Package 'outbreaker2'

May 9, 2026

Version 1.1.4

Date 2025-12-16

Title Bayesian Reconstruction of Disease Outbreaks by Combining Epidemiologic and Genomic Data

Maintainer Finlay Campbell <finlaycampbell93@gmail.com>

Description Bayesian reconstruction of disease outbreaks using epidemiological and genetic information. Jombart T, Cori A, Didelot X, Cauchemez S, Fraser C and Ferguson N. 2014. <doi:10.1371/journal.pcbi.1003457>. Campbell, F, Cori A, Ferguson N, Jombart T. 2019. <doi:10.1371/journal.pcbi.1006930>.

License MIT + file LICENSE

Suggests coda, microbenchmark, testthat, knitr, rmarkdown, igraph, epicontacts, adegenet

Depends R (>= 3.5.0)

Imports utils, methods, stats, grDevices, Rcpp, ape, ggplot2, magrittr, visNetwork

LazyData true

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation yes

Author Thibaut Jombart [aut],
Finlay Campbell [aut, cre],
Rich Fitzjohn [aut],
Gerry Tonkin-Hill [ctb],
Kristjan Eldjarn [ctb],
Alexis Robert [ctb]

Repository CRAN

Date/Publication 2025-12-17 06:40:16 UTC

Contents

`bind_to_function` *Encloses argument in a function's environment*

Description

This function takes a function `f` and a series of named arguments, and returns a closure of `f` which will only rely on one single argument 'param'. This is used to reduce the number of arguments passed around to likelihood or movement functions. This functionality is used internally when creating closures of custom moves in `bind_moves`.

Usage

```
bind_to_function(f, ...)
```

Arguments

`f`	The function to which arguments are bound.
`...`	Named arguments to bind to the function's environment.

Author(s)

Initial code by Rich FitzJohn (see 'references') with some adaptations by Thibaut Jombart.

References

Initial code comes from the `partially_apply` function in the 'rodeint' package `richfitz/rodeint`.

chains_pal *Color palettes used in outbreaker*

Description

These functions are different color palettes (color-generating functions) used in outbreaker.

Usage

```
chains_pal(n)

cases_pal(n)
```

Arguments

n	a number of colors to be created

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>)

Examples

```
plot(1:8, col = chains_pal(8), cex = 10, pch = 20)
```

create_config *Set and check parameter settings of outbreaker*

Description

This function defines settings for outbreaker. It takes a list of named items as input, performs various checks, set defaults where arguments are missing, and return a correct list of settings. If no input is given, it returns the default settings.

