Package ‘epibase’

January 11, 2013

Version 0.1-0

Date 2013/01/11

Title basic tools for the analysis of disease outbreaks.

Author {The Hackout team (In alphabetic order: David Aanensen, Marc Baguelin,Paul Birrell, Simon Cauchemez, Anton Camacho, Caroline Colijn, Anne Cori,Xavier Didelot, Ken Eames, Christophe Fraser, Simon Frost, Niel Hens,Joseph Hugues, Thibaut Jombart, Lulla Opatowski, Oliver Ratmann, Samuel Soubeyrand, Marc Suchard, Jacco Wallinga, Rolf Ypma)}

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

Suggests EpiEstim

Depends R (>= 2.10),methods,MASS,ape,networkDynamic,ggplot2,proto,sna,grImport,scales,plyr

Description basic tools for the analysis of disease outbreaks.

License GPL (>=2)

LazyLoad yes

R topics documented:

`fake_SARS_HK` *Fake dataset similar to the SARS outbreak in Hong Kong in 2003*

Description

This data set is a fake data set that was designed to reproduce interesting features of the SARS (severe acute respiratory syndrom) outbreak in Hong Kong in 2003. It has a similar size and dynamics, as well as realistic breakdown of cases according to their source of infection.

The dataset contains two objects.

`individualData` is a datafram containing individual identifiers and an exposure code that can take the following values:

- 1 if individual was infected in Prince of Wales hospital (where a large spreading event (SSE) occurred),
- 2 if individual was infected in Amoy Gardens residence (where another SSE occurred),
- 3 if individual was infected in another hospital,
- 4 if source of infection is unknown.

`clinicalData` is a list of the following dataframes:

- `hospitalisation` contains individuals ID and the date of admission and discharge from hospital,
- `death` contains individuals ID and their date of death,
- `exposure` contains individuals ID and dates of start and end of exposure, defined for individuals who are known to have been in contact with known infected individuals during this period of time,
- `symptoms` contains individuals ID and their dates of symptoms onset.

`HorseFluOutbreak` *Dataset from the Newmarket 2003 equine Influenza outbreak*

Description

These are the different datasets for the outbreak: HorseFluSeq: a FASTA sequence file with approx 2000 sequences obtained by cloning and Sanger sequencing (accession number, nucleotide sequence) HorseFluYardInfo: a file with the geo-location of each yard (yard identifier, decimal longitude, decimal latitude) HorseFluSeqID: Information about the sample that each sequence was taken from, multiple samples were taken from a few horses. (Accession, tube identidier, clone identifier) HorseFluShedding: Data on the viral shedding load for each sample (Tube identifier , viral shedding load, yard identifer, horse identifier, swad date) HorseFluHorseInfo: Limited information relating to the host (age, sex, date of first vaccinaton)

References

Hughes J, Allen RC, Baguelin M, Hampson K, Baillie GJ, et al. (2012) Transmission of Equine Influenza Virus during an Outbreak Is Characterized by Frequent Mixed Infections and Loose Transmission Bottlenecks. PLoS Pathog 8(12): e1003081. doi:10.1371/journal.ppat.1003081

`infectorTableToNetwork` *Convert transmission tree to a network*

Description

Convert transmission tree to a network

Usage

```
infectorTableToNetwork(transmissiontreeData)
```

Arguments

`transmissiontreeData`	Matrix of who infected whom

Value

Network of who infected whom

`obkData-class` *Formal class "obkData"*

Description

The class `obkData` is a formal (S4) class for storing data collected during outbreaks. This includes:

- individual data (age, sex, onset of symptoms, ...)
- sample data (swabs, serology, accession numbers, ...)
- genetic sequences
- contact information
- clinical data

Objects from the class obkData

`obkData` objects can be created using `new("obkData", ...)`, where '...' can be the following optional arguments (defaults are all `NULL`):

`individuals` a data.frame with a mandatory column named 'individualID', providing unique identifiers for the individuals.

`samples` a data.frame with 3 mandatory columns named 'individualID', 'sampleID', and 'date', providing identifiers for the individuals, for the samples, and dates. Dates must be provided in a way convertible to `Date` (see `?as.Date`). Default format for dates if in character strings is " argument date.format

`clinical` ... - to be filled

`dna` a list of DNA sequences in `DNAbin` or `character` format.

`contacts` ... - to be filled

`trees` a list of phylogenetic trees in the class `multiPhylo` (from the ape package)

Slots

obkData contain the following slots; note that in most cases, it is better to retrieve information via accessors (see below), rather than by accessing the slots manually. Empty slots are all NULL.

individuals: a data.frame containing individual information, with individual labels stored as row names.

samples: a data.frame containing sample information; the first three columns are ’individualID’, ’sampleID’, and ’date’.

clinical: ...

contacts: an object of the class obkContacts storing contact information.

dna: an object of the class obkSequences storing DNA sequences.

trees: an object of the class multiPhylo storing list of trees.

