Package 'OutbreakTools'

August 11, 2014

Version 0.1-11

Date 2014/10/05

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 (>= 3.0.0), methods, ggplot2, network, knitr

Imports utils, RColorBrewer, ape, reshape2, sna, plyr, ggmap, scales,rjson, networkDynamic

Description basic tools for the analysis of disease outbreaks.

License GPL (>= 2)

LazyLoad yes

VignetteBuilder knitr

Collate 'auxClasses.R' 'generics.R' 'obkContacts.R' 'obkSequences.R' 'obkData.R' 'obkData_Accessors.R' 'subset.R' 'obkData_basics.R' 'uniqSequences.R' 'dna2uniqSequences.R' 'make.phylo.R' 'make.attributes.R' 'get.incidence.R' 'phylo2ggphy.R' 'plotggMST.R' 'plotggphy.R' 'plot.R' 'plotIndividualTimeline.R' 'simuEpi.R' 'utils.R' 'plotGeo.R' 'annotatedTreeReader.R' 'importFromJSON.R' 'zzz.R' 'OutbreakTools.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-11 12:37:37

R topics documented:

`OutbreakTools-package` *The OutbreakTools package*

Description

`OutbreakTools` (previously known as 'epibase') is a package providing basic tools for the analysis of disease outbreaks. Its main features lie in handling possibly very different types of data within a coherent framework, represented by the class obkData (for "outbreak data"). This class allows to store and manipulate data on samples, individuals, records interventions, genetic sequences, phylogenetic trees and contact networks, most of this information being time-stamped and possibly geo-referenced.

For a more complete overview of the package, look at the tutorial vignette `OutbreakTools` accessible using:
`vignette("OutbreakTools", package="OutbreakTools")`

The main features of `OutbreakTools` include:

- obkData: the main class of objects storing outbreak data; this formal class is provided with a number of accessors, documented in the same page.
- obkSequences: an auxiliary class for handling sequences from several genes.
- obkContacts: an auxiliary class for handling possibly dynamic contact networks between individuals.
- `subset.obkData`: (simply use `subset`) a method to subset the dataset using various criteria.
- `make.phylo`: create/plot phylogenies from the genes contained in an obkData object.
- `read.annotated.nexus`: read an annotated phylogeny into a phylo object.
- `plotggMST`: create/plot minimum spanning trees from the genes contained in an obkData object.
- `dna2uniqSequences`: derive unique sequences from a sequence alignment possibly containing identical genomes.
- `plotIndividualTimeline`: graphical representation of samples as times series.
- `plotGeo`: graphical representation of samples as geographic distributions.
- `plotggphy`: advanced graphical representation of phylogenetic trees using `ggplot2`.
- `simuEpi`: outbreak simulation using an SIR model and a simple model of sequence evolution.
- `datasets`: `HorseFlu`, `HorseFluRaw`, `ToyOutbreak`, `ToyOutbreakRaw`, `FluH1N1pdm2009`

To cite OutbreakTools, please use the reference given by `citation("OutbreakTools")`.

Details

Package:	OutbreakTools
Type:	Package
Version:	0.1-0
Date:	2014-01-10
License:	GPL (>=2)

Author(s)

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

See Also

OutbreakTools depends on or is related to several packages. The main non-base packages OutbreakTools relies on are:

- ape: phylogenetic reconstruction
- ggplot2: advanced graphics
- network/networkDynamics: network handling
- sna: social network analysis
- ggmap: various map tools for ggplot2

Other useful packages include:

- EpiEstim: reproduction number estimation
- adegenet: genetic data analysis
- outbreaker: outbreak reconstruction using pathogen sequences
- phangorn: maximum-likelihood phylogenetic reconstruction
- igraph: graph theory algorithms

`auxiliary classes` *Auxiliary classes for OutbreakTools*

Description

These formal (S4) classes are defined for internal use in `OutbreakTools`. They include unions of existing classes with the type `NULL`, and conversion of existing S3 classes into S4.

Author(s)

Thibaut Jombart

`dna2uniqSequences` *Convert a DNAbin with duplicated sequences to the class 'uniqSequences'*

Description

Function to convert a DNAbin from the class 'DNAbin' to the class 'uniqSequences'

Usage

```
dna2uniqSequences(dna)
```

Arguments

dna	an object of the class "DNAbin"

Author(s)

Joseph Hughes

Examples

```
### get uniq sequences for individual 49

data(HorseFlu)
x <- HorseFlu

### create a subset DNAbin
id49 <- subset(x,individuals="49") # warning message is returned
dna49 <- get.dna(id49)[[1]]

### get a particular sequence id with summary.seq$uniqseqID4[5]
### get number of sequences length(summary.seq$uniqseqID4)
uniqdna49 <- dna2uniqSequences(dna49)
```

FluH1N1pdm2009 *Dataset from the 2009 influenza A/H1N1 pandemic*

Description

This dataset is a `list` containing the following objects:

1. `individuals`: a data frame containing 514 individual IDs as well as their locations.
2. `samples`: a data frame containing 514 individual IDs, their sample IDs and dates as well as the IDs of the associated genetic sequences.
3. `dna`: a `DNAbin` object containing 514 genetic sequences of influenza A/H1N1/2009 haemagglutinin (HA).
4. `tree`: a `multiphylo` object containing the maximum posterior probability tree obtained via the Beast analysis of the 514 genetic sequences.

Author(s)

Anton Camacho

References

This dataset is part of Trevor Bedford's tutorial on the Beast software: *Inferring spatiotemporal dynamics of the H1N1 influenza pandemic from sequence data*, available at https://github.com/trvrb/influenza-dynamics-practical. In particular, the maximum posterior probability tree is taken from the sampled trees available at: https://github.com/trvrb/dynamics-practical/blob/master/output/pandemic_geo.trees.

Examples

```
## load the dataset
data(FluH1N1pdm2009)
attach(FluH1N1pdm2009)

head(individuals)
head(samples)


## create obkData object
x <- new("obkData", individuals = individuals, dna = FluH1N1pdm2009$dna,
      dna.individualID = samples$individualID, dna.date = samples$date,
      trees = FluH1N1pdm2009$trees)

## have a look at the summary
summary(x)

## plot the phylogeny
plotggphy(x, tip.color="location")

p <- plotggphy(x, ladderize = TRUE, branch.unit = "year")
p

detach(FluH1N1pdm2009)
```

`get.data` *Access data in "obkData" objects*

Description

`get.data` is a generic function with a method for obkData objects. It can be used to retrieve various information, known only by the name of the field looked for.

Usage

```
get.data(x, ...)

## S4 method for signature 'obkData'
get.data(x, data, where=NULL, drop=TRUE,
       showSource=FALSE, ...)
```

Arguments

`x`	an obkData object.
`data`	a character string indicating the name of the data field to look for.

where	an optional character string indicating the name of the slot in which the information should be looked for; if NULL, the method will look in all slots starting with @individuals, then @samples, and finally in @records.
drop	a logical indicating if a vector should be returned (TRUE), as opposed to a data.frame with on single column (FALSE).
showSource	a logical indicating if information about individualIDs and slot sources should be returned. If TRUE a dataframe with three columns will be returned, as opposed to a vector/single column if FALSE.
...	currently not used.

Author(s)

Thibaut Jombart, Lulla Opatowski

Examples

```
## LOAD DATA ##
data(ToyOutbreak)
ls()


## VARIOUS USE OF GET.DATA ##
# list all the data with name 'Sex' from the obkData object
get.data(ToyOutbreak, "Sex")

# list all the data with 'date' from the obkData object
get.data(ToyOutbreak, "date")
get.data(ToyOutbreak, "date", showSource=TRUE)
get.data(ToyOutbreak, "date", where="records")

# Extract from the obkData object a given field
x <- get.data(ToyOutbreak, "records")
names(x) # x contains the whole 'records' list
head(x$Fever)
x <- get.data(ToyOutbreak, "samples")
names(x) # x contains the whole 'samples' data.frame
```

---

get.dates *Retrieve dates data*

Description

get.dates is a generic function with a method for obkData, obkSequences and obkContacts objects. It can be used to retrieve dates stored in the object.