Usage

```
create_config(..., data = NULL)

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

Arguments

...	further arguments to be passed to other methods.
data	an optional list of data items as returned by outbreaker_data; if provided, this allows for further checks of the outbreaker settings.
x	an outbreaker_config object as returned by create_config.

Details

Acceptables arguments for ... are:

init_tree the tree used to initialize the MCMC. Can be either a character string indicating how this tree should be computed, or a vector of integers corresponding to the tree itself, where the i-th value corresponds to the index of the ancestor of 'i' (i.e., init.tree[i] is the ancestor of case i). Accepted character strings are "seqTrack" (uses seqTrack algorithm to generate the initial tree - see function seqTrack in the package adegenet), "random" (ancestor randomly selected from preceding cases), and "star" (all cases coalesce to the first case). Note that for SeqTrack, all cases should have been sequenced.

init_alpha a vector of integers indicating the initial values of alpha, where the i-th value indicates the ancestor of case 'i'; defaults to NULL, in which ancestries are defined from init_tree.

init_kappa a (recycled) vector of integers indicating the initial values of kappa; defaults to 1.

init_t_inf a vector of integers indicating the initial values of t_inf, i.e. dates of infection; defaults to NULL, in which case the most likely t_inf will be determined from the delay to reporting/symptoms distribution, and the dates of reporting/symptoms, provided in data.

init_mu initial value for the mutation rates.

init_pi initial value for the reporting probability.

init_eps initial value for the contact reporting coverage.

init_lambda initial value for the non-infectious contact rate.

n_iter an integer indicating the number of iterations in the MCMC, including the burnin period.

move_alpha a vector of logicals indicating, for each case, if the ancestry should be estimated ('moved' in the MCMC), or not, defaulting to TRUE; the vector is recycled if needed.

move_t_inf a vector of logicals indicating, for each case, if the dates of infection should be estimated ('moved' in the MCMC), or not, defaulting to TRUE; the vector is recycled if needed.

move_mu a logical indicating whether the mutation rates should be estimated ('moved' in the MCMC), or not, all defaulting to TRUE.

move_pi a logical indicating whether the reporting probability should be estimated ('moved' in the MCMC), or not, all defaulting to TRUE.

move_eps a logical indicating whether the contact reporting coverage should be estimated ('moved' in the MCMC), or not at all, defaulting to TRUE.

move_lambda a logical indicating whether the non-infectious contact rate should be estimated ('moved' in the MCMC), or not at all, defaulting to TRUE.

move_kappa a logical indicating whether the number of generations between two successive cases should be estimated ('moved' in the MCMC), or not, all defaulting to TRUE.

move_pi a logical indicating whether the reporting probability should be estimated ('moved' in the MCMC), or not, all defaulting to TRUE.

n_iter the number of iterations of the MCMC.

sample_every the frequency at which MCMC samples are retained for the output.

sd_mu the standard deviation for the Normal proposal for the mutation rates.

sd_pi the standard deviation for the Normal proposal for the reporting probability.

sd_eps the standard deviation for the Normal proposal for the contact reporting coverage.

sd_lambda the standard deviation for the Normal proposal for the non-infectious contact rate.

prop_alpha_move the proportion of ancestries to move at each iteration of the MCMC.

prop_t_inf_move the proportion of infection dates to move at each iteration of the MCMC.

batch_size the size of the batch of random number pre-generated.

paranoid a logical indicating if the paranoid mode should be used; this mode is used for performing additional tests during outbreaker; it makes computations substantially slower and is mostly used for debugging purposes.

min_date earliest infection date possible, expressed as days since the first sampling.

max_kappa an integer indicating the largest number of generations between any two linked cases; defaults to 5.

prior_mu a numeric value indicating the rate of the exponential prior for the mutation rate 'mu'.

prior_pi a numeric vector of length 2 indicating the first and second parameter of the beta prior for the reporting probability 'pi'.

prior_eps a numeric vector of length 2 indicating the first and second parameter of the beta prior for the contact reporting coverage 'eps'.

prior_lambda a numeric vector of length 2 indicating the first and second parameter of the beta prior for the non-infectious contact rate 'lambda'.

ctd_directed a logical indicating if the contact tracing data is directed or not. If yes, the first column represents the infector and the second column the infectee. If ctd is provided as an epicontacts objects, directionality will be taken from there.

negative_si a logical indicating whether negative serial intervals are epidemiologically possible. If not, ancestries with negative serial intervals are discarded.

pb a logical indicating if a progress bar should be displayed.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

See Also

outbreaker_data to check and process data for outbreaker.

Examples

```
## see default settings
create_config()

## change defaults
create_config(move_alpha = FALSE, n_iter = 2e5, sample_every = 1000)
```

`create_param` *Initializes outputs for outbreaker*

Description

This function creates initial outputs and parameter states for outbreaker.

Usage