Methods

Here is a list of methods available for obkData objects. Most of these methods are accessors, that is, functions which are used to retrieve the content of the object. Specific manpages can exist for more complex functions. These are indicated by a ’*’ symbol next to the method’s name. This list also contains methods for conversion from obkData to other classes.

Author(s)

Thibaut Jombart, Simon Frost, Lulla Opatowski, Paul Birrell, Anne Cori, Marc Baguelin, Caroline Colijn... add your name/email here

Examples

```
## EMPTY OBJECT ##
new("obkData")

## INDIVIDUAL INFO ONLY ##
new("obkData", individuals=data.frame("individualID"=letters))
new("obkData", individuals=data.frame("individualID"=letters, age=1:26, 1:26))


samp <- data.frame(individualID=c(’toto’,’toto’,’titi’), sampleID=c(1,3,2), date=c("2001-02-13","2001-03-01"


## SAMPLE INFO ONLY ##
new("obkData", sample=samp)
new("obkData", sample=samp[,c(1:3)] )
new("obkData", sample=samp[,c(1:3,4,4,4)] )

## SAMPLE & INDIV INFO - MISSING INDIV ##
new("obkData", sample=samp[,c(1:3,4,4,4)] , individuals=data.frame("individualID"=letters, age=1:26))

## SAMPLE & INDIV INFO ##
ind <- data.frame("individualID"=c("toto","John Doe", "titi"), age=c(20,18,67), sex=c("m","m","?"))
new("obkData", sample=samp, ind=ind)


## DNA INFO, NOTHING ELSE ##
library(ape)
```

```
data(woodmouse)
dat.dna <- as.list(woodmouse)

new("obkData", dna=dat.dna) # should be empty

## SAMP + DNA INFO ##
samp <- data.frame(individualID=c('toto','toto','titi'), sampleID=c(1,3,2), date=c("2001-02-13","2001-03-01"

samp <- cbind.data.frame(samp, sequenceID=c(1,2,3))

## sequences given as indices
new("obkData", samples=samp, dna=dat.dna) # (note the nice sample ordering)

## sequences given as IDs
samp$sequenceID <- c("No304","No306","No305")
new("obkData", samples=samp, dna=dat.dna) # (note the nice sample ordering)

## sequences given as IDs, with wrong IDs
## samp$sequenceID <- c("No304","No306","Arrrhhh") # this would generate a meaningful error
## new("obkData", samples=samp, dna=dat.dna) # (note the nice sample ordering)


## multiple sequences per individual
samp$sequenceID <- c("No304","No306","No305")
samp <- samp[c(1,1,2,2,2,3),]
samp$sequenceID <- 1:6
new("obkData", samples=samp, dna=dat.dna)


## multiple sequences per individual, locus information
samp$locus <- c("gene1","gene2")[c(1,1,1,2,1,2)]
new("obkData", samples=samp, dna=dat.dna)
```

`obkSequences-class` *Formal class "obkSequences"*

Description

The class `obkSequences` is a formal (S4) class for storing a DNA sequences obtained from a sample during a disease outbreak. Sequences from different loci can be stored.

An `obkSequences` object can be constructed from a list of sequences (stored as `DNAbin` or `character` vectors), with optional information about loci.

Objects from the class obkSequences

`obkSequences` objects can be created by calls to `new("obkSequences", ...)`, where '...' can be the following arguments:

`dna` a list of DNA sequences in `DNAbin` or `character` format.

`locus` an optional vector indicating the locus of each sequences; its length must match that of the list of sequences.

Slots

The following slots are the content of instances of the class `obkSequences`; note that in most cases, it is better to retrieve information via accessors (see below), rather than by accessing the slots manually.

`dna`**:** a list of `DNAbin` matrices.

Methods

Here is a list of methods available for `obkSequences` objects. Most of these methods are accessors, that is, functions which are used to retrieve the content of the object. Specific manpages can exist for accessors with more than one argument. These are indicated by a '*' symbol next to the method's name. This list also contains methods for conversion from `obkSequences` to other classes.

show `signature(x = "obkSequences")`: printing of the object.

get.nlocus `signature(x = "obkSequences")`: returns the number of loci in the sample.

get.nsequences `signature(x = "obkSequences")`: returns the number of sequences in the sample.

get.locus `signature(x = "obkSequences")`: returns the names of the loci in the sample.

get.dna* `signature(x = "obkSequences")`: returns the dna sequences in the sample for a given locus (`locus` argument).