Usage

```
get.dates(x, ...)

get.ndates(x, ...)

## S4 method for signature 'obkData'
get.dates(x, data=c("all", "individuals", "records",
    "dna","context", "contacts"), ...)

## S4 method for signature 'obkData'
get.ndates(x, data=c("all", "individuals", "records",
    "dna","context", "contacts"), ...)

## S4 method for signature 'obkSequences'
get.dates(x, ...)

## S4 method for signature 'obkSequences'
get.ndates(x, ...)

## S4 method for signature 'obkContacts'
get.dates(x, ...)

## S4 method for signature 'obkContacts'
get.ndates(x, ...)
```

Arguments

x	an input object to seek dates from.
data	a character string indicating the name of the data field to look for. It can be 'all'(default), 'individuals', 'records' or 'context'.
...	currently not used.

Author(s)

Lulla Opatowski, Thibaut Jombart.

Examples

```
## LOAD DATA ##
data(ToyOutbreak)

## VARIOUS USE OF GET.DATES ##
get.dates(ToyOutbreak)

get.ndates(ToyOutbreak, "records")
get.dates(ToyOutbreak, "records")

get.ndates(ToyOutbreak, "contacts")
```

```
get.dates(ToyOutbreak, "contacts")
```

`get.incidence` *Compute incidence*

Description

`get.incidence` is a generic function with a method for `Date`, obkData, obkSequences and obkContacts objects. In obkSequences and obkContacts objects, it uses collection/occurence dates to derive incidence data. In obkData, more options are available as different types of information can be used. The procedure looks for a time-stamped data with a given named field, and as an option allows for selecting specific dates based on the values of that field, provided as a range (numeric data), a list of retained values, or a regular expression. This allows for defining 'positives' in a flexible way.

In all procedures, one can specify the first and last date, as well as the time interval to be used.

Usage

```
get.incidence(x, ...)

## S4 method for signature 'Date'
get.incidence(x, from=NULL, to=NULL,
          interval=1, add.zero=TRUE, ...)

## S4 method for signature 'obkSequences'
get.incidence(x, from=NULL, to=NULL,
          interval=1, add.zero=TRUE, ...)

## S4 method for signature 'obkContacts'
get.incidence(x, from=NULL, to=NULL,
          interval=1, add.zero=TRUE, ...)

## S4 method for signature 'obkData'
get.incidence(x, data, where=NULL,
          val.min=NULL, val.max=NULL, val.kept=NULL, regexp=NULL,
          from=NULL, to=NULL,
          interval=1, add.zero=TRUE, ...)
```

Arguments

`x`	an input object to seek incidence from.
`from`	the first date to be considered, in `Date` format, or a character string in the form 'YYYY-mm-dd' (e.g. 1982-08-20); if numeric, will be interpreted as number of days from the first date.

to	the last date to be considered, in Date format, or a character string in the form 'YYYY-mm-dd' (e.g. 1982-08-20); if numeric, will be interpreted as number of days from the first date.
interval	the time interval to be used to computed incidence, in number of days.
add.zero	a logical indicating if a zero should be added at the end of the output.
data	a character string indicating the name of the data field to look for, as in get.data.
where	an optional character string indicating the where the data field should be looked for, as in get.data.
val.min	an optional numeric value indicating the minimum value of the data to be retained in the incidence (e.g. minimum body temperature).
val.max	an optional numeric value indicating the maximum value of the data to be retained in the incidence (e.g. maximum CD4 counts).
val.kept	an optional vector containing all values to be retained in the incidence.
regexp	an optional character string providing a regular expression to be used to select which data are used in the incidence.
...	further arguments to be passed to 'grep', for the obkData procedure.

Details

Incidence is computed for intervals (\[D1;D2\[) which are named after the first date (D1), including the first date, and excluding the second one (D2).

Value

A data.frame with two columns, the first one being dates in the Date format, and the second being integers giving the incidence (number of new items for that date).

Author(s)

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

Examples

```
## SIMPLE EXAMPLE ##
dates <- as.Date("2001-01-01") + c(0,1,2,1,1,4)
get.incidence(dates)
get.incidence(dates, interval=2)
get.incidence(dates, interval=2, from="2001-01-03")

## OBKDATA/OBKCONTACTS/OBKSEQUENCES OBJECTS ##
data(ToyOutbreak)

## incidence of DNA sequences collection
get.incidence(ToyOutbreak, "dna")
plot(get.incidence(ToyOutbreak, "dna"), type="s",
     main="Incidence of DNA sequences collection")
```

```
## incidence of contacts
get.incidence(ToyOutbreak, "contacts")
plot(get.incidence(ToyOutbreak, "contacts"), type="s",
     main="Incidence of contact occurence")


## incidence of temperature measurements
get.incidence(ToyOutbreak, "temperature")

## same, keeping temperatures above 39
get.incidence(ToyOutbreak, "temperature", val.min=39)
plot(get.incidence(ToyOutbreak, "temperature", val.min=39),
     type="s", main="Incidence: temperature>=39")

## same, temperature>40, interval=2days
inc <- get.incidence(ToyOutbreak, "temperature", val.min=40, interval=2)
plot(inc, type="s", main="Incidence: temperature>=40")
```

`get.individuals` *Retrieve individual identifiers*

Description

`get.individuals` is a generic function with a method for obkData, obkSequences and obkContacts objects. It can be used to retrieve individuals identifiers stored in the object.

Usage

```
get.individuals(x, ...)

get.nindividuals(x, ...)

## S4 method for signature 'obkData'
get.individuals(x,
    data=c("all", "individuals", "records", "contacts", "dna"), ...)

## S4 method for signature 'obkData'
get.nindividuals(x,
    data=c("all", "individuals", "records", "contacts", "dna"), ...)

## S4 method for signature 'obkSequences'
get.individuals(x, ...)

## S4 method for signature 'obkSequences'
get.nindividuals(x, ...)

## S4 method for signature 'obkContacts'
```

```
get.individuals(x, ...)

## S4 method for signature 'obkContacts'
get.nindividuals(x, ...)
```

Arguments

x	an input object to seek individuals from.
data	a character string indicating the name of the data field to look for. It can be 'all'(default), 'individuals', 'records' or 'context'.
...	currently not used.

Author(s)

Thibaut Jombart, Lulla Opatowski, Simon Frost.

Examples

```
## LOAD DATA ##
data(ToyOutbreak)

## VARIOUS USE OF GET.INDIVIDUALS ##
get.individuals(ToyOutbreak)

get.nindividuals(ToyOutbreak, "records")
get.individuals(ToyOutbreak, "records")

get.nindividuals(ToyOutbreak, "contacts")
get.individuals(ToyOutbreak, "contacts")
```

HorseFlu *Dataset from the Newmarket 2003 equine influenza outbreak*

Description

This dataset is an obkData object with the following components:

- individuals: a dataframe which contains all static information relating to a horse (individualID), training yard the horse belongs to (yardID), date of birth, sex, latitude, longitude.
- dna: an obkSequences object containing DNA sequences and their meta-information.
- records: a list of dataframes containing information about the dates of first and last vaccinations, and shedding data (viral copy number) for some samples.

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

Examples

```
## Not run:

## LOAD DATA ##
data(HorseFlu)

## EXAMINE CONTENT ##
summary(HorseFlu)

## individual info
head(HorseFlu@individuals)

## DNA sequences
HorseFlu@dna

## records info
lapply(HorseFlu@records, head)


## How many individuals and sequences?
get.nindividuals(HorseFlu)
get.nsequences(HorseFlu)

## How many sequences per individual?
ind <- table(get.data(HorseFlu, "individualID", where="dna"))
ind
barplot(sort(ind), horiz=TRUE, las=1,
       xlab="number of samples", cex.names=.8)

## How many sequences for this individual?
ind.42 <- subset(HorseFlu, individualID="42")
get.nsequences(ind.42)

## How many samples?
length(unique(get.data(HorseFlu, "sampleID", where="dna")))

## How many sequences per sample?
table(get.data(HorseFlu, "sampleID", where="dna"))



## End(Not run)
```

HorseFluRaw *Raw dataset from the Newmarket 2003 equine influenza outbreak*

Description

This dataset is a list containing the following components:

- `dna`: a DNAbin object alignment of FASTA sequences obtained by cloning and Sanger sequencing (sequenceID, nucleotide sequence); sequence labels contain sequence identifiers, individual IDs, and dates, separated by "_".
- `individuals`: a dataframe which contains all static information relating to a horse (individualID), training yard the horse belongs to (yardID), date of birth, sex, latitude, longitude.
- `records`: a list of dataframes containing information about the dates of first and last vaccinations, and shedding data (viral copy number) for some samples.
- `dna.info`: an indication of the samples from which the DNA sequences were obtained.

