Package ‘outbreaker’

August 17, 2017

Version 1.1-8

Date 2017-08-17

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

Author Thibaut Jombart <thibautjombart@gmail.com>, Anne Cori, Joel Hellewell

Maintainer Thibaut Jombart <thibautjombart@gmail.com>

Depends R (>= 3.0.0), parallel

Imports utils, graphics, ape, igraph, adegenet

URL http://sites.google.com/site/therepiproject/r-pac/outbreaker

Description Bayesian reconstruction of disease outbreaks using epidemiological and genetic information.

License GPL (>= 2)

SystemRequirements gsl (>= 1.12)

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-08-17 10:57:55 UTC

R topics documented:

get.mu *Derive mutation rate estimation from outbreak's outputs*

Description

The function `get.mu` is used to obtain a distribution of the mutation rate from outbreaker's ouptput (functions `outbreaker` and `outbreaker.parallel`). The mutation rates used in outbreaker's model are expressed per generation of infection, which can be problematic to interprete biologically. `get.mu` derives classical estimates of the mutation rate per unit of time, with one value being estimated for each chain of the MCMC. By default, the mutation rate is expressed in number of nucleotide changes per unit time and per genome. If `genome.size` is provided, the mutation rate is expressed in number of nucleotide changes per unit time and per site.

Usage

```
get.mu(x, burnin = 20000, genome.size = NULL)
```

Arguments

`x`	the output of `outbreaker` or `outbreaker.parallel`.
`burnin`	an integer indicating the number of steps of the MCMC to be discarded as burnin period. Defaults to 20,000.
`genome.size`	the size of the genome; if not provided, mutation rate will be expressed in number of mutations per unit of time and per genome.

Value

A vector of mutation rates derived from the MCMC.

Author(s)

Thibaut Jombart <`t.jombart@imperial.ac.uk`>

Examples

```
## load data
data(fakeOutbreak)
attach(fakeOutbreak)

mu <- get.mu(res, genome.size=ncol(dat$dna))
hist(mu, col="grey",
     main="Inferred distribution of mu",
     xlab="mutations/site/day")
abline(v=1e-4,lty=2, lwd=4, col="royalblue")
mtext(side=3, "Dashed line = actual value")

detach(fakeOutbreak)
```

get.Rt *Derive reproduction numbers from outbreak's outputs*

Description

These functions are used to compute reproduction numbers and derive incidence curves from outbreaker's ouptput (functions `outbreaker` and `outbreaker.parallel`). They all rely on the entire outbreak having been sampled.

- `get.R` derive distributions of individual effective reproduction numbers.
- `get.Rt` derives effective reproduction numbers averaged for each time step.
- `get.incid` derives incidence curves for each time step.

Usage

```
get.Rt(x, burnin = 20000, plot = TRUE, type = c("boxplot", "lines"),
  lines = FALSE, fill.col = "gold", lines.col = transp("grey"), ...)

get.R(x, burnin = 20000, ...)

get.incid(x, burnin = 20000, plot = TRUE, type = c("boxplot", "lines"),
  lines = FALSE, fill.col = "gold", lines.col = transp("grey"), ...)
```

Arguments

`x` the output of `outbreaker` or `outbreaker.parallel`.

`burnin` an integer indicating the number of steps of the MCMC to be discarded as burnin period. Defaults to 20,000.

`plot` a logical indicating whether a plot should be displayed.

`type` a character indicating the type of plot to be used.

`lines` a logical indicating whether individual lines should be added to the plot.

`fill.col` the color to be used for the boxplot.

`lines.col` the color to be used to the lines.

`...` further arguments to be passed to other functions.

Value

These functions return a `data.frame` containing the plotted information.

Author(s)

Thibaut Jombart <`t.jombart@imperial.ac.uk`>

Examples

```
## load data
data(fakeOutbreak)
attach(fakeOutbreak)

## individual R
barplot(table(get.R(res)), main="Individual effective reproduction numbers")

## R(t)
get.Rt(res)

## incidence
get.incid(res)

detach(fakeOutbreak)
```

get.tTree *Simple transmission tree from outreaber's output*

Description

The S3 class tTree is used for storing simplified transmission trees, obtained from outbreaker's ouptput (functions outbreaker and outbreaker.parallel) using get.tTree. Some additional features are available for tTree objects, including plotting (plot), conversion to igraph graphs (as.igraph), and identification of mutations on the branches of the tree (findMutations).