```
create_param(data = outbreaker_data(), config = create_config())
```

Arguments

`data`	A list of data items as returned by `outbreaker_data`, or arguments passed to this function.
`config`	A list of settings as returned by `create_config`, or arguments passed to this function.

Value

A list containing two components `$store` and `$current`. `store` is a list with the class `outbreaker_store`, used for storing 'saved' states of the MCMC. `current` is a list with the class `outbreaker_param`, used for storing 'current' states of the MCMC.

`outbreaker_store` class content:

- `size`: The length of the list, corresponding to the number of samples saved from the MCMC.
- `step`: A vector of integers of length `size`, storing the steps of the MCMC corresponding to the saved samples.
- `post`: A numeric vector of length `size`, storing log-posterior values.
- `like`: A numeric vector of length `size`, storing log-likelihood values.
- `prior`: A numeric vector of length `size`, storing log-prior values.
- `alpha`: A list of length `size`. Each item of the list is an integer vector of length `data$N`, storing indices (from 1 to N) of infectors for each case.

- t_inf: A list of length size. Each item of the list is an integer vector of length data$N, storing dates of infections for each case.
- mu: A numeric vector of length size, storing values of the mutation rate.
- kappa: A list of length size. Each item of the list is an integer vector of length data$N, storing the number of generations before the last sampled ancestor for each case.
- pi: A numeric vector of length size, storing values of the reporting probability.
- eps: A numeric vector of length size, storing values of the contact reporting coverage.
- lambda: A numeric vector of length size, storing values of the non-infectious contact rate.
- counter: A counter used to keep track of the current iteration of the MCMC (used internally).

outbreaker_param class content:

- alpha: An integer vector of length data$N, storing indices (from 1 to N) of infectors for each case.
- t_inf: An integer vector of length data$N, storing dates of infections for each case.
- mu: The value of the mutation rate.
- kappa: An integer vector of length data$N, storing the number of generations before the last sampled ancestor for each case.
- pi: The value of the reporting probability.
- eps: The value of the contact reporting coverage.
- lambda: The value of the non-infectious contact rate.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

Examples

```
## load data
x <- fake_outbreak
data <- outbreaker_data(dates = x$sample, dna = x$dna, w_dens = x$w)

## modify config settings
config <- create_config(move_alpha = FALSE, n_iter = 2e5, sample_every = 1000)

## create param object
param <- create_param(data = data, config = config)
```

`custom_likelihoods` *Customise likelihood functions for outbreaker*

Description

This function is used to specify customised likelihood functions for outbreaker. Custom functions are specified as a named list or series of comma-separated, named arguments, indicating which log-likelihood component they compute. Values currently available are:

Usage

```
custom_likelihoods(...)

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

Arguments

`...`	a named list of functions, each computing a log-likelihood component.
`x`	an `outbreaker_config` object as returned by `create_config`.

Details

- `genetic`: the genetic likelihood; by default, the function `cpp_ll_genetic` is used.
- `timing_sampling`: the likelihood of sampling times; by default, the function `cpp_ll_timing_sampling` is used.
- `timing_infections`: the likelihood of infection times; by default, the function `cpp_ll_timing_infections` is used.
- `reporting`: the likelihood of the reporting process; by default, the function `cpp_ll_reporting` is used.
- `contact`: the likelihood of the contact tracing data; by default, the function `cpp_ll_contact` is used.

All log-likelihood functions should have the following arguments, in this order:

- `data`: a list of named items containing input data as returned by `outbreaker_data`.
- `param`: a list of parameters with the class `create_param`.

Value

A named list of list(function, arity) pairs with the class `custom_likelihood`, each function implementing a customised log-likelihood component of outbreaker. Functions which are not customised will result in a list(NULL, 0) component. Any function with arity 3 must have the third parameter default to NULL.

a list of named functions

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

See Also

See customization vignette for detailed examples on how to customize likelihoods.

Examples

```
## specify a null model by disabling all likelihood components
f_null <- function(data, param) {
  return(0.0)
}

null_model <- custom_likelihoods(genetic = f_null,
                                 timing_sampling = f_null,
                                 timing_infections = f_null,
                                 reporting = f_null,
                                 contact = f_null)

null_config <- list(find_import = FALSE,
                    n_iter = 100,
                    sample_every = 1)

## load data
x <- fake_outbreak
data <- outbreaker_data(dates = x$sample, dna = x$dna, w_dens = x$w)

res_null <- outbreaker(data = data,
                       config = null_config,
                       likelihoods = null_model)

## visualise ancestries to see if all transmission trees have been explored
plot(res_null, type = "alpha")
```

---

custom_moves *Customise samplers for outbreaker*

Description

This function is used to specify customised movement functions (a.k.a. samplers) for outbreaker. Custom functions are specified as a named list or series of comma-separated, named arguments, indicating which type of movement they implement. Values currently available are:

Usage

```
custom_moves(...)

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

Arguments

...	A list or a series of named, comma-separated functions implementing movements of parameters or augmented data.
x	an outbreaker_moves object as returned by create_moves.

Details

- mu: movement of the mutation rate; by default, the function cpp_move_mu is used.
- pi: movement of the reporting probability; by default, the function cpp_move_pi is used.
- eps: movement of the contact reporting coverage; by default, the function cpp_move_eps is used.
- lambda: the movement of the non-infectious contact rate; the function cpp_move_lambda is used.
- alpha: movement of the transmission tree, by randomly proposing infectors in the pool of cases infected before; by default, the function cpp_move_alpha is used.
- swap_cases: movement of the transmission tree, by swapping infectors and infected cases; by default, the function cpp_move_swap_cases is used.
- t_inf: movement of the date of infection; by default, the function cpp_move_t_inf is used.
- kappa: movement of the number generations between cases; by default, the function cpp_move_kappa is used.

Movement functions must have an argument param, which is a list of parameters and augmented data of the class create_param. Each movement function will be enclosed with its other arguments, so that the resulting function will have a single argument 'param'. For non-standard movements (i.e. none of the names specified above), the closure will contain:

- data: a list of named items containing input data as returned by outbreaker_data.
- config: a list of named items containing input data as returned by create_config.
- likelihoods: a list of named custom likelihood functions as returned by custom_likelihoods.
- priors: a list of named custom prior functions as returned by custom_priors.

Value

A list of movement functions with a single argument 'param', with class outbreaker_moves.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

See Also

See customization vignette for detailed examples on how to customise movement functions.

`custom_priors` *Customise priors for outbreaker*

Description

Priors can be specified in several ways in outbreaker2 (see details and examples). The most flexible way to specify a prior is to provide a prior function directly. This function must take an argument 'param', which is a list which contains all the states of the parameters and augmented data. See the documentation of create_param for more information.

Usage

```
custom_priors(...)

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

Arguments

`...`	A list or a series of named, comma-separated functions implementing priors. Each function must have a single argument, which corresponds to a 'outbreaker_param' list.
`x`	an `outbreaker_config` object as returned by `create_config`.

Details

There are three ways a user can specify priors:

1) Default: this is what happens when the 'config' has default values of prior parameters.
2) Customized parameters: in this case, the prior functions are the default ones from the package, but will use custom parameters, specified by the user through `create_config`.

3) Customized functions: in this case, prior functions themselves are specified by the user, through the '...' argument of 'custom_priors'. The requirements is that such functions must have either hard-coded parameters or enclosed values. They will take a single argument which is a list containing all model parameters with the class 'outbreaker_param'. ALL PRIORS functions are expected to return values on a LOG SCALE.

Priors currently used for the model are:

- `mu` (mutation rate): default function is an exponential distribution implemented in `outbreaker:::cpp_prior_mu`. New prior functions should use `x$mu` to refer to the current value of `mu`, assuming their argument is called `x`.
- `pi` (reporting probability): default function is a beta distribution implemented in `outbreaker:::cpp_prior_pi`. New prior functions should use `x$pi` to refer to the current value of `pi`, assuming their argument is called `x`.

- eps (contact reporting coverage): default function is a beta distribution implemented in outbreaker:::cpp_prior_eps New prior functions should use x$eps to refer to the current value of eps, assuming their argument is called x.
- lambda (non-infectious contact rate): default function is a beta distribution implemented in outbreaker:::cpp_prior_lambda. New prior functions should use x$lambda to refer to the current value of lambda, assuming their argument is called x.

Value

A list of custom functions with class custom_priors. Values set to NULL will be ignored and default functions will be used instead.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

See Also

See customization vignette for detailed examples on how to customize priors.