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

Examples

```
## Not run:
## LOAD DATA ##
data(HorseFluRaw)
attach(HorseFluRaw)

## EXAMINE CONTENT ##

## individual info
head(individuals)

## records info
lapply(records, head)

## DNA sequences
dna

## CREATE OBKDATA ##
x <- new("obkData", individuals=individuals, records=records,
         dna=dna, sampleID=dna.info$sampleID)


## MANIPULATE OBJECT ##

## How many individuals and sequences?
```

```
get.nindividuals(x)
get.nsequences(x)

## How many sequences per individual?
ind <- table(get.data(x, "individualID", where="dna"))
ind
barplot(sort(ind), horiz=TRUE, las=1,
       xlab="number of samples", cex.names=.8)

## How many sequences for this individual?
ind.42 <- subset(x, individualID="42")
get.nsequences(ind.42)

## How many samples?
length(unique(get.data(x, "sampleID", where="dna")))

## How many sequences per sample?
table(get.data(x, "sampleID", where="dna"))

detach(HorseFluRaw)



## End(Not run)
```

JSON2obkData *Import data from JSON file into OutbreakTools*

Description

JSON2obkData imports outbreak data stored in one or several JSON files and converts it into an obkData object.

! This feature is currently experimental !

Usage

```
JSON2obkData(individuals=NULL, records=NULL, contacts=NULL, context=NULL)
```

Arguments

individuals	a character string containing data on individuals in JSON format
records	a character string containing data on records and samples in JSON format
contacts	a character string containing data on contacts in JSON format
context	a character string containing data on context in JSON format

Value

An obkData object.

Author(s)

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

`make.phylo` *Obtain phylogenies from outbreak data*

Description

The function `make.phylo` is used to obtain phylogenies for each of the genes sequenced in an obkData object. Phylogenetic trees can optionally be plotted before being returned. Colors are used to distinguish samples, individuals, dates, or any other requested information. `make.phylo` is a wrapper for functions from the ape package: `dist.dna` for computing genetic distances, and specific methods for getting trees such as `nj`.

Usage

```
make.phylo(x, ...)

## S4 method for signature 'obkData'
make.phylo(x, locus=NULL,
           result=c("obkData","multiPhylo"), model = "N",
           pairwise.deletion = FALSE, method=nj,
           plot=FALSE, ask=TRUE,...)
```

Arguments

`x`	an obkData object.
`result`	a character string indicating the type of result to be returned; if `obkData`, then the phylogenies are stored in `x@trees`, possibly removing previous phylogenies; if `multiPhylo`, the result is a list of `phylo` objects with the class `multiPhylo`.
`locus`	a character or integer vector indicating the loci to be used; if NULL, all loci are used, producing one phylogeny each; see `get.dna`.
`model, pairwise.deletion`	arguments passed to `dist.dna`.
`method`	a function producing phylogenetic trees with the class `phylo` (ape package) from a distance matrix with class `dist`; examples include `nj`, `bionj`, `fastme.bal`, and `fastme.ols`.
`plot`	a logical indicating whether plots of the trees should be produced.
`ask`	a logical indicating whether user input must be waited for before drawing new plots.
`...`	further graphical arguments to be passed to `plot.phylo`.

Value

A `obkData` with new phylogenies, or a list with class `multiPhylo`.

Author(s)

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

See Also

read.annotated.tree and read.annotated.nexus to read annotated phylogenies into phylo objects.

Examples

```
## Example using simulate outbreak ##

set.seed(3)
x <- simuEpi(N = 200, D = 20, beta = 0.002, nu = 0.1, mu = 0.002)$x
get.trees(x) # no tree here

## GET TREE AND DISPLAY IT
x <- make.phylo(x, plot=TRUE)
get.trees(x) # newly created trees


## DISPLAY TREE USING PLOTGGPHY

## root tree to first case ##
if(require(ape)){
tre <- root(get.trees(x)[[1]],1)
}

## plot tree, use color for dates ##
par(mar=c(1,1,4,1))
plot(tre, edge.width=2, type="fan", show.tip=FALSE)
title("NJ tree of a simulated outbreak")
mtext("each disk indicates 1 mutation from the root")
```

---

make.tip.attributes *Obtain meta data for items in* obkData *objects*

Description

Thes functions derive all available data relating to genetic sequences, tips of a tree, or individuals, based on matching individuals identifiers and collection dates. Note that for individuals, repeated measures will result in a 'wide' format for a data.frame, with possibly many missing entries resulting from uneven sampling.

Usage

```
make.sequence.attributes(x)

make.tip.attributes(x, which.tree=1)

make.individual.attributes(x)
```

Arguments

x	an obkData object.
which.tree	an integer indicating which tree to use.

Value

A data.frame named after the sequences, the tips, or the individuals, with various meta-information.

Author(s)

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

Examples

```
data(ToyOutbreak)
data(HorseFlu)

head(make.tip.attributes(ToyOutbreak))

head(make.sequence.attributes(HorseFlu))

## no repeated measures
head(make.individual.attributes(ToyOutbreak))

## note wide format when there are repeated measures
head(make.individual.attributes(HorseFlu))
```

---

obkContacts-class *Formal class "obkContacts"*

Description

The class obkContacts is a formal (S4) class for storing contact data between individuals. Contacts can be static, or change in time (dynamic) at specified dates. Contacts are handled as network and networkDynamic objects, for static and dynamic contacts, respectively. See the corresponding packages for more information on this data structure.

An obkContacts object can be constructed from individual identifiers (stored as character vectors), representing the 'sender' and 'receiver' individuals, whether the contacts should be regarded as directed, with optional information about the timing of contacts.

Usage

```
get.contacts(x, ...)
get.ncontacts(x, ...)
## S4 method for signature 'obkContacts'
get.ncontacts(x, from=NULL, to=NULL, ...)
## S4 method for signature 'obkContacts'
get.contacts(x, from=NULL, to=NULL, ...)

## S4 method for signature 'obkContacts'
show(object)

## S4 method for signature 'obkContacts'
plot(x, y=NULL, labels=get.individuals(x),
...)

## S4 method for signature 'obkContacts'
as.matrix(x,
         matrix.type=c("adjacency","incidence","edgelist"),
         use.labels=TRUE, ...)

## S4 method for signature 'obkContacts'
as.data.frame(x, row.names=NULL, optional=FALSE,
         use.labels=TRUE, ...)
```

Arguments

x,object	an obkContacts object.
y	used for compatibility with the generic definition of plot; unused.
labels	a vector of characters giving annotation for the vertices.
matrix.type	a character indicating which type of matrix to extract(adjacency, incidence, or edge matrix).
from,to	dates indicating the time span to consider for active contacts (for dynamic networks only).
...	arguments passed to other methods; currently not used, except for plot.
row.names,optional	unused arguments, there for compatibility with the generic as.data.frame.
use.labels	a logical indicating whether labels should be used to indicate individuals, as opposed to their index in the network.

Objects from the class obkContacts

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

from a vector of individual identifiers in character format representing the source of the contact.

`to` a vector of individual identifiers in `character` format representing the sink of the contact.

`directed` a Boolean for whether contacts should be regarded as directed (`TRUE`) or not (`FALSE`).

`start` an optional vector of times representing the start of a contact; can be dates or numeric values.

`end` an optional vector of times representing the end of a contact; can be dates or numeric values..

`duration` an optional vector of durations of a contact, as an alternative way of representing dynamic contacts than using `end`; if numeric values are used together with dates in 'start' and 'end', these will be interpreted as numbers of days.

Slots

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

`contacts`**:** an object of class `network` or `networkDynamic`.

`origin`**:** an object of class `Date` indicating the origin of the contacts (i.e., this corresponds to "day 0".