Usage

```
get.tTree(x, burnin = 20000, best = c("ancestries", "tree"))

## S3 method for class 'tTree'
as.igraph(x, edge.col = "black", col.edge.by = "prob",
  col.pal = NULL, annot = c("dist", "n.gen", "prob"), sep = "/", ...)

## S3 method for class 'tTree'
findMutations(x, dna, ...)

## S3 method for class 'tTree'
plot(x, y = NULL, edge.col = "black",
  col.edge.by = "prob", col.pal = NULL, annot = c("dist", "n.gen",
  "prob"), sep = "/", ...)
```

Arguments

`x` for `get.tTree`, the output of `outbreaker` or `outbreaker.parallel`. For other functions, a `tTree` object.

`burnin` an integer indicating the number of steps of the MCMC to be discarded as burnin period. Defaults to 20,000.

`best` a character string matching "ancestries" or "tree", indicating which criterion is used to define the consensus tree; "ancestries" retains, for each case, the most supported ancestor; "tree" retains the most supported tree; note that the latter may exist only in the case of very small epidemics.

`edge.col` the color used for the edges; overriden if `col.edge.by` is provided.

`col.edge.by` a character string indicating how edges should be colored. Can be "dist" (by number of mutations), "n.gen" (by number of generations), or "prob" (by posterior support for the ancestries).

`col.pal` the palette of colors to be used for edges; if NULL, a grey palette is used, with larger values in darker shades.

`annot` same as `col.edge.by`, but specifies the information used to annotated the edges; several values can be provided, in which case different fields will be concatenated to generate the annotation.

`sep` a character indicating the separator for different field (see `annot`).

`...` further arguments to be passed to other functions.

`dna` a DNAbin object containing the aligned sequences of the isolates in the tree.

`y` unused - there for compatibility with the generic of `plot`.

Value

`tTree` objects are lists with the following components:

- idx: integer, the index of the cases
- collec.dates: the collection dates of the isolates
- idx.dna: the index of the cases to which each DNA sequence corresponds
- ances: the index of the inferred ancestor, for each case
- inf.dates: the inferred infection date, for each case
- p.ances: the posterior probability of the inferred ancestor (i.e., proportion in the posterior distribution of ancestors)
- nb.mut: the number of mutations between isolates and their inferred ancestor, for each isolate
- n.gen: the number of generations between isolates and their inferred ancestor, for each isolate
- p.gen: the posterior probability of the inferred number of generations between each case and its inferred ancestor
- inf.curves: the infectivity curves for each case

The plot function invisibly returns the conversion of the `tTree` object into a `igraph` graph.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
data(fakeOutbreak)
attach(fakeOutbreak)

## represent posterior ancestries
if(require(adegenet)){
transGraph(res, annot="", main="Posterior ancestries - support > 0.01",
   threshold=0.01, col.pal=spectral)
}
## get consensus ancestries
tre <- get.tTree(res)
plot(tre, annot="", main="Consensus ancestries")

## show match data/consensus ancestries
col <- rep("lightgrey", 30)
col[which(dat$ances != tre$ances)] <- "pink"
plot(tre, annot="", vertex.color=col, main="Consensus ancestries")
mtext(side=3, text="cases with erroneous ancestries in pink")


detach(fakeOutbreak)
```

---

outbreaker *Outbreaker: disease outbreak reconstruction using genetic data*

Description

outbreaker is a tool for the reconstruction of disease outbreaks using pathogens genome sequences. It relies on a probabilistic model of disease transmission which takes the genetic diversity, collection dates, duration of pathogen colonization and time interval between cases into account. It is embedded in a Bayesian framework which allows to estimate the distributions of parameters of interest. It currently allows to estimate:

- transmission trees
- dates of infection
- missing cases in a chain of transmission
- mutation rates
- imported cases
- (indirectly) effective reproduction numbers

Usage

