshold = Inf,
  pop = Inf,
  percent_immune = 0,
  generation_time = NULL,
  t0 = 0,
  tf = Inf
)
```

Arguments

`n_chains`	Number of chains to simulate.
`statistic`	The chain statistic to track as the stopping criteria for each chain being simulated when `stat_threshold` is not `Inf`; A `<string>`. It can be one of: • "size": the total number of cases produced by a chain before it goes extinct. • "length": the total number of generations reached by a chain before it goes extinct.
`offspring_dist`	Offspring distribution: a `<function>` like the ones provided by R to generate random numbers from given distributions (e.g., `rpois` for Poisson). More specifically, the function needs to accept at least one argument, `n`, which is the number of random numbers to generate. It can accept further arguments, which will be passed on to the random number generating functions. Examples that can be provided here are `rpois` for Poisson distributed offspring, `rnbinom` for negative binomial offspring, or custom functions.
`...`	Parameters of the offspring distribution as required by R.
`stat_threshold`	A stopping criterion for individual chain simulations; a positive number coercible to integer. When any chain's cumulative statistic reaches or surpasses `stat_threshold`, that chain ends. Defaults to `Inf`. For example, if `statistic = "size"` and `stat_threshold = 10`, then any chain that produces 10 or more cases will stop. Note that setting `stat_threshold` does not guarantee that all chains will stop at the same value.
`pop`	Population size; An `<Integer>`. Used alongside `percent_immune` to define the susceptible population. Defaults to `Inf`.
`percent_immune`	Percent of the population immune to infection at the start of the simulation; A `<numeric>` between 0 and 1. Used alongside `pop` to initialise the susceptible population. Defaults to 0.
`generation_time`	The generation time function; the name of a user-defined named or anonymous function with only one argument `n`, representing the number of generation times to sample.

`t0`	Start time (if generation time is given); either a single value or a vector of same length as `n_chains` (number of initial cases) with corresponding initial times. Defaults to 0, meaning all cases started at time 0.
`tf`	A number for the cut-off for the infection times (if generation time is given); Defaults to `Inf`.

Value

An `<epichains>` object, which is basically a `<data.frame>` with columns:

- `chain` - an ID for active/ongoing chains,
- `infectee` - a unique ID for each infectee.
- `infector` - an ID for the infector of each infectee.
- `generation` - a discrete time point during which infection occurs, and optionally,
- `time` - the time of infection.

Definition of a transmission chain

A transmission chain as used here is an independent case and all the secondary cases linked to it through transmission. The chain starts with a single case, and each case in the chain generates secondary cases according to the offspring distribution. The chain ends when no more secondary cases are generated.

Calculating chain sizes and lengths

The function simulates the chain size for chain i at time t, $I_{t,i}$, as:

$$I_{t,i} = \sum_{i}^{I_{t-1}} X_{t,i},$$

and the chain length/duration for chain i at time t, $L_{t,i}$, as:

$$L_{t,i} = \mathsf{min}(1, X_{t,i}),$$

where $X_{t,i}$ is the secondary cases generated by chain i at time t, and $I_{0,i} = L_{0,i} = 1$.

The distribution of secondary cases, $X_{t,i}$ is modelled by the offspring distribution (`offspring_dist`).

Specifying `generation_time`

The argument `generation_time` must be specified as a function with one argument, `n`.

For example, assuming we want to specify the generation time as a random log-normally distributed variable with `meanlog = 0.58` and `sdlog = 1.58`, we could define a named function, let's call it "generation_time_fn", with only one argument representing the number of generation times to sample: `generation_time_fn <- function(n){rlnorm(n, 0.58, 1.38)}`, and assign the name of the function to `generation_time` in the simulation function, i.e. `` `simulate_*`(..., generation_time = generation_time_fn) ``, where `...` are the other arguments to `simulate_*()` and * is a placeholder for the rest of simulation function's name.

Alternatively, we could assign an anonymous function to `generation_time` in the `simulate_*()` call, i.e. `simulate_*(..., generation_time = function(n){rlnorm(n, 0.58, 1.38)})` OR `simulate_*(..., generation_time = \(n){rlnorm(n, 0.58, 1.38)})`, where `...` are the other arguments to `simulate_*()`.

Author(s)

James M. Azam, Sebastian Funk

References

Jacob C. (2010). Branching processes: their role in epidemiology. International journal of environmental research and public health, 7(3), 1186–1204. doi:10.3390/ijerph7031204

Blumberg, S., and J. O. Lloyd-Smith. 2013. "Comparing Methods for Estimating R0 from the Size Distribution of Subcritical Transmission Chains." Epidemics 5 (3): 131–45. doi:10.1016/j.epidem.2013.05.002.

Farrington, C. P., M. N. Kanaan, and N. J. Gay. 2003. "Branching Process Models for Surveillance of Infectious Diseases Controlled by Mass Vaccination.” Biostatistics (Oxford, England) 4 (2): 279–95. doi:10.1093/biostatistics/4.2.279.

Examples