Methods

Here is a list of methods available for `obkContacts` 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 `obkContacts` to other classes.

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

plot `signature(x = "obkContacts", y=NULL)`: plot the contact network; relies on `plot.network`, to which arguments can be passed via `...`.

get.ncontacts `signature(x = "obkContacts")`: returns the number of contacts; for dynamic contact networks, starting dates can be specified by the argument `from`, and end date by the argument `to`.

get.contacts `signature(x = "obkContacts")`: returns the contacts (as `network` or `networkDynamic`); for dynamic contact networks, starting dates can be specified by the argument `from`, and end date by the argument `to`.

get.nindividuals* `signature(x = "obkContacts")`: returns the number of individuals in the contact data (see manpage of `get.individuals` for details)..

get.individuals* `signature(x = "obkContacts")`: returns the identifiers of individuals in the contact data(see manpage of `get.individuals` for details)..

get.ndates* `signature(x = "obkContacts")`: returns the number of dates at which contact structure changes (see manpage of `get.dates` for details).

get.dates* `signature(x = "obkContacts")`: returns the dates at which contact structure changes (see manpage of `get.dates` for details).

as.matrix `signature(x = "obkContacts",matrix.type=c("adjacency","incidence","edgelist"))`: extract an adjacency, incidence, or edge matrix from the object.

as.data.frame `signature(x = "obkContacts", ...)`: for static contacts, returns an edge list; for dynamic contacts, returns and edge list with information on the timing of contacts.

Author(s)

Original class: Simon Frost (<sdwfrost@gmail.com>)
Plot, various accessors, conversion methods: Thibaut Jombart (<tjombart@imperial.ac.uk>)

Examples

```
## THIS IS A TOY EXAMPLE ##
cf <- c("a", "b", "a", "c", "d")
ct <- c("b", "c", "c", "d", "b")
onset <- c(1, 2, 3, 4, 5)
terminus <- c(1.2, 4, 3.5, 4.1, 6)
oc.static <- new("obkContacts",cf,ct,FALSE) # static network
oc.dynamic <- new("obkContacts",cf,ct,FALSE,onset,terminus)
oc.static
oc.dynamic

plot(oc.static, main="Contact network")



## PLOTTING A DYNAMIC CONTACT NETWORK ##
## LOAD DATA
data(ToyOutbreak) # ToyOutbreak includes a dynamic contact network

## PLOT THE STATIC NETWORK
plot(ToyOutbreak@contacts)

## PLOT THE DYNAMIC NETWORK OVER THE FIRST 4 DAYS OF THE OUTBREAK
par(mfrow=c(2,2))

plot(ToyOutbreak@contacts)

plot(get.contacts(ToyOutbreak, from=0, to=2), displaylabels=TRUE,
     main="Contact network - days 1-2")

plot(get.contacts(ToyOutbreak, from=2.9, to=3.1), displaylabels=TRUE,
     main="Contact network - day 3")

plot(get.contacts(ToyOutbreak, from=3.9, to=4.1), displaylabels=TRUE,
     main="Contact network - day 4")

par(mfrow=c(1,1))

# extract matrices of adjacency, incidence, or edge list
as.matrix(ToyOutbreak@contacts)
as.matrix(ToyOutbreak@contacts, "incidence")
as.matrix(ToyOutbreak@contacts, "edgelist")

as.data.frame(ToyOutbreak@contacts)
```

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, ...)
- time-stamped samples and records (swabs, serology, accession numbers, ...)
- genetic sequences
- contact information
- contextual information
- phylogenetic trees

Usage

```
## S4 method for signature 'obkData'
initialize(.Object, individuals=NULL, records=NULL,
          dna=NULL, trees=NULL, contacts=NULL, context=NULL,
          contacts.start=NULL, contacts.end=NULL,
          contacts.duration=NULL, contacts.directed=FALSE,
          date.format=NULL, dna.individualID=NULL, dna.date=NULL,
          dna.date.format=date.format, dna.sep="_", quiet=FALSE,
          check=TRUE, individuals=NULL, records=NULL, ...)

get.nrecords(x, ...)
get.records(x, ...)
## S4 method for signature 'obkData'
get.records(x, ...)
## S4 method for signature 'obkData'
get.nrecords(x, ...)

get.ncontext(x, ...)
get.context(x, ...)
## S4 method for signature 'obkData'
get.context(x, ...)
## S4 method for signature 'obkData'
get.ncontext(x, ...)

get.ntrees(x, ...)
get.trees(x, ...)
## S4 method for signature 'obkData'
get.ntrees(x, ...)
## S4 method for signature 'obkData'
get.trees(x, ...)
```

```
## S4 method for signature 'obkData'
get.dna(x, locus=NULL, id=NULL, ...)

## S4 method for signature 'obkData'
get.ncontacts(x, from=NULL, to=NULL, ...)
## S4 method for signature 'obkData'
get.contacts(x, from=NULL, to=NULL, ...)
```

Arguments

.Object	the prototype of an obkData object, created automatically by new.
individuals	a data.frame with a mandatory column named 'individualID', providing unique identifiers for the individuals; if missing, row names are used as identifiers.
records	a list of data.frames, each of which has 2 mandatory fields, 'individualID' and 'date'. Dates can be specified as Date or characters, in which case they will be converted to dates. Most sensible formats will be detected automatically and processed. Unusual formats should be provided through the argument date.format. Each item of the list should be named according to the type of information recorded, e.g. 'swabs', 'temperature', or 'hospitalisation' (admission / discharge events).
dna	a list matrices of DNA sequences in DNAbin or character format, each component of the list being a different gene. A matrix can be provided if there is a single gene.
dna.date	a vector of collection dates for the DNA sequences; see obkSequences manpage for more information.
dna.individualID	a vector of individual from which DNA sequences where obtained; see obkSequences manpage for more information.
dna.date.format	a character string indicating the format of the date in dna.date if ambiguous; see obkSequences manpage for more information.
dna.sep	the character string used to separate fields (e.g. sequenceID/individualID/date) in sequences labels; see obkSequences manpage for more information.
contacts	a matrix of characters indicating contacts using two columns; if contacts are directed, the first column is 'from', the second is 'to'; values should match individual IDs (as returned by get.individuals(x)); if numeric values are provided, these are converted to integers and assumed to correspond to individuals returned by get.individuals(x).
context	a list of data.frames, each of which has 1 mandatory field: 'date'. Each item of the list should be named according to the type of information recorded, e.g. 'intervention', 'vaccination', 'climat' (temperature, humidity, etc.), or schools (opening/closure).
contacts.start	a vector of dates indicating the beginning of each contact.
contacts.end	a vector of dates indicating the end of each contact.

`contacts.duration`
: another way to specify contacts.end, as duration of contact in days.

`contacts.directed`
: a logical indicating if contacts are directed; defaults to FALSE.

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

`date.format`
: a character string indicating the date format (see `as.Date`); if `NULL`, date format is detected automatically, which is usually a sensible option.

`x`
: an `obkData` object.

`locus`
: an indication of the locus, either by its name, or using integers or logicals matching `get.locus`.

`id`
: an indication of the sequences, either by their names, or using integers or logicals matching `get.sequences`.

`from,to`
: dates indicating the time span to consider for active contacts (for dynamic networks only).

`...`
: arguments passed to other methods.

`quiet`
: a logical indicating whether informative messages should be displayed to the screen.

`check`
: a logical indicating whether supplementary consistency checks should be performed.

Objects from the class obkData

`obkData` objects can be created using `new("obkData", ...)`, where '...' corresponds to the arguments of the corresponding initialize method (see 'Usage' section in this page).

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.

`records`**:** a `list` of `data.frame`, each containing records of a particular type; the first two columns are 'individualID'and 'date'.

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

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

`context`**:** a `list` of `data.frame`, each member of the list containing contextual information related to the population of a particular type; the first column is 'date'.

`trees`**:** an object of the class `multiphylo` storing list of trees.