Examples

```
## BASIC CONFIGURATION
custom_priors()


## SPECIFYING PRIOR PARAMETERS
## - this will need to be passed to outbreaker
default_config <- create_config()
new_config <- create_config(prior_mu = 1e-5,
                        prior_pi = c(2, 1))

## - to check the prior manually, default settings:
param <- list(mu = 0.001, pi = 0.9)
outbreaker2:::cpp_prior_mu(param, default_config)
outbreaker2:::cpp_prior_pi(param, default_config)

outbreaker2:::cpp_prior_mu(param, new_config)
outbreaker2:::cpp_prior_pi(param, new_config)

## these correspond to:
dexp(0.001, 0.01, log = TRUE)
dbeta(0.9, 2, 1, log = TRUE)


## SPECIFYING A PRIOR FUNCTION

## flat prior for pi between 0.5 and 1
f <- function(x) {ifelse(x$pi > 0.5, log(2), log(0))}
priors <- custom_priors(pi = f)
priors # this should be passed to outbreaker
```

```
## test the prior manually
priors$pi(list(pi=1))
priors$pi(list(pi=.6))
priors$pi(list(pi=.2))
priors$pi(list(pi=.49))
```

`fake_outbreak` *Small simulated outbreak*

Description

This dataset is a small (30 cases) simulated outbreak originally used to illustrate `outbreaker`, and used for the same purposes in `outbreaker2`. This list contains the following:

Usage

```
fake_outbreak
```

Format

An object of class `list` of length 6.

Details

- `$onset`: A vector of integers representing dates of onset.
- `$sample`: A vector of integers representing the dates of isolation.
- `$dna`: A DNAbin matrix of pathogen genome sequences.
- `$w`: A numeric vector giving the empirical distribution of the generation time.
- `$ances`: A vector of integers indicating, for each case, the true infectors. `NA` represents imported cases.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>

Examples

```
names(fake_outbreak)
fake_outbreak
```

`get_cpp_api` *Access internal C++ rountines used in outbreaker2*

Description

This function returns an environment containing all C++ functions (bound to R using Rcpp) used for priors, likelihoods, and movements of parameters in outbreaker2.

Usage

```
get_cpp_api()
```

Value

An environment containing Rcpp bindings to C++ functions used internally in outbreaker2. All functions are named as cpp_[type]_[component], where 'type' can be:

- 'prior': prior computation.
- 'll': likelihood computation.
- 'move': movement of parameters or augmented data.

And where 'component' can be:

- 'mu': (parameter) mutation rate.
- 'pi': (parameter) reporting probability.
- 'lambda': (parameter) non-infectious contact rate.
- 'eps': (parameter) contact reporting coverage.
- 'alpha': (augmented data) ancestries of the cases.
- 'kappa': (augmented data) generation between cases on transmission chains.
- 't_inf': (augmented data) dates of infections.
- 'timing_infections': (likelihood component) timing between infectors and infectees.
- 'timing_sampling': (likelihood component) timing between infection and reporting / isolation.
- 'timing': (likelihood component) sum of the two timing components.
- 'genetic': (likelihood component) genetic diversity accumulated along transmission chains.
- 'reporting': (likelihood component) reporting of cases.
- 'all': (likelihood component) sum of all likelihood components.
- 'swap_cases': (type of movement) swap infectors and infectees on a transmission tree.

For a description of the arguments of these functions, see the Rcpp_API vignette (`vignette("Rcpp_API", package = "outbreaker2")`).

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

Examples

```
## get functions in an environment
api <- get_cpp_api()
api

## check content of the environment
ls(api)

## test the prior for mu
args(api$cpp_prior_mu)

config <- create_config()

api$cpp_prior_mu(list(mu = 0.00123), config)

dexp(0.00123, rate = config$prior_mu, log = TRUE)
```

---

outbreaker *outbreaker2: main function for reconstructing disease outbreaks*

Description

The function outbreaker is the main function of the package. It runs processes various inputs (data, configuration settings, custom priors, likelihoods and movement functions) and explores the space of plausible transmission trees of a densely sampled outbreaks.

