Package 'nbTransmission'

July 22, 2025

Title Naive Bayes Transmission Analysis

Version 1.2.0

Date 2025-05-14

Maintainer Sarah V Leavitt <sv1205@bu.edu>

Description Estimates the relative transmission probabilities between cases in an infectious disease outbreak or cluster using naive Bayes. Included are various functions to use these probabilities to estimate transmission parameters such as the generation/serial interval and reproductive number as well as finding the contribution of covariates to the probabilities and visualizing results. The ideal use is for an infectious disease dataset with metadata on the majority of cases but more informative data such as contact tracing or pathogen whole genome sequencing on only a subset of cases. For a detailed description of the methods see Leavitt et al. (2020) <doi:10.1093/ije/dyaa031>.

Depends R (>= 3.4.0)

RoxygenNote 7.3.2

Encoding UTF-8

License MIT + file LICENSE

URL https://sarahleavitt.github.io/nbTransmission/

BugReports https://github.com/sarahleavitt/nbTransmission/issues

LazyData true

Imports dplyr, rlang, caret, lubridate, poisbinom, stats, tidyr, utils, broom

Suggests ggplot2, Hmisc, igraph, knitr, methods, pheatmap, RColorBrewer, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Sarah V Leavitt [aut, cre], Anne Shapiro [aut]

Repository CRAN

Date/Publication 2025-05-14 17:40:06 UTC

Contents

`clusterInfectors` *Clusters the infectors based on their transmission probabilities*

Description

The function `clusterInfectors` uses either kernel density estimation or hierarchical clustering to cluster the infectors for each infectee. This clustering provides a way to separate out the few top possible infectors for each infectee if there is such a cluster.

Usage

```
clusterInfectors(
  df,
  indIDVar,
  pVar,
  clustMethod = c("n", "kd", "hc_absolute", "hc_relative"),
  cutoff
)
```

Arguments

`df`	The name of the dateset with transmission probabilities (column `pVar`), individual IDs (columns `<indIDVar>.1` and `<indIDVar>.2`).
`indIDVar`	The name (in quotes) of the individual ID columns (data frame `df` must have variables called `<indIDVar>.1` and `<indIDVar>.2`).
`pVar`	The name (in quotes) of the column with transmission probabilities.

clustMethod	The method used to cluster the infectors; one of "n", "kd", "hc_absolute", "hc_relative" (see details).
cutoff	The cutoff for clustering (see details).

Details

This function provides a way to find the most likely infectors for each infectee using various clustering methods indicated by the clustmethod. The methods can be one of c("n", "kd", "hc_constant", "hc_relative").

If clustMethod == "n" then this function simply assigns the top n possible infectors in the top cluster where n is defined by the value of cutoff.

If clustMethod == "kd" then kernel density estimation is used to split the infectors. The density for the probabilities for all infectors is estimated using a binwidth defined by the value of cutoff. If the density is made up of at least two separate curves (separated by a region where the density drops to 0) then the infectors with probabilities greater than the lowest 0 region are assigned to the high probability cluster. If the density of the probabilities does not drop to 0 then all infectors are assigned to the low probability cluster (indicating no real clustering).

If clustMethod == "hc_absolute" or clustMethod == "hc_relative", then hierarchical clustering with minimum distance is used to split the possible infectors into two clusters. This method functionally splits the infectors by the largest gap in their probabilities.

Then if clustMethod == "hc_absolute", those infectees where the gap between the two clusters is less than cutoff have all of their possible infectors reassigned to the low probability cluster (indicating no real clustering). If clustMethod == "hc_relative", then all infectees where the gap between the two clusters is less than cutoff times the second largest gap in probabilities are reassigned to the low probability cluster (indicating no real clustering).

Value

The original data frame (df) with a new column called cluster which is a factor variable with value 1 if the infector is in the high probability cluster or 2 if the infector is in the low probability cluster.

See Also

nbProbabilities

Examples

```
## Use the nbResults data frame included in the package which has the results
## of the nbProbabilities() function on a TB-like outbreak.

## Clustering using top n
# High probability cluster includes infectors with highest 3 probabilities
clust1 <- clusterInfectors(nbResults, indIDVar = "individualID", pVar = "pScaled",
                           clustMethod = "n", cutoff = 3)
table(clust1$cluster)

## Clustering using hierarchical clustering
```

```
# Cluster all infectees, do not force gap to be certain size
clust2 <- clusterInfectors(nbResults, indIDVar = "individualID", pVar = "pScaled",
                           clustMethod = "hc_absolute", cutoff = 0)
table(clust2$cluster)


# Absolute difference: gap between top and bottom clusters is more than 0.05
clust3 <- clusterInfectors(nbResults, indIDVar = "individualID", pVar = "pScaled",
                           clustMethod = "hc_absolute", cutoff = 0.05)
table(clust3$cluster)

# Relative difference: gap between top and bottom clusters is more than double any other gap
clust4 <- clusterInfectors(nbResults, indIDVar = "individualID", pVar = "pScaled",
                           clustMethod = "hc_relative", cutoff = 2)
table(clust4$cluster)

## Clustering using kernel density estimation
# Using a small binwidth of 0.01
clust5 <- clusterInfectors(nbResults, indIDVar = "individualID", pVar = "pScaled",
                           clustMethod = "kd", cutoff = 0.01)
table(clust5$cluster)
```

`estimateR` *Estimates the effective reproductive number*

Description

The function `estimateR` uses the relative transmission probabilities to estimate the individual-level, time-level, and average effective reproductive numbers for an outbreak.