```
outbreaker(dna = NULL, dates, idx.dna = NULL, mut.model = 1,
  spa.model = 0, w.dens, f.dens = w.dens, dist.mat = NULL,
  init.tree = c("seqTrack", "random", "star"), init.kappa = NULL,
  init.mu1 = NULL, init.mu2 = init.mu1, init.spa1 = NULL,
  n.iter = 1e+05, sample.every = 500, tune.every = 500, burnin = 20000,
  import.method = c("genetic", "full", "none"), find.import.n = 50,
  pi.prior1 = 10, pi.prior2 = 1, spa1.prior = 1, move.mut = TRUE,
  move.ances = TRUE, move.kappa = TRUE, move.Tinf = TRUE,
  move.pi = TRUE, move.spa = TRUE, outlier.threshold = 5,
  max.kappa = 10, quiet = TRUE, res.file.name = "chains.txt",
  tune.file.name = "tuning.txt", seed = NULL)

outbreaker.parallel(n.runs, parallel = TRUE, n.cores = NULL, dna = NULL,
  dates, idx.dna = NULL, mut.model = 1, spa.model = 0, w.dens,
  f.dens = w.dens, dist.mat = NULL, init.tree = c("seqTrack", "random",
  "star"), init.kappa = NULL, init.mu1 = NULL, init.mu2 = init.mu1,
  init.spa1 = NULL, n.iter = 1e+05, sample.every = 500,
  tune.every = 500, burnin = 20000, import.method = c("genetic", "full",
  "none"), find.import.n = 50, pi.prior1 = 10, pi.prior2 = 1,
  spa1.prior = 1, move.mut = TRUE, move.ances = TRUE, move.kappa = TRUE,
  move.Tinf = TRUE, move.pi = TRUE, move.spa = TRUE,
  outlier.threshold = 5, max.kappa = 10, quiet = TRUE,
  res.file.name = "chains.txt", tune.file.name = "tuning.txt",
  seed = NULL)
```

Arguments

`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; alternatively, a matrix of single characters strings, in which case only the mutation model 1 is available.

`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.

`idx.dna` an optional integer vector indicating to which case each dna sequence in `dna` corresponds. Not required if each case has a sequence, and the order of the sequences matches that of the cases.

`mut.model` an integer indicating the mutational model to be used; 1: one single mutation rate; 2: two rates, transitions (mu1) / transversions (mu2); if 'dna' is a sequence of character strings (not a DNAbin object), only the model 1 is available.

`spa.model` an integer indicating the spatial model to be used. 0: no spatial model (default). 1: exponential kernel (under development).

`w.dens` a vector of numeric values indicating the generation time distribution, reflecting the infectious potential of a case t=0, 1, 2, ... time steps after infection. By convention, w.dens[1]=0, meaning that an newly infected patient cannot be instantaneously infectious. 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.

`dist.mat` a matrix of pairwise spatial distances between the cases.

`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 output as initialize tree), "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.kappa` as `init.tree`, but values indicate the number of generations between each case and its most recent sampled ancestor.

`init.mu1, init.mu2`
initial values for the mutation rates (mu1: transitions; mu2: transversions).

`init.spa1` initial values of the spatial parameter.

`n.iter` an integer indicating the number of iterations in the MCMC, including the burnin period; defaults to `100,000`.

`sample.every` an integer indicating the frequency at which to sample from the MCMC, defaulting to 500 (i.e., output to file every 500 iterations).

`tune.every` an integer indicating the frequency at which proposal distributions are tuned, defaulting to 500 (i.e., tune proposal distribution every 500 iterations).

`burnin` an integer indicating the number of iterations for the burnin period, after which the chains are supposed to have mixed; estimated values of parameter are only relevant after the burnin period. Used only when imported cases are automatically detected.

`import.method` a character string indicating which method to use for detecting imported cases; available choices are 'gen' (based on genetic likelihood), 'full' (based on full likelihood), and 'none' (no imported case detection).

`find.import.n` an integer indicating how many chains should be used to determine imported cases; note that this corresponds to chains that are output after the burnin, so that a total of (burnin + output.every*find.import.n) chains will be used in the prior run to determine imported cases. Defaults to `50`.

`pi.prior1, pi.prior2`