Usage

```
outbreaker(
  data = outbreaker_data(),
  config = create_config(),
  priors = custom_priors(),
  likelihoods = custom_likelihoods(),
  moves = custom_moves()
)
```

Arguments

data	a list of named items containing input data as returned by outbreaker_data.
config	a set of settings as returned by create_config.
priors	a set of log-prior functions as returned by custom_priors.
likelihoods	a set of log-likelihood functions as returned by custom_likelihoods.
moves	a set of movement functions as returned by custom_moves.

Details

The emphasis of 'outbreaker2' is on modularity, which enables customisation of priors, likelihoods and even movements of parameters and augmented data by the user. This the dedicated vignette on this topic `vignette("outbreaker2_custom")`.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

References

Jombart T, Cori A, Didelot X, Cauchemez S, Fraser C and Ferguson N (2014). Bayesian reconstruction of disease outbreaks by combining epidemiologic and genomic data. PLoS Computational Biology.

See Also

`outbreaker_data` to process input data, and `create_config` to process/set up parameters

- `outbreaker_data`: function to process input data.
- `create_config`: function to create default and customise configuration settings.
- `custom_priors`: function to specify customised prior functions.
- `custom_likelihoods`: function to specify customised likelihoods functions.
- `custom_moves`: function to create default and customise movement functions.

Examples

```
## get data
data(fake_outbreak)
dat <- fake_outbreak

## Not run:
## run outbreaker
out <- outbreaker(data = list(dna = dat$dna, dates = dat$onset, w_dens = dat$w),
config = list(n_iter = 2e4, sample_every = 200))
plot(out)
as.data.frame(out)

## run outbreaker, no DNA sequences
out2 <- outbreaker(data = list(dates = dat$onset, w_dens = w),
config = list(n_iter = 2e4, sample_every = 200))
plot(out2)
as.data.frame(out2)


## End(Not run)
```

`outbreaker_data` *Process input data for outbreaker*

Description

This function performs various checks on input data given to outbreaker. It takes a list of named items as input, performs various checks, set defaults where arguments are missing, and return a correct list of data input. If no input is given, it returns the default settings.

Usage

```
outbreaker_data(..., data = list(...))
```

Arguments

`...`	a list of data items to be processed (see description).
`data`	optionally, an existing list of data item as returned by `outbreaker_data`.

Details

Acceptables arguments for ... are:

dates dates a vector indicating the collection dates, provided either as integer numbers or in a usual date format such as `Date` or `POSIXct` format. By convention, zero will indicate the oldest date. If the vector is named, the vector names will be used for matching cases to contact tracing data and labelled DNA sequences.

dna the DNA sequences in `DNAbin` format (see `read.dna` in the ape package); this can be imported from a fasta file (extension .fa, .fas, or .fasta) using `adegenet`'s function fasta2DNAbin.

ctd the contact tracing data provided as a matrix/dataframe of two columns, indicating a reported contact between the two individuals whose ids are provided in a given row of the data, or an epicontacts object. In the case of the latter, linelist IDs will be used for matching dates and DNA sequences.

w_dens a vector of numeric values indicating the generation time distribution, reflecting the infectious potential of a case t = 1, 2, ... time steps after infection. By convention, it is assumed that newly infected patients cannot see new infections on the same time step. If not standardized, this distribution is rescaled to sum to 1.

f_dens similar to `w_dens`, except that this is the distribution of the colonization time, i_e. time interval during which the pathogen can be sampled from the patient.

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

Examples

```
x <- fake_outbreak
outbreaker_data(dates = x$sample, dna = x$dna, w_dens = x$w)
```

outbreaker_package *outbreaker2: a platform for disease outbreak reconstruction*

Description

This package provides a statistical platform for reconstructing transmission trees ('who infects whom') in densely sampled disease outbreaks. It reimplements, as a particular case, the model of 'outbreaker' (see references). 'outbreaker2' extends and replaces 'outbreaker'.

Details

The emphasis of 'outbreaker2' is on modularity, which enables customisation of priors, likelihoods and even movements of parameters and augmented data by the user. This the dedicated vignette on this topic `vignette("outbreaker2_custom")`.