Methods

Here is a list of accessors available for obkData objects. These functions are used to retrieve the content of the object. Specific manpages can exist for the more complex functions. These are indicated by a '*' symbol next to the method's name.

show `signature(x = "obkData")`: printing of the object's contents.

head `signature(x = "obkData")`: printing of the object's contents - showing only the firt lines of each record.

tail `signature(x = "obkData")`: printing of the object's contents - showing only the last lines of each record.

summary `signature(x = "obkData")`: printing a summary of the object.

get.individuals* `signature(x = "obkData")`: returns the itendifiers of individuals in a given source of data (see manpage of `get.individuals` for details).

get.nindividuals* `signature(x = "obkData")`: returns the number of individuals in a given source of data (see manpage of `get.individuals` for details).

get.dates* `signature(x = "obkData")`: returns the dates in a given source of data (see manpage of `get.dates` for details).

get.dates* `signature(x = "obkData")`: returns the number of dates in a given source of data (see manpage of `get.dates` for details).

get.records `signature(x = "obkData")`: returns the names of existing records tables in the data.

get.nrecords `signature(x = "obkData")`: returns the number of records tables in the data.

get.locus `signature(x = "obkData")`: returns the names of the sequenced loci.

get.nlocus `signature(x = "obkData")`: returns the number of sequenced loci.

get.nsequences* `signature(x = "obkData")`: returns the number of sequences in the data; see obkSequences manpage for further details.

get.dna* `signature(x = "obkData")`: returns the sequences for a given locus (`locus` argument, required if more than one locus was sequenced)see obkSequences manpage for further details.

get.context `signature(x = "obkData")`: returns the names of existing context tables in the data.

get.ncontext `signature(x = "obkData")`: returns the number of context tables in the data.

get.trees `signature(x = "obkData")`: return a list of trees of the class `multiPhylo`, if present, and `NULL` otherwise.

get.trees `signature(x = "obkData")`: return the number of trees present in the data.

get.contacts* `signature(x = "obkData")`: returns the contacts (as `network` or `networkDynamic`); for dynamic contact networks, starting dates can be specified by the argument `from`, and end date by the argument `to`; see obkContacts manpage for further details.

get.ncontacts* `signature(x = "obkData")`: returns the number of contacts; for dynamic contact networks, starting dates can be specified by the argument `from`, and end date by the argument `to`; see obkContacts manpage for further details.

get.data* `signature(x = "obkData")`: search for a matching fields in the object and returns the corresponding values; in the absence of match, `NULL` is returned. Several values can be provided; they can be names of the slots, or any variable stored within the data.frames `samples` or `individuals`.

Author(s)

Thibaut Jombart, Simon Frost, Lulla Opatowski, Paul Birrell, Anne Cori, Marc Baguelin, Caroline Colijn

See Also

- subset.obkData to subset the data in various ways.
- plot.obkData to plot the data.

Examples

```
## LOAD DATA ##
data(ToyOutbreakRaw)
attach(ToyOutbreakRaw)

## CONSTRUCTING AN OBKDATA OBJECT ##
x <- new ("obkData", individuals=individuals, records=records,
          contacts=contacts, contacts.start=contacts.start,
          contacts.end=contacts.end, dna=dna,
          dna.individualID=dna.info$individualID,
          dna.date=dna.info$date, sample=dna.info$sample, trees=trees)

detach(ToyOutbreakRaw)


## EXAMINING THE OBJECT ##
head(x@individuals)
names(x@records)
lapply(x@records, head)
x@contacts
x@dna
x@trees


## HEAD, TAIL, SUMMARY ##
head(x)
tail(x)
summary(x)


##  ACCESSORS
get.nlocus(x)
get.locus(x)
get.nindividuals(x)
head(get.individuals(x))
get.individuals(x, data="contacts")
get.nsequences(x)
get.dna(x, locus="gene2")
get.dna(x, locus=1)
head(get.data(x, "Fever"))
head(get.data(x, "Age", showSource=TRUE))
```

```
head(get.data(x, c("Age","Sex", "infector"), showSource=TRUE))


## GRAPHICS ##
## default plot (timeline of information) ##
plot(x)
plot(x, colorBy='Sex')
plot(x, colorBy='Sex', orderBy='Sex')
plot(subset(x, 1:50), colorBy='Sex', size=4)

## plot contacts ##
plot(x, "contacts", main="Contacts")

## Not run:
## visualize data on a map ##
plot(x,'geo',location=c('lon','lat'),zoom=15,colorBy='Sex')

## plot the tree ##
plotggphy(x)
plotggphy(subset(x, 1:50), tip.color="Sex", color.pal="Set1")

## End(Not run)
```

---

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, alonside meta-information on the sequences.

An obkSequences object can be constructed from a list of matrices containing each a set of sequences of a given gene/locus. Sequences may be stored as character strings or as DNAbin objects. Information on individuals and collection dates, as well as other meta data for the sequences, can be provided.

Usage

```
get.nlocus(x, ...)
get.locus(x, ...)

get.nsequences(x, ...)
get.sequences(x, ...)

get.dna(x, ...)

## S4 method for signature 'obkSequences'
get.nsequences(x, what=c("total","bylocus"), ...)
```

```
## S4 method for signature 'obkSequences'
get.sequences(x, ...)

## S4 method for signature 'obkSequences'
get.dna(x, locus=NULL, id=NULL, ...)

## S4 method for signature 'obkSequences'
show(object)
```

Arguments

x,object	an obkSequences object.
what	a character string indicating whether numbers of sequences should be provided in total, or per locus.
locus	an indication of the locus, either by its name, or using integers or logicals matching get.locus.
id	an indication of the sequences, either by their names, or using integers or logicals matching get.sequences.
...	arguments passed to other methods.

Details

obkSequences are meant to store DNA sequences for which patient/individual IDs and collection dates are known. This information can be provided using the arguments individualID and date, or through the labels of the sequences. In that case, the expected format is:
[sequenceID][sep][individualID][sep][date]
By default, the separator is "_", so a valid name would look like:
"seq123_John Doe_2013/06/23"

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 sequence matrices in DNAbin or character format.

individualID an optional vector providing IDs of patients/individuals for each sequence; its length must match the total number of sequences; if missing, will be seeked from the labels of the sequences (see details).

date an optional vector providing collection dates each sequence; its length must match the total number of sequences; if missing, will be seeked from the labels of the sequences (see details).

... an optional list of vectors or a data.frame providing other information about each sequence; vector lengths / number of rows of the data.frame must match the total number of sequences.

date.format a character string indicating the date format (see as.Date); if NULL, date format is detected automatically.

quiet a logical indicating whether informative messages should be hidden; this does not affect warnings or error messages.

sep a character used to separate fields in the labels of the sequences (see details).

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.

meta: a data.frame where the first two columns are individualID and date, with optional further columns, and one row per sequence; rows are named after sequence labels.

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. Accessors with more than one argument 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; the argument what can be "total" (default), in which case the total number of sequences is returned, or "bylocus", in which case the number of sequences per locus is returned.

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

get.dna signature(x = "obkSequences"): returns dna sequences, which can be optionally specified by locus (argument locus) or by sequence identifier (argument id); loci can be indicated by index or by name (use get.locus to know available loci); sequence IDs must be characters.

Author(s)

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

Examples

```
## construct an obkSequences object ##

data(ToyOutbreakRaw)
attach(ToyOutbreakRaw)
x <- new("obkSequences", dna, individualID=dna.info$individualID,
    date=dna.info$date)
x
detach(ToyOutbreakRaw)

## Load ToyOutbreak, a simulated outbreak stored in a obkData object  ##
data(ToyOutbreak)
x <- ToyOutbreak
```

```
## show obkObject
##summary(x) # generates an error

## access raw content
get.dna(x)

## access data by locus
get.nlocus(x)
get.locus(x)
get.dna(x, locus=1)
get.dna(x, locus="gene2")
get.dna(x, locus=1:2)
get.nsequences(x)
get.nsequences(x, "bylocus")

get.sequences(x)

## access data by sequence ID
get.dna(x, id=c("10"))
get.dna(x, id=c("100","354"))
get.dna(x, id=c(100:105))
```