two numeric values being the parameters of the Beta distribution used as a prior for π. This prior is Beta(10,1) by default, indicating that a majority of cases are likely to have been observed. Use Beta(1,1) for a flat prior.

`spa1.prior` parameters of the prior distribution for the spatial parameters. In the spatial model 1, `spa1.prior` is the mean of an exponential distribution.

`move.mut, move.pi, move.spa`
logicals indicating whether the named items should be estimated ('moved' in the MCMC), or not, all defaulting to TRUE. `move.mut` handles both mutation rates.

`move.ances, move.kappa, move.Tinf`
vectors of logicals of length 'n' indicating for which cases different components should be moved during the MCMC.

`outlier.threshold`
a numeric value indicating the threshold for detecting low likelihood values corresponding to imported cases. Outliers have a likelihood `outlier.threshold` smaller than the average.

`max.kappa` an integer indicating the maximum number of generations between a case and its most recent sampled ancestor; defaults to 10.

`quiet` a logical indicating whether messages should be displayed on the screen.

`res.file.name` a character string indicating the name of the file used to store MCMC outputs.

`tune.file.name` a character string indicating the name of the file used to store MCMC tuning outputs.

`seed` an integer used to set the random seed of the C procedures.

`n.runs` an integer indicating the number of independent chains to run, either in parallel (if `parallel` is used), or serially (otherwise).

`parallel` a logical indicating whether the package `parallel` should be used to run parallelized computations; by default, it is used if available.

`n.cores` an integer indicating the number of cores to be used for parallelized computations; if NULL (default value), then up to 6 cores are used, depending on availability.

Details

The function `outbreaker` is the basic implementation of the model. `outbreaker.parallel` allows to run several independent MCMC in parallel across different cores / processors of the same computer. This requires the base package `parallel`.

The spatial module implemented in outbreaker is currently under development. Please contact the author before using it.

For more resources including tutorials, forums, etc., see: http://sites.google.com/site/therepiproject/r-pac/outbreaker

Value

Both procedures return a list with the following components:

- chains: a data.frame containing MCMC outputs (which are also stored in the file indicated in `res.file.name`).
- collec.dates: (data) the collection dates.
- w: (data) the generation time distribution (argument `w.dens`)
- f: (data) the distribution of the time to collection (argument `f.dens`)
- D: a matrix of genetic distances (in number of mutations) between all pairs of sequences.
- idx.dna: (data) the index of the case each dna sequence corresponds to
- tune.end: an integer indicating at which iteration the proposal auto-tuning procedures all stopped.

- find.import: a logical indicating if imported cases were to be automatically detected.
- burnin: an integer indicating the pre-defined burnin, used when detecting imported cases.
- find.import.at: an integer indicating at which iteration of the preliminary MCMC imported cases were detected.
- n.runs: the number of independent runs used.
- call: the matched call.

Author(s)

Thibaut Jombart (<t.jombart@imperial.ac.uk>)

References

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

See Also

- plotChains to visualize MCMC chains.
- transGraph and get.tTree to represent transmission trees.
- get.R and get.Rt to get reproduction numbers distributions.
- get.incid to get estimates of incidence.
- get.mu to get the mutation rate distribution.
- simOutbreak to simulate outbreaks.
- selectChains to select chains from parallel runs which converged towards different posterior modes.
- fakeOutbreak, a toy dataset used to illustrate the method.
- For more resources including tutorials, forums, etc., see: http://sites.google.com/site/therepiproject/r-pac/outbreaker

Examples

```
## EXAMPLE USING TOYOUTBREAK ##
## LOAD DATA, SET RANDOM SEED
data(fakeOutbreak)
attach(fakeOutbreak)

## VISUALIZE DYNAMICS
matplot(dat$dynam, type="o", pch=20, lty=1,
   main="Outbreak dynamics", xlim=c(0,28))
legend("topright", legend=c("S","I","R"), lty=1, col=1:3)