Author(s)

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

Examples

```
## THIS IS A TOY EXAMPLE ##
library(ape)
data(woodmouse)

## test constructor / show
new("obkSequences") # empty object
new("obkSequences", woodmouse) # no locus info
new("obkSequences", as.matrix(woodmouse), locus=rep(c('loc1', 'loc2', 'locXX'), c(10,4,1)))


## test accessors
x <- new("obkSequences", as.matrix(woodmouse), locus=rep(c('loc1', 'loc2', 'locXX'), c(10,4,1)))
get.dna(x, locus=1)
get.dna(x, locus="locXX")
get.nlocus(x)
get.nsequences(x)
```

`phylo2ggphy` *Function to convert phylogenies from the class 'phylo' to the class 'ggphy'*

Description

Function to convert phylogenies from the class 'phylo' to the class 'ggphy'

Usage

```
phylo2ggphy(phylo, tip_dates = NULL,
  branch_unit = "subst")
```

Arguments

`phylo`	an object of the class "phylo"
`tip_dates`	a vector containing the sample dates of the tip in "Date" format, the dates must be ordered like the tips
`branch_unit`	the unit of the branch. Either "year", "month", "day" or "subst". If a time unit is provided, together with tip_dates, then the x-axis of the phylogeny will be in the Date format

Author(s)

Anton Camacho

Examples

```
see misc/plot_ggphy_test.R
```

`phylofromtranstree` *Create phylogenetic tree from transmission tree*

Description

Create phylogenetic tree from transmission tree

Usage

```
phylofromtranstree(transmissiontreeData)
```

Arguments

`transmissiontreeData`	Matrix of who infected whom

Value

phylogenetic tree representing how samples of the infectious agents may be related `plotEpi` *Plot the number of susceptible, infected and recovered as a function of time*

Description

Plot the number of susceptible, infected and recovered as a function of time

Usage

```
plotEpi(S)
```

Arguments

`S`	Matrix containing the numbers to be plotted

`plotTranstree` *Plot transmission tree using graphviz*

Description

Plot transmission tree using graphviz

Usage

```
plotTranstree(dat)
```

Arguments

`dat`	Matrix of who infected whom

`plot_ggphy` *Function to plot phylogenies of the class 'ggphy'*

Description

Function to plot phylogenies of the class 'ggphy'

Usage

```
plot_ggphy(ggphy, tip_labels = F, tip_attribute = NULL,
  var_tip_labels = NULL, var_tip_colour = NULL)
```

Arguments

`ggphy`	An object of the class "ggphy"
`tip_labels`	Logical.Should tip labels be plotted?
`tip_attribute`	Dataframe with at least two columns. One column must contain the tip labels, the remainings are tip attributes
`var_tip_labels`	Character. The name of the column of tip_attribute that contains the tip labels.
`var_tip_colour`	Character. The name of the column of tip_attribute that contains the attribute to be colour-codded.

Author(s)

Anton Camacho

Examples

```
see misc/plot_ggphy_test.R
```

`read.annotated.nexus` *Read annotated tree file in NEXUS format*

Description

This function reads one or more annotated trees from a NEXUS formatted file.

Usage

```
  read.annotated.nexus(file)
```

Arguments

`file`	a file name specificed by either a variable of mode character or a double-quoted string

Details

See read.nexus in the ape package for a specification of NEXUS formatted tree files. This function additionally extracts BEAST annotations for all branches/nodes in the trees and returns these annotations as lists of lists in the resulting "phylo" objects

Value

An object of class "phylo" with an additional slot called annotations. This slot is a list indexed by the nodes.

Author(s)

Marc Suchard

Examples

```
see misc/annotated_example.R
```

`simuEpi` *Simulate an epidemic following a SIRS model*

Description

Simulate an epidemic following a SIRS model

Usage

```
simuEpi(N = 1000, D = 50, beta = 0.2, nu = 0.1, f = 0.5)
```

Arguments

`N`	Size of the population
`D`	Duration of simulation
`beta`	Rate of infection
`nu`	Rate of recovery
`f`	Rate of loss of immunity

Value

simulated epidemic

`singapore` *Dataset from a small SARS outbreak in Singapore in 2003*

Description

These are the different datasets for the outbreak:

singapore: a file with 14 sequences with dates

phylogeny: the phylogeny as recontructed in Liu et al. PLoS Med 2005

transmissionNetwork: a possible transmission tree inferred from the phylogeny reconstructed

References

Liu, J., et al. (2005). "SARS transmission pattern in Singapore reassessed by viral sequence variation analysis." PLoS Med 2(2): e43.

`testSimu` *Test function: simulate an epidemic and produce various plots*

Description

Test function: simulate an epidemic and produce various plots

Usage

```
testSimu()
```

Index