---

phylo2ggphy *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 = NULL,
  verbose = FALSE)
```

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
verbose	if TRUE additional information is provided at execution

Author(s)

Anton Camacho

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

Author(s)

Caroline Colijn

Plot obkData *Plot outbreak data*

Description

The plot method of obkData objects includes various options:

- `timeline`: plots requested information as time series; uses `plotIndividualTimeline`.
- `geo`: plots requested information as geographic distribution, on a map; uses `plotGeo`.
- `mst`: minimum-spanning tree based on genetic distances; uses `plotggMST`.
- `phylo`: plot phylogenetic trees stored in the object; uses `plotggphy`.
- `contacts`: plot contacts between individuals stored in the object; uses `plot.obkContacts`.

Usage

```
## S4 method for signature 'obkData'
plot(x, y=c("timeline","geo","mst","phylo","contacts"), ...)
```

Arguments

x	a obkData object
y	a character indicating the type of plot to be generated; can be "timeline" (default), "geo", "mst", "phylo", or "contacts".
...	further arguments to be passed to the plotting function, depending on the type of plot selected (see description above).

Author(s)

Rolf Ypma <Rolf.Ypma@rivm.nl>, Thibaut Jombart <t.jombart@imperial.ac.uk>

See Also

plotIndividualTimeline, plotGeo, plotggMST, plotggphy, plot.obkContacts.

Examples

```
data(HorseFlu)
head(HorseFlu)

## default plot: time line
plot(HorseFlu,orderBy='yardID',colorBy='yardID')
plot(HorseFlu,orderBy='yardID',colorBy='yardID',
     selection=1:30) # only the first 30

## plot contacts
data(ToyOutbreak)
plot(ToyOutbreak, "contacts")

## Not run:
## plot geographic distribution of individuals
plot(HorseFlu, "geo", location=c('lon','lat'), colorBy='sex',
     zoom=12,center='9')

## plot minimum spanning tree for first 10 individuals
get.nindividuals(HorseFlu)
plot(subset(HorseFlu, individuals=1:10), "mst")

## End(Not run)
```

---

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

Author(s)

Xavier Didelot

---

plotGeo *Function to plot cases on a map*

Description

Function to plot the geographic distribution of data in an obkData.

Usage

```
plotGeo(x, location=NULL, zoom='auto', source='google',
        colorBy=NULL, shapeBy=NULL, center=NULL, ...)
```

Arguments

x	the main obkData object
location	the name of the columns containing location data; if a vector of length 2 is provided, these are expected to be longitudes and latitudes.
zoom	a numeric indicating the level of zooming; higher number gives smaller scale (higher zoom).
source	a character string indicating the internet source from which to download maps.
colorBy	the name of the attribute to use for coloring the symbols.
shapeBy	the name of the attribute to use for shaping the symbols.
center	individualID of individual to put at the center of the map. If left empty, method will focus on the center of all points
...	further arguments passed to geom_point.

Author(s)

Original version by Rolf Ypma. Tweaks by Thibaut Jombart.

Examples

```
## Not run:
## load the obkData of equine influenza outbreak
data(HorseFlu)
x <- HorseFlu

## plot the individuals on a map
plotGeo(x,location=c('lon','lat'),zoom=8)

## color by sex
plotGeo(x,location=c('lon','lat'),zoom=8,colorBy='sex')

## zoom in on the small cluster, by centering on individual '9'
plotGeo(x,location=c('lon','lat'),colorBy="sex",zoom=14,center='9',size=4,
  alpha=(.7))

plotGeo(x,location=c('lon','lat'),colorBy="yardID", shapeBy="sex", zoom=14,center='9',size=4,
  alpha=(.7))

## another example ##
## load obkData object containing data about a simulated outbreak
data(ToyOutbreak)

## plot the individuals on a map
plotGeo(ToyOutbreak,location=c('lon','lat'), zoom=8)
plotGeo(ToyOutbreak,location=c('lon','lat'), zoom=13, colorBy='Sex', size=3)

## color by age, zooming on the first case of the outbreak: indivudal 1
plotGeo(ToyOutbreak,location=c('lon','lat'), zoom=15,
        colorBy='Age', center='1', size=5)

## End(Not run)
```

---

plotggMST *Function to plot a minimum spanning tree of the class 'obkData'*

Description

Function to plot a minimum spanning tree for an individual using the mst function from ape and positioning the nodes according to the fruchtermanreingold in sna. The size of the nodes correspond to the number of copies for that particular sequence.

Usage

```
plotggMST(x,individualID=NULL,locus=NULL)
```

Arguments

x	An object of the class "obkData"
locus	a character or integer indicating the loci to be used; if NULL, checks will be made to check that only one locus is in the object.
individualID	a character or integer to specify the individual identifier to draw the minimum spanning tree for.

Author(s)

Joseph Hughes

Examples

```
## Not run:
## load data
data(HorseFlu)
x <- HorseFlu

## plot minimum spanning tree for individual 42
plotggMST(x,individualID=42)

## another example data
data(ToyOutbreak)
x <- ToyOutbreak

## plot minimum spanning tree for gene1
plotggMST(x, locus="gene1")


## End(Not run)
```

---

plotggphy *Function to plot phylogenies using* ggplot2

Description

Function to plot phylogenies using ggplot2

Usage

```
plotggphy(x, which.tree = 1, ladderize = TRUE,
  show.tip.label = NULL, tip.label.size = 3,
  build.tip.attribute = TRUE, tip.color = NULL,
  tip.alpha = NULL, tip.shape = NULL, tip.size = NULL,
  branch.unit = NULL, tip.dates = NULL,
  guess.tip.dates.from.labels = FALSE,
  set.guess = list(prefix = "_", order = 1, from = "last"),
  axis.date.format = NULL, major.breaks = NULL,
```

```
    minor.breaks = NULL, color.palette = "Spectral",
    legend.position = "right")
```

Arguments

x	An obkData object.
which.tree	Numeric. Specify the order of the tree to be plotted. Currently the function cannot plot multiple tree.
ladderize	If TRUE, the phylogeny is ladderized
show.tip.label	Logical. If TRUE, the labels of the tip are shown; if NULL (default), labels are shown for tree sizes up to 50 tips.
tip.label.size	Numeric. Size of the tip labels.
build.tip.attribute	Logical. If TRUE, then a data frame tip.attribute is constructed by merging the data frames individuals and samples.
tip.color	Character. Can be either the name of a color (e.g. "Black") or the name of a column of tip.attribute. In the first case, all tips have the specified color. In the second case, tips are color-codded according to the specified attribute.
tip.alpha	Character (or Numeric). Can be either the value of transparency (between 0 and 1) or the name of a column of tip.attribute. In the first case, all tips have the specified transparency. In the second case, tips are transparency-codded according to the specified attribute.
tip.shape	Character (or Numeric). Can be either the value of a shape (e.g. 16 correpond to filled circles) or the name of a column of tip.attribute. In the first case, all tips have the specified shape. In the second case, tips are shape-codded according to the specified attribute.
tip.size	Character (or Numeric). Can be either the value of tip size or the name of a column of tip.attribute. In the first case, all tips have the specified size. In the second case, tips are size-codded according to the specified attribute.
branch.unit	Character. The unit of the branch can be either "year", "month", "day" or "subst". If a time unit is provided, together with use.tip.dates, then the x-axis of the phylogeny is plotted in date format using standard POSIX specification.
tip.dates	Character. If branch.unit is in unit of time, tip.dates indicates the name of the column of tip.attribute that contains the sampling dates of the tip. See also guess.tip.dates.from.labels.
guess.tip.dates.from.labels	Logical. If TRUE then tip.dates are guessed from the tip labels using the information provided by 'set.guess'.
set.guess	List. A list of three elements: prefix, order and from. For instance, if labels are formated like this: A/Shenzhen/40/2009_China_2009-06-09 then set.guess = list(prefix="_",order=3,from="first") or set.guess = list(prefix="_",order=1,from="last").
axis.date.format	Character. When x-axis is in date format, this argument allow to change the format of the tick labels. See strptime for more details.

`major.breaks` Character. Major x-axis breaks (only when x is in date format). Ex: "weeks", "15days", "months", etc.

`minor.breaks` Character. Minor x-axis breaks (only when x is in date format). Ex: "weeks", "15days", "months", etc.

`color.palette` Character. The palette for tip colors. Only palettes from the package RColorBrewer are available. See `brewer.pal` documentation for more details.

`legend.position`
Character (or numeric). The position of legends. ("left", "right", "bottom", "top", or two-element numeric vector)

Author(s)

Original versoin by Anton Camacho, modified by Thibaut Jombart.

Examples