## VISUALIZE TRANSMISSION TREE
plot(dat, annot="dist", main="Data - transmission tree")
```

```
mtext(side=3, "arrow annotations are numbers of mutations")


## Not run:
## RUN OUTBREAKER - PARALLEL VERSION
## (takes < 1 min))
set.seed(1)
res <-  outbreaker.parallel(n.runs=4, dna=dat$dna,
   dates=collecDates,w.dens=w, n.iter=5e4)

## End(Not run)


## ASSESS CONVERGENCE OF CHAINS
plotChains(res)
plotChains(res, burnin=2e4)

## REPRESENT POSTERIOR ANCESTRIES
transGraph(res, annot="", main="Posterior ancestries", thres=.01)

## GET CONSENSUS ANCESTRIES
tre <- get.tTree(res)
plot(tre, annot="", main="Consensus ancestries")

## SHOW DISCREPANCIES
col <- rep("lightgrey", 30)
col[which(dat$ances != tre$ances)] <- "pink"
plot(tre, annot="", vertex.color=col, main="Consensus ancestries")
mtext(side=3, text="cases with erroneous ancestries in pink")

## GET EFFECTIVE REPRODUCTION OVER TIME
get.Rt(res)

## GET INDIVIDUAL EFFECTIVE REPRODUCTION
head(get.R(res))
boxplot(get.R(res), col="grey", xlab="Case",
        ylab="Effective reproduction number")

## GET MUTATION RATE PER TIME UNIT
## per genome
head(get.mu(res))

## per nucleotide
mu <- get.mu(res, genome.size=1e4)
head(mu)

summary(mu)
hist(mu, border="lightgrey", col="grey", xlab="Mutation per day and nucleotide",
     main="Posterior distribution of mutation rate")

detach(fakeOutbreak)
```

outbreaker.package *outbreaker: disease outbreak reconstruction using epidemiological and genetic data*

Description

This package implements the model introduced by Jombart et al. (PLoS Comput. Biol, 2014) for disease outbreak reconstruction using epidemiological and genetic data.

Details

Check tutorials and documentation at: https://sites.google.com/site/therepiproject/r-pac/outbreaker

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

plotChains *Plot outbreaker's results*

Description

These are the main functions used for generating graphics from the raw output of outbreaker and outbreaker.parallel.

Usage

```
plotChains(x, what = "post", type = c("series", "density"), burnin = 0,
  dens.all = TRUE, col = funky(x$n.runs), lty = 1, lwd = 1,
  main = what, legend = TRUE, posi = "bottomleft", ...)

transGraph(x, labels = NULL, burnin = x$burnin, threshold = 0.2,
  col.pal = NULL, curved.edges = TRUE, annot = c("dist", "support"),
  sep = "/", ...)

plotOutbreak(x, burnin = x$burnin, thres.hide = 0.2, col = NULL,
  col.pal = colorRampPalette(c("blue", "lightgrey")), edge.col.pal = NULL,
  col.edge.by = "prob", annot = c("dist", "prob"), sep = "/",
  cex.bubble = 1, edge.max.dist = 10, lwd.arrow = 2, xlim = NULL, ...)
```

Arguments

`x` the output of `outbreaker` or `outbreaker.parallel`.

`what` a character chains giving the name of the item to be plotted. See `names(x$chains)` for possible values. By default, log-posterior values are plotted

`type` a character indicating if the chains should be plotted as time series ("series"), or as density ("density").

`burnin` an integer indicating the number of MCMC steps to discard before plotting chains.

`dens.all` a logical indicating if, in the case of multiple runs, the overall density of the different chains should be plotted in addition to individual densities.

`col` a vector of colors to be used to plot different chains.

`lty` a vector of integers specifying line types for the different chains.

`lwd` same as `lty`, but for line width.

`main` the title to be added to the plot.

`legend` a logical indicating if a legend should be plotted for the different runs.

`posi` a character string indicating the position of the legend (see `?legend`).

`...` further arguments to be passed to other functions.

`labels` the labels to be used to name the nodes of the graph (cases).

`threshold` the minimum support for ancestries to be plotted; 'support' is defined as the frequency of a given ancestor in the posterior distribution; defaults to 0.2.

`col.pal, edge.col.pal`
the color palette to be used for the edges (ancestries).

`curved.edges` a logical indicating whether edges should be curved.

`annot` a character indicating which information should be used to annotate the edges; this can be the distances between ancestors and descendents ("dist") and the posterior support for ancestries ("support"); if both are requested, fields will be concatenated.

`sep` a character indicating the separator to be used when concatenating several types of annotation.

`thres.hide` a threshold of posterior support for displaying ancestries; ancestries with less than this frequency in the posterior are hidden.

`col.edge.by` a character string indicating which information should be used to color the edges ('dist': genetic distance; 'prob': support for the ancestry)

`cex.bubble` a numeric value indicating the size factor for the bubbles representing the generation time distribution.

`edge.max.dist` a number indicating the threshold distance bounding the color palette used for the edges; useful to avoid showing edges corresponding to distances larger than a given number.

`lwd.arrow` a numeric value indicating the size factor for the arrows.

`xlim` the limits of the X axis; if NULL, determined from the data.

Details

- `plotChains` is used for plotting MCMCs
- `transGraph` plots a graph of inferred ancestries
- `plotOutbreak` attempts to synthetize the reconstruction of small outbreaks

Author(s)

Thibaut Jombart <`t.jombart@imperial.ac.uk`>

Examples