```
# Using a Poisson offspring distribution and simulating from an infinite
# population up to chain size 10.
set.seed(32)
chains_pois_offspring <- simulate_chains(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  lambda = 2
)
chains_pois_offspring

# Using a Negative binomial offspring distribution and simulating from a
# finite population up to chain size 10.
set.seed(32)
chains_nbinom_offspring <- simulate_chains(
  n_chains = 10,
  pop = 100,
  percent_immune = 0,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  generation_time = function(n) rep(3, n),
  mu = 2,
  size = 0.2
)
chains_nbinom_offspring
```

simulate_chain_stats *Simulate a vector of transmission chains statistics (sizes/lengths)*

Description

It generates a vector of transmission chain sizes or lengths using the same model as `simulate_chains()` but without tracking details of the individual chains. This function is useful when only the chain sizes or lengths are of interest.

It uses a simple branching process model that simulates independent chains, using an offspring distribution for each chain. Each chain uses a threshold chain size or length as the stopping criterion especially where R0 > 1. The function also optionally accepts population related inputs such as the population size (defaults to Inf) and percentage of the population initially immune (defaults to 0).

Usage

```
simulate_chain_stats(
  n_chains,
  statistic = c("size", "length"),
  offspring_dist,
  ...,
  stat_threshold = Inf,
  pop = Inf,
  percent_immune = 0
)
```

Arguments

`n_chains`	Number of chains to simulate.
`statistic`	The chain statistic to track as the stopping criteria for each chain being simulated when `stat_threshold` is not `Inf`; A `<string>`. It can be one of: • "size": the total number of cases produced by a chain before it goes extinct. • "length": the total number of generations reached by a chain before it goes extinct.
`offspring_dist`	Offspring distribution: a `<function>` like the ones provided by R to generate random numbers from given distributions (e.g., `rpois` for Poisson). More specifically, the function needs to accept at least one argument, `n`, which is the number of random numbers to generate. It can accept further arguments, which will be passed on to the random number generating functions. Examples that can be provided here are `rpois` for Poisson distributed offspring, `rnbinom` for negative binomial offspring, or custom functions.
`...`	Parameters of the offspring distribution as required by R.
`stat_threshold`	A stopping criterion for individual chain simulations; a positive number coercible to integer. When any chain's cumulative statistic reaches or surpasses `stat_threshold`, that chain ends. It also serves as a censoring limit so that results above the specified value, are set to `Inf`. Defaults to `Inf`.

`pop`	Population size; An `<Integer>`. Used alongside `percent_immune` to define the susceptible population. Defaults to `Inf`.
`percent_immune`	Percent of the population immune to infection at the start of the simulation; A `<numeric>` between 0 and 1. Used alongside `pop` to initialise the susceptible population. Defaults to 0.

Value

An object of class `<epichains_summary>`, which is a numeric vector of chain sizes or lengths with extra attributes for storing the simulation parameters.

`simulate_chain_stats()` vs `simulate_chains()`

`simulate_chain_stats()` is a time-invariant version of `simulate_chains()`. In particular, it does not track the details of individual transmission events but deals with eventual chain statistics, that is, the statistic realised by a chain after dying out.

It is useful for generating a vector of chain sizes or lengths for a given number of chains, if details of who infected whom and the timing of infection are not of interest.

This function is used in `{epichains}` for calculating likelihoods in the `likelihood()` function and for sampling from the borel distribution (See ?epichains::rborel). It is used extensively in the vignette on modelling disease control, where only data on observed chain sizes and lengths are available.

Definition of a transmission chain

A transmission chain as used here is an independent case and all the secondary cases linked to it through transmission. The chain starts with a single case, and each case in the chain generates secondary cases according to the offspring distribution. The chain ends when no more secondary cases are generated.

Calculating chain sizes and lengths

The function simulates the chain size for chain i at time t, $I_{t,i}$, as:

$$I_{t,i} = \sum_{i}^{I_{t-1}} X_{t,i},$$

and the chain length/duration for chain i at time t, $L_{t,i}$, as:

$$L_{t,i} = \mathsf{min}(1, X_{t,i}),$$

where $X_{t,i}$ is the secondary cases generated by chain i at time t, and $I_{0,i} = L_{0,i} = 1$.

The distribution of secondary cases, $X_{t,i}$ is modelled by the offspring distribution (`offspring_dist`).

Author(s)

James M. Azam, Sebastian Funk

References

Jacob C. (2010). Branching processes: their role in epidemiology. International journal of environmental research and public health, 7(3), 1186–1204. doi:10.3390/ijerph7031204

Blumberg, S., and J. O. Lloyd-Smith. 2013. "Comparing Methods for Estimating R0 from the Size Distribution of Subcritical Transmission Chains." Epidemics 5 (3): 131–45. doi:10.1016/j.epidem.2013.05.002.

Farrington, C. P., M. N. Kanaan, and N. J. Gay. 2003. "Branching Process Models for Surveillance of Infectious Diseases Controlled by Mass Vaccination." Biostatistics (Oxford, England) 4 (2): 279–95. doi:10.1093/biostatistics/4.2.279.

Examples