```
## load the dataset
data(FluH1N1pdm2009)
attach(FluH1N1pdm2009)

x <- new("obkData", individuals = individuals, dna = FluH1N1pdm2009$dna,
      dna.individualID = samples$individualID, dna.date = samples$date,
      trees = FluH1N1pdm2009$trees)

detach(FluH1N1pdm2009)


## have a look at the summary
summary(x)

## first simple tree
p <- plotggphy(x, ladderize = FALSE)

## build tip attribute and use sample dates to scale the x-axis as date time
p <- plotggphy(x, branch.unit = "year")

## change x breaks and labels
p <- plotggphy(x, branch.unit = "year", major.breaks = "month",
       axis.date.format = "%b%Y")

## color-code tip location
p <- plotggphy(x, branch.unit = "year", show.tip=FALSE, tip.color = "location")

## change tip size and transparency
p <- plotggphy(x, branch.unit = "year", tip.color = "location",
               tip.size = 3, tip.alpha = 0.75)
```

plotIndividualTimeline

Plot a timeline of recorded data

Description

This function plots a timeline of recorded data per individual, using an obkData object. Colors can be used to display meta-information on individuals, while different types of records are indicated using different symbols.

Usage

```
plotIndividualTimeline(x, what="", selection=NULL, ordering=NULL,
                       orderBy=NULL, colorBy=NULL, periods=NULL,
                       plotNames=length(selection)<50, ...)
```

Arguments

x	the main obkData object
what	a character string indicating which type of record should be displayed; partial matching is permitted; 'dna' should be used to include dates of DNA sequence collections; several values can be provided as a vector; if left by default (""), all available data are use.
selection	a vector of integers indicating the subset of individuals to plot
ordering	a vector of the same length as selection, which specifies the order of individuals on the plot. Overridden by orderBy.
orderBy	a character indicating the name of the column that should be used to order the individuals. Overrides ordering.
colorBy	a character indicating the name of the column by which individuals should be coloured
periods	an Nx2 matrix of strings, giving pairs of column names for periods to be plotted
plotNames	a logiacal indicating whether the individualIDs should be shown at the y-axis
...	further arguments passed to geom_point.

Author(s)

Original version by Rolf Ypma, modified by Thibaut Jombart.

Examples

```
## simple example using ToyOutbreak
data(ToyOutbreak)
plotIndividualTimeline(ToyOutbreak)
plotIndividualTimeline(ToyOutbreak, what="DateInfected", colorBy="Sex")
plotIndividualTimeline(ToyOutbreak, what="DateInfected", colorBy="Age",
```

```
   orderBy="Sex")

plotIndividualTimeline(ToyOutbreak,selection=1:15,orderBy='lat')
plotIndividualTimeline(ToyOutbreak,selection=1:15,orderBy='lat',
   colorBy="Sex", size=4)


## example using HorseFlu
data(HorseFlu)

## plot all information, coloring by yard - messy!
plotIndividualTimeline(HorseFlu,colorBy='yardID')

## sort on yard, only DNA sequence collection
plotIndividualTimeline(HorseFlu, what="dna", orderBy='yardID',colorBy='yardID')

## just plot the first 30, vaccination dates
plotIndividualTimeline(HorseFlu, what=c("FirstVac","LastVac"),
   selection=1:30,orderBy='yardID',colorBy='yardID', size=3)
```

`read annotated trees` *Read annotated tree files in Newick or NEXUS format*

Description

These functions read annotated trees from files in Newick or NEXUS formats. Except for the annotations, these functions mimic ape's functions `read.tree` and `read.nexus`.

Annotations are ordered by edges, i.e. matching the `edge.length` slot of a `phylo` object.

Usage

```
read.annotated.tree(file="", text = NULL, tree.names = NULL, skip = 0,
                    comment.char = "#", keep.multi = FALSE, ...)
read.annotated.nexus(file, tree.names = NULL)
```

Arguments

`file`	a file name specified by either a variable of mode character, or a double-quoted string; if 'file = ""' (the default) then the tree is input on the keyboard, the entry being terminated with a blank line
`text`	alternatively, the name of a variable of mode character which contains the tree(s) in parenthetic format. By default, this is ignored (set to 'NULL', meaning that the tree is read in a file); if 'text' is not 'NULL', then the argument 'file' is ignored.

tree.names	if there are several trees to be read, a vector of mode character that gives names to the individual trees; if 'NULL' (the default), the trees are named '"tree1"', '"tree2"', ...
skip	the number of lines of the input file to skip before beginning to read data (this is passed directly to' scan()').
comment.char	a single character, the remaining of the line after this character is ignored (this is passed directly to 'scan()').
keep.multi	if 'TRUE' and 'tree.names = NULL' then single trees are returned in '"multiPhylo"' format, with any name that is present (see details). Default is 'FALSE'.
...	further arguments to be passed to 'scan()'.

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

```
## Not run:
## read annotated tree from Nexus file
## (note: do not use 'system file' for your own file!)
tre <- read.annotated.nexus(system.file("files/BEAST-expl.nex", package="OutbreakTools"))

## ladderize the tree
tre <- ladderize(tre)

## this tree has annotations
##
names(tre)
class(tre$annotations)
length(tre$annotations)

## for each edge (each edge is identified by a terminal node), we have:
tre$annotations[[1]]
names(tre$annotations[[1]])

## extract rates from annotations
rates <- unlist(sapply(tre$annotations, function(e) e$rate_median))

## plot tree, show median rates as colors
```

```
plot(tre, show.tip=FALSE, edge.col=num2col(rates, col.pal=seasun))


## End(Not run)
```

---

simuEpi *Simulate an epidemic following a SIR model*

Description

Simulate an epidemic following a SIR model, together with a transmission tree and a set of sequences consistent with the trajectory of the epidemic.

Usage

```
simuEpi(N = 1000, D = 10, beta = 0.001, nu = 0.1, L = 1000, mu =
        0.001, plot=TRUE, makePhyloTree=FALSE)
```

Arguments

N	Size of the population
D	Duration of simulation
beta	Rate of infection
nu	Rate of recovery
L	Length of genetic sequences
mu	Probability of mutation per base per transmission event
plot	logical indicating whether or not to plot the SIR trajectory over time and save it in the output.
makePhyloTree	Logical; whether to create a neighbour-joining tree from the simulated sequences.

Value

A list containing the SIR dynamics (`$dynamics`), an `obkData` of the outbreak (`$x`), and an optional `ggplot` graphic (`$plot`).

Author(s)

Original version by Xavier Didelot and Caroline Colijn. Graphics modification by Thibaut Jombart.

Examples

```
## Not run:
## Simulate an outbreak of 200 individuals over 20 time steps ##
set.seed(3)

x <- simuEpi(N = 200, D = 20, beta = 0.002, nu = 0.1, mu = 0.002)

## x is a list:
class(x)
names(x)

## x$dynamics contains demographic info
x$dynamics

## x$plot is a ggplot object
class(x$plot)
x$plot

## x$x is the obkData object
summary(x$x)
plotIndividualTimeline(x$x, colorBy="DateInfected")


## Same, with a phylogenetic tree
x <- simuEpi(N = 200, D = 20, beta = 0.002, nu = 0.1, mu = 0.002,
             plot=TRUE, makePhyloTree=TRUE)

plotggphy(x$x, ladderize=TRUE, show.tip=TRUE,
      branch.unit="year", tip.label.size=4)

## End(Not run)
```

`subset` *Subset data in "obkData" objects*

Description

`subset` is a generic function with methods for obkData, obkSequences and obkContacts objects. It can be used to subset data by specified individuals, samples, loci, sequences, or date range. Note that several subsetting criteria can be specified at the same time.

Usage

```
## S4 method for signature 'obkData'
subset(x, individuals=NULL, locus=NULL, sequences=NULL,
       date.from=NULL, date.to=NULL, date.format=NULL, ...)

## S4 method for signature 'obkSequences'
subset(x, sequences=NULL, locus=NULL, individuals=NULL,
```

```
       date.from=NULL, date.to=NULL, date.format=NULL, ...)