```
data(fakeOutbreak)
attach(fakeOutbreak)

## examine MCMC
plotChains(res)
plotChains(res,type="dens")
plotChains(res,type="dens", what="mu1", burnin=2e4)

## represent posterior ancestries
transGraph(res, annot="", main="Posterior ancestries")
transGraph(res, annot="", main="Posterior ancestries - support > 0.5",
   threshold=0.5)
if(require(adegenet)){
transGraph(res, annot="", main="Posterior ancestries - support > 0.01",
   threshold=0.01, col.pal=spectral)
}
## summary plot
plotOutbreak(res,cex.bubble=0.5, thres.hide=0.5,
   main="Outbreak reconstruction")


detach(fakeOutbreak)
```

`selectChains` *Select 'good' runs from independent MCMC chains*

Description

The function `selectChains` is used to discard 'bad' MCMC chains from outbreaker's ouptput (functions `outbreaker` and `outbreaker.parallel`). This is useful whenever several chains were run and converged towards different posterior modes or distributions. This can happen for instance when imported cases are hard to disentangle, resulting in different runs identifying different imports and therefore having different likelihood.

Usage

```
selectChains(x, select = "visual", alpha = 0.001, ...)
```

Arguments

`x` the output of `outbreaker` or `outbreaker.parallel`.

`select` a character string matching `visual` or `auto`, or a vector of integers indicating the runs to be discarded.

`alpha` the alpha threshold to be used to the automatic procedure (see details)

`...` further arguments to be passed to `plotChains`.

Details

Three modes are available, depending on the argument `select` (see also arguments below):

- `visual`: (default) interactive mode plotting the log-posterior values for the different chains and asking the user to identify runs to be discarded.
- `auto`: an automatic procedure is used to discard 'bad' runs; see details.
- `[numbers]`: numbers indicating the runs to be discarded.

The automatic procedure relies on the following recursive process:

- 1. Make the ANOVA of the log-posterior values as a function of the run identifier.
- 2a. If the P-value is greater than alpha (non-significant), exit.
- 2b. Otherwise, discard the run with the lowest mean log-posterior value, and go back to 1.

Value

These functions similar objects to the inputs, from which 'bad' runs have been discarded.

Author(s)

Thibaut Jombart <`t.jombart@imperial.ac.uk`>

`simOutbreak` *Simulation of pathogen genotypes during disease outbreaks*

Description

The function `simOutbreak` implements simulations of disease outbreaks. The infectivity of cases is defined by a generation time distribution. The function `as.igraph` allows to convert simulated transmission trees into `igraph` objects.

Usage