```
# Simulate chain sizes with a poisson offspring distribution, assuming an
# infinite population and no immunity.
set.seed(32)
simulate_chain_stats(
  n_chains = 20,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  lambda = 0.9
)
# Simulate chain sizes with a negative binomial distribution and assuming
# a finite population and 10% immunity.
set.seed(32)
simulate_chain_stats(
  pop = 1000,
  percent_immune = 0.1,
  n_chains = 20,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  mu = 0.9,
  size = 0.36
)
```

`summary.epichains` *Summary method for* `<epichains>` *class*

Description

This calculates the chain statistic (size/length) for the simulated chains and returns an object with the same information as that returned by an equivalent `simulate_chain_stats()` call.

Usage

```
## S3 method for class 'epichains'
summary(object, ...)
```

Arguments

`object`	An `<epichains>` object.
`...`	Not used.

Value

An `<epichains_summary>` object containing the chain summary statistics as follows:

- "size": the total number of offspring produced by a chain before it goes extinct.
- "length": the number of generations achieved by a chain before it goes extinct.

Author(s)

James M. Azam

Examples

```
# Using a negative binomial offspring distribution and simulating from a
# finite population up to chain size 10.
set.seed(32)
sim_chains_nbinom <- simulate_chains(
  n_chains = 10,
  pop = 100,
  percent_immune = 0,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  mu = 2,
  size = 0.2
)
# Summarise the simulated chains
sim_chains_nbinom_summary <- summary(sim_chains_nbinom)
sim_chains_nbinom_summary

# Same results can be obtained using `simulate_chain_stats()`
set.seed(32)
sim_summary_nbinom <- simulate_chain_stats(
  n_chains = 10,
  pop = 100,
  percent_immune = 0,
  statistic = "size",
  offspring_dist = rnbinom,
  stat_threshold = 10,
  mu = 2,
  size = 0.2
)
sim_summary_nbinom

# Check that the results are the same
setequal(sim_chains_nbinom_summary, sim_summary_nbinom)
```

summary.epichains_summary

Summary method for `<epichains_summary>` *class*

Description

Summary method for `<epichains_summary>` class

Usage

```
## S3 method for class 'epichains_summary'
summary(object, ...)
```

Arguments

`object`	An `<epichains_summary>` object.
`...`	Not used.

Value

A list of chain summaries. The list contains the following elements:

- `n_chains`: the number of chains simulated.
- `max_stat`: the maximum chain statistic (size/length) achieved by the chains.
- `min_stat`: the minimum chain statistic (size/length) achieved by the chains.

Author(s)

James M. Azam

Examples

```
# Using a Poisson offspring distribution and simulating from an infinite
# population up to chain size 10.
set.seed(32)
chain_stats <- simulate_chain_stats(
  n_chains = 10,
  statistic = "size",
  offspring_dist = rpois,
  stat_threshold = 10,
  lambda = 2
)
summary(chain_stats)
```

Index