## S4 method for signature 'obkContacts'
subset(x, individuals=NULL, date.from=NULL,
       date.to=NULL, date.format=NULL, ...)
```

Arguments

x	an obkData object.
individuals	labels of retained individuals; if integer, numeric or logical are provided, they are assumed to indicate individuals as returned by get.individuals.
locus	labels of retained loci; if integer, numeric or logical are provided, they are assumed to indicate locus as returned by get.locus.
sequences	labels of retained loci; if integer, numeric or logical are provided, they are assumed to indicate sequences as returned by get.sequences.
date.from, date.to	the range of dates (in Date format) of retained samples.
date.format	the format for dates if in character strings; defaults to "%Y-%m-%d" (see ?as.Date).
...	currently not used.

Author(s)

Thibaut Jombart

Examples

```
## LOAD ToyOutbreak obkData object ##
data(ToyOutbreak)
x <- ToyOutbreak


## TEST VARIOUS SUBSETTING ##
## by individual
get.individuals(x)
plot(x)
plot(subset(x, individuals=1:10))
subset(x, individuals=1)
subset(x, individuals="15")

## by locus
get.locus(x)
subset(x, locus=2)

## by locus and sequences
subset(x, locus=1, sequence=1:10)
plot(subset(x, locus=1, sequence=1:10))
plot(subset(x, locus=1, sequence=1:10), "phy")
```

```
## by dates
get.dates(x)
plotIndividualTimeline(subset(x, date.to="2000-01-05"), "dna")
```

`ToyOutbreak` *Simulated outbreak dataset*

Description

This dataset is a fake dataset containing information on an outbreak amongst 418 individuals. It contains an `obkData` object called `ToyOutbreak`.

`ToyOutbreak@individuals` is a `data.frame` containing:

- the ID of each individual, stored in the row names of the `data.frame`
- `infector`, the ID of the infector of each individual
- `DateInfected`, the date at which each individual was infected
- `Sex`, the sex of each individual
- `Age`, the age of each individual
- `lat`, the latitute corresponding to each individual
- `lon`, the longitude corresponding to each individual

`ToyOutbreak@records` is a list containing one `data.frame` called `Fever` containing:

- `individualID`, the ID of each individual,
- `date`, the date at which temperature was measured in each individual,
- `temperature`, the temperature measured in each individual.

`ToyOutbreak@contacts` is an `obkContacts` object describing the contacts inferred from the first four days of the epidemic tree (it is inferred that a contact occurred between each case and his/her infector within the day preceding infection).

`ToyOutbreak@dna` is an `obkSequences` object storing simulated DNA sequences (for 2 genes, `gene1` and `gene2`).

`ToyOutbreak@tree` is a `multiphylo` object storing one phylogenetic tree derived from the concatenated genes.

Author(s)

Anne Cori, Lulla Opatowski

Examples

```
## Load data ##
data(ToyOutbreak)

## Overview of the dataset ##
summary(ToyOutbreak)

## Plotting the dynamic contact network ##
par(mfrow=c(2,2))
plot(get.contacts(ToyOutbreak),main="Contact network - days 0-3",
     displaylabels=TRUE)
plot(get.contacts(ToyOutbreak, from=0, to=1.1),
     main="Contact network - days 0-1", displaylabels=TRUE)
plot(get.contacts(ToyOutbreak, from=2, to=2.1),
     main="Contact network - day 2", displaylabels=TRUE)
plot(get.contacts(ToyOutbreak, from=3, to=3.1),
     main="Contact network - day 3", displaylabels=TRUE)

## Mapping the outbreak (by sex) ##
tryCatch(
plot(ToyOutbreak,'geo',location=c('lon','lat'),isLonLat=TRUE,zoom=13,
     colorBy='Sex'), error=function(e)
cat("\nProblem fetching the map - check your internet connection.\n"))
```

`ToyOutbreakRaw` *Raw simulated outbreak dataset*

Description

This dataset is a fake dataset containing information on an outbreak amongst 418 individuals. It contains a list called `ToyOutbreakRaw` of the following objects:

- `ToyOutbreakRaw$individuals`, a `data.frame` containing:

  - the ID of each individual, stored in the row names of the `data.frame`
  - `infector`, the ID of the infector of each individual
  - `DateInfected`, the date at which each individual was infected
  - `Sex`, the sex of each individual
  - `Age`, the age of each individual
  - `lat`, the latitute corresponding to each individual
  - `lon`, the longitude corresponding to each individual

- `ToyOutbreakRaw$records`, a list containing one `data.frame` called `Fever` containing:

  - `individualID`, the ID of each individual,
  - `date`, the date at which temperature was measured in each individual,

- temperature, the temperature measured in each individual.

- ToyOutbreakRaw$contacts, a matrix with two columns called from and to describing contacts between individuals

- ToyOutbreakRaw$contacts.start, a vector containing the start dates of each contact stored in ToyOutbreakRaw$contacts.

- ToyOutbreakRaw$contacts.end, a vector containing the end dates of each contact stored in ToyOutbreakRaw$contacts.

- ToyOutbreakRaw$dna is a list of two DNAbin matrices corresponding to two different genes.

- ToyOutbreakRaw$dna.info is a data.frame containing:

- individualID, the ID of individuals for each sequence.
- sampleID, an indicator of samples from which sequences were obtained.
- date, the date at samples were collected.

- ToyOutbreakRaw$tree is a multiphylo object storing a phylogenetic tree.

Author(s)

Anne Cori, Lulla Opatowski

Examples

```
## Load data ##
data(ToyOutbreakRaw)
attach(ToyOutbreakRaw)

## Constructing an obkData object ##
x <- new ("obkData", individuals=individuals, records=records,
          contacts=contacts, contacts.start=contacts.start,
          contacts.end=contacts.end, dna=dna,
          dna.individualID=dna.info$individualID,
          dna.date=dna.info$date, sample=dna.info$sample, trees=trees)

detach(ToyOutbreakRaw)


## Examining the object ##
summary(x)

head(x@individuals)
head(x@records)
names(x@records)
head(x@records$Fever)
x@contacts
x@dna
x@trees

## Plotting the dynamic contact network ##
par(mfrow=c(2,2))
```

```
plot(get.contacts(x),main="Contact network - days 0-3",displaylabels=TRUE)
plot(get.contacts(x, from=0, to=1.1), main="Contact network - days 0-1",
     displaylabels=TRUE)
plot(get.contacts(x, from=2, to=2.1), main="Contact network - day 2",
     displaylabels=TRUE)
plot(get.contacts(x, from=3, to=3.1), main="Contact network - day 3",
     displaylabels=TRUE)

## Mapping the outbreak (by sex) ##
tryCatch(
   plot(x,'geo',location=c('lon','lat'),
        isLonLat=TRUE,zoom=13,colorBy='Sex'),
        error=function(e)
cat("\nProblem fetching the map - check your internet connection.\n"))
```

`uniqSequences-class` *Formal class "uniqSequences"*

Description

The class `uniqSequences` is a formal (S4) class for storing unique sequences in the form of a `DNAbin` and an associated list of vectors containing the original sequenceID.

Objects from the class uniqSequences

`uniqSequences` objects can be created using `new("uniqSequences",uniqID="listOrNULL", uniqdna="DNAbin")`:

`uniqdna` an object of type DNAbin.

`uniqID` a list of vectors with the vector names corresponding to the labels of DNAbin.

Slots

`uniqSequences` contain the following slots.

`uniqdna`**:** a `DNAbin` containing the unique sequences.

`uniqID`**:** a `list` containing the unique IDs for each sequence and the names of the original sequences as a vector.

Author(s)

Joseph Hughes

Examples

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

## LOAD RAW DATA
data(HorseFlu)

## GET SEQUENCES FROM THE FIRST 3 INDIVIDUALS
## all sequences
dna <- get.dna(subset(HorseFlu, individuals=1:3))[[1]]
dna

if(require(ape)){
plot(nj(dist.dna(dna)), type="unr")
title("NJ tree - all sequences")

## only unique sequences
dna2 <- dna2uniqSequences(dna)
dna2
plot(nj(dist.dna(dna2@uniqdna)), type="unr")
title("NJ tree - only unique sequences")

}
```

Index