```
simOutbreak(R0, infec.curve, n.hosts = 200, duration = 50,
  seq.length = 10000, mu.transi = 1e-04, mu.transv = mu.transi/2,
  rate.import.case = 0.01, diverg.import = 10, group.freq = 1,
  spatial = FALSE, disp = 0.1, area.size = 10, reach = 1,
  plot = spatial, stop.once.cleared = TRUE)

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

## S3 method for class 'simOutbreak'
x[i, j, drop = FALSE]

## S3 method for class 'simOutbreak'
labels(object, ...)

## S3 method for class 'simOutbreak'
as.igraph(x, edge.col = "black", col.edge.by = "dist",
  vertex.col = "gold", edge.col.pal = NULL, annot = c("dist", "n.gen"),
  sep = "/", ...)

## S3 method for class 'simOutbreak'
plot(x, y = NULL, edge.col = "black",
  col.edge.by = "dist", vertex.col = "gold", edge.col.pal = NULL,
  annot = c("dist", "n.gen"), sep = "/", ...)
```

Arguments

`R0` the basic reproduction number; to use several groups, provide a vector with several values.

`infec.curve` a `numeric` vector describing the individual infectiousness at time t=0, 1, . . .

`n.hosts` the number of susceptible hosts at the begining of the outbreak

`duration` the number of time steps for which simulation is run

`seq.length` an integer indicating the length of the simulated haplotypes, in number of nucleotides.

`mu.transi` the rate of transitions, in number of mutation per site and per time unit.

`mu.transv` the rate of transversions, in number of mutation per site and per time unit.

`rate.import.case` the rate at which cases are imported at each time step.

`diverg.import` the number of time steps to the MRCA of all imported cases.

`group.freq` the frequency of the different groups; to use several groups, provide a vector with several values.

`spatial` a logical indicating if a spatial model should be used.

`disp` the magnitude of dispersal (standard deviation of a normal distribution).

`area.size` the size of the square area to be used for spatial simulations.

`reach` the mean of the exponential kernel used to determine new infections.

`plot` a logical indicating whether an animated plot of the outbreak should be displayed; only available with the spatial model.

`stop.once.cleared`
a logical indicating if the simulation should stop when the global force of infection is close to zero (<10^-12); TRUE by default.

`x, object` `simOutbreak` objects.

`...` further arguments to be passed to other methods

`i, j, drop` `i` is a vector used for subsetting the object. For instance, `i=1:3` will retain only the first three haplotypes of the outbreak. `j` and `drop` are only provided for compatibility, but not used.

`edge.col` the color of the edges of the plotted graph; overridden by `col.edge.by`.

`col.edge.by` a character indicating the type of information to be used to color the edges; currently, the only valid value is "dist" (distances, in number of mutations). Other values are ignored.

`vertex.col` the colors to be used for the vertices (i.e., cases).

`edge.col.pal` the color palette to be used for the edges; if NULL, a grey scale is used, with darker shades representing larger values.

`annot` a character indicating the information to be used to annotate the edges; currently accepted values are "dist" (genetic distances, in number of mutations), and "n.gen" (number of generations between cases).

`sep` a character used to separate fields used to annotate the edges, whenever more than one type of information is used for annotation.

`y` present for compatibility with the generic 'plot' method. Currently not used.

`col` the color of the vertices of the plotted graph.

Value

=== simOutbreak class ===
`simOutbreak` objects are lists containing the following slots:

- n: the number of cases in the outbreak
- dna: DNA sequences in the DNAbin matrix format
- dates: infection dates
- dynam: a data.frame containing, for each time step (row), the number of susceptible, infected, or recovered in the population.
- id: a vector of integers identifying the cases

- ances: a vector of integers identifying infectors ('ancestor')
- nmut: the number of mutations corresponding to each ancestry
- ngen: the number of generations corresponding to each ancestry
- call: the matched call

Author(s)

Implementation by Thibaut Jombart <t.jombart@imperial.ac.uk>.

Epidemiological model designed by Anne Cori and Thibaut Jombart.

Examples

```
## Not run:
dat <- list(n=0)

## simulate data with at least 30 cases
while(dat$n < 30){
   dat <- simOutbreak(R0 = 2, infec.curve = c(0, 1, 1, 1), n.hosts = 100)
}
dat