The main functions of the package are:

- `outbreaker`: main function to run analyses.
- `outbreaker_data`: function to process input data.
- `create_config`: function to create default and customise configuration settings.
- `custom_priors`: function to specify customised prior functions.
- `custom_likelihoods`: function to specify customised likelihoods functions.
- `custom_moves`: function to create default and customise movement functions.

Author(s)

Thibaut Jombart <thibautjombart@gmail.com>.

print.outbreaker_chains *Basic methods for processing outbreaker results*

Description

Several methods are defined for instances of the class `outbreaker_chains`, returned by `outbreaker`, including: `print`, `plot`, `summary`

Usage

```
## S3 method for class 'outbreaker_chains'
print(x, n_row = 3, n_col = 8, ...)

## S3 method for class 'outbreaker_chains'
plot(
  x,
  y = "post",
  type = c("trace", "hist", "density", "alpha", "t_inf", "kappa", "network"),
  group = NULL,
  burnin = 0,
  min_support = 0.1,
  labels = NULL,
  ...
)

## S3 method for class 'outbreaker_chains'
summary(object, burnin = 0, method = c("mpa", "decycle"), ...)
```

Arguments

x	an outbreaker_chains object as returned by outbreaker.
n_row	the number of rows to display in head and tail; defaults to 3.
n_col	the number of columns to display; defaults to 8.
...	further arguments to be passed to other methods.
y	a character string indicating which element of an outbreaker_chains object to plot.
type	a character string indicating the kind of plot to be used (see details).
group	a vector of character strings indicating the parameters to display, or "all" to display all global parameters (non node-specific parameters).
burnin	the number of iterations to be discarded as burnin.
min_support	a number between 0 and 1 indicating the minimum support of ancestries to be plotted; only used if 'type' is 'network'.
labels	a vector of length N indicating the case labels (must be provided in the same order used for dates of symptom onset).
object	an outbreaker_chains object as returned by outbreaker.
method	the method used to determine consensus ancestries. 'mpa' (maximum posterior ancestry) simply returns the posterior ancestry with the highest posterior support for each case, even if this includes cycles. 'decycle' will return the maximum posterior ancestry, except when cycles are detected, in which case the link in the cycle with the lowest support is pruned and the tree recalculated.

Details

type indicates the type of graphic to plot:

- `trace` to visualise MCMC traces for parameters or augmented data (plots the log-likelihood by default)
- `hist` to plot histograms of quantitative values
- `density` to plot kernel density estimations of quantitative values
- `alpha` to visualise the posterior frequency of ancestries
- `network` to visualise the transmission tree; note that this opens up an interactive plot and requires a web browser with Javascript enabled; the argument 'min_support' is useful to select only the most supported ancestries and avoid displaying too many links
- `kappa` to visualise the distributions generations between cases and their ancestor/infector

Author(s)

Thibaut Jombart (<thibautjombart@gmail.com>).

See Also

See introduction vignette for detailed examples on how to visualise `outbreaker_chains` objects.

`sim_ctd` *Simulate contact data from a transmission tree*

Description

This function simulates contact data from a transmission tree. The model assumes that all transmission pairs have experienced contact, and that there is no false-positive reporting of contacts. The probability of contact occuring between a non-transmision pair is given by the parameter lambda. The probability of reporting a contact (transmission pair or not) is given by the parameters eps.

Usage

```
sim_ctd(tTree, eps, lambda)
```

Arguments

`tTree`	a dataframe or matrix of two columns, with each row providing the ids (numerical or as characters) of a transmission pair
`eps`	the contact reporting coverage, defined as the probability of reporting a contact (transmission pair or not)
`lambda`	the non-infectious contact rate, defined as the probability of contact between a non-transmission pair.

Author(s)

Finlay Campbell (<finlaycampbell93@gmail.com>)

Examples

```
## load data
x <- fake_outbreak
tTree <- data.frame(from = x$ances, to = seq_along(x$ances))

## simulate contact data with coverage of 80% and 10% probability of
## contact between non-transmission pairs
ctd <- outbreaker2:::sim_ctd(tTree, eps = 0.8, lambda = 0.1)

## inspect contact data
head(ctd)
```

Index