## plot first 30 cases
N <- dat$n
plot(dat[1:(min(N,30))], main="First 30 cases")
mtext(side=3, text="nb mutations / nb generations")

## plot a random subset (n=10) of the first cases
x <- dat[sample(1:min(N,30), 10, replace=FALSE)]
plot(x, main="Random sample of 10 of the first 30 cases")
mtext(side=3, text="nb mutations / nb generations")

## plot population dynamics
head(dat$dynam,15)
matplot(dat$dynam[1:max(dat$onset),],xlab="time",
   ylab="nb of individuals", pch=c("S","I","R"), type="b")


## spatial model
w <-  exp(-sqrt((1:40)))
x <- simOutbreak(2, w, spatial=TRUE,
                 duration=500, disp=0.1, reach=.2)

## spatial model, no dispersal
x <- simOutbreak(.5, w, spatial=TRUE,
                 duration=500, disp=0, reach=5)

## End(Not run)
```

simulated outbreak dataset

Toy outbreak dataset used to illustrate outbreaker

Description

This toy outbreak dataset was simulated using simOutbreak. This dataset is a list containing the following components:

- dat: the data, output of simOutbreak; see `dat$call` for the actual command line that was used.
- w: the generation time distribution.
- collecDates: simulated collection dates dates.
- res: the results of `outbreaker.parallel`; see `res$call` for the actual command line that was used.

Author(s)

Thibaut Jombart <t.jombart@imperial.ac.uk>

Examples

```
## Not run:
## COMMAND LINES TO GENERATE SIMILAR DATA ##
w <- c(0, 0.5, 1, 0.75)
## note: this works only if outbreak has at least 30 case
dat <- simOutbreak(R0 = 2, infec.curve = w, n.hosts = 100)[1:30]
collecDates <- dat$onset + sample(0:3, size=30, replace=TRUE, prob=w)

## End(Not run)

## EXAMPLE USING TOYOUTBREAK ##
## LOAD DATA, SET RANDOM SEED
data(fakeOutbreak)
attach(fakeOutbreak)

## VISUALIZE DYNAMICS
matplot(dat$dynam, type="o", pch=20, lty=1,
   main="Outbreak dynamics", xlim=c(0,28))
legend("topright", legend=c("S","I","R"), lty=1, col=1:3)

## VISUALIZE TRANSMISSION TREE
plot(dat, annot="dist", main="Data - transmission tree")
mtext(side=3, "arrow annotations are numbers of mutations")


## Not run:
```

```
## RUN OUTBREAKER - PARALLEL VERSION
## (takes < 1 min))
set.seed(1)
res <-  outbreaker.parallel(n.runs=4, dna=dat$dna,
   dates=collecDates,w.dens=w, n.iter=5e4)

## End(Not run)


## ASSESS CONVERGENCE OF CHAINS
plotChains(res)
plotChains(res, burnin=2e4)

## REPRESENT POSTERIOR ANCESTRIES
transGraph(res, annot="", main="Posterior ancestries", thres=.01)

## GET CONSENSUS ANCESTRIES
tre <- get.tTree(res)
plot(tre, annot="", main="Consensus ancestries")

## SHOW DISCREPANCIES
col <- rep("lightgrey", 30)
col[which(dat$ances != tre$ances)] <- "pink"
plot(tre, annot="", vertex.color=col, main="Consensus ancestries")
mtext(side=3, text="cases with erroneous ancestries in pink")

## GET EFFECTIVE REPRODUCTION OVER TIME
get.Rt(res)

## GET INDIVIDUAL EFFECTIVE REPRODUCTION
head(get.R(res))
boxplot(get.R(res), col="grey", xlab="Case",
        ylab="Effective reproduction number")

## GET MUTATION RATE PER TIME UNIT
## per genome
head(get.mu(res))

## per nucleotide
mu <- get.mu(res, genome.size=1e4)
head(mu)

summary(mu)
hist(mu, border="lightgrey", col="grey", xlab="Mutation per day and nucleotide",
     main="Posterior distribution of mutation rate")

detach(fakeOutbreak)
```

Index