Usage

```
estimateR(
  df,
  indIDVar,
  dateVar,
  pVar,
  timeFrame = c("days", "months", "weeks", "years"),
  rangeForAvg = NULL,
  bootSamples = 0,
  alpha = 0.05,
  progressBar = TRUE
)
```

Arguments

df	The name of the dateset with transmission probabilities (column pVar), individual IDs (columns <indIDVar>.1 and <indIDVar>.2), and the dates of observation (columns <dateVar>.1 and <dateVar>.2).
indIDVar	The name (in quotes) of the individual ID columns (data frame df must have variables called <indIDVar>.1 and <indIDVar>.2).
dateVar	The name (in quotes) of the columns with the dates that the individuals are observed (data frame df must have variables called <dateVar>.1 and <dateVar>.2) which must be date or date-time (POSIXt) objects.
pVar	The column name (in quotes) of the transmission probabilities.
timeFrame	The time frame used to calculate Rt (one of "days", "months", "weeks", "years").
rangeForAvg	A vector with the start and ending time period to be used to calculate the average effective reproductive number.
bootSamples	The number of bootstrap samples; if 0, then no confidence intervals are calculated.
alpha	The alpha level for the confidence intervals.
progressBar	A logical indicating if a progress bar should be printed (default is TRUE).

Details

The effective reproductive number is the average number of cases an infectious case will produce in a population of both susceptible and non-susceptibe individuals. The rational behind this reproductive number estimation is Wallinga and Teunis (2004) where the individual-level reproductive number is estimated by summing the relative probability that the individual infected any other individual.

If p_{ij} equals the relative probability that case i was infected by case j, then the individual-level reproductive number (R_j) is calculated by:

$$R_j = \sum_{m \neq j} p_{mj}$$

The time-level reproductive number is then estimated by averaging the individual-level reproductive numbers for all individuals observed in the time frame (can specify days, weeks, months, years).

Finally, the time-level reproductive numbers are averaged to estimate the average effective reproductive number within rangeForAvg. To get the best estimate of the average effective reproductive number, one should only consider the stable portion of the outbreak (exclude the beginning and end).

If bootSamples > 0, bootstrap confidence intervals will be estimated for both the time-level and average reproductive numbers using parametric bootstrapping.

Value

A list with five elements:

1. RiDf - a data frame with the individual-level reproductive numbers. Column names:

- <indIDVar> - the individual ID with name specified.
- <dateVar> - the date the individual was observed with name specified.
- Ri - the individual-level reproductive number.
- nInfectees - the number of possible infectees for this individual.

2. RtDf - a data frame with the time-level reproductive numbers. Column names:
   - time - the time frame corresponding to the reproductive number estimate (day for "days" and "weeks", month for "months", year for "years").
   - timeRank - the rank of the time frame.
   - Rt - the time-level reproductive number for this time frame.
   - ciLower - lower bound of confidence interval for Rt (only if bootSamples > 0).
   - ciUpper - upper bound of confidence interval for Rt (only if bootSamples > 0).
3. RtAvgDf - a data frame with the average effective reproductive. Column names:
   - RtAvg - the average time-level reproductive number between the range specified in rangeForAvg.
   - ciLower - lower bound of confidence interval for Rt (only if bootSamples > 0).
   - ciUpper - upper bound of confidence interval for Rt (only if bootSamples > 0).
4. timeFrame - a vector with the timeFrame input
5. rangeForAvg - a vector with the rangeForAvg input

References

Wallinga J, Teunis P. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*. 2004 Sep 15;160(6):509-16.

See Also

nbProbabilities estimateRi estimateRt estimateRtAvg

Examples

```
## Use the nbResults data frame included in the package which has the results
## of the nbProbabilities() function on a TB-like outbreak.

## Getting initial estimates of the reproductive number
# (without specifying rangeForAvg and without confidence intervals)
rInitial <- estimateR(nbResults, dateVar = "infectionDate",
               indIDVar = "individualID", pVar = "pScaled",
               timeFrame = "months")

## Finding the stable portion of the outbreak for rangeForAvg using plot of Rt
cut1 <- 25
cut2 <- 125

# Optional plot to determine the cutpoints above
# ggplot(data = rInitial$RtDf, aes(x = timeRank, y = Rt)) +
#    geom_point() +
#    geom_line() +
#    geom_hline(data = rInitial$RtAvgDf, aes(yintercept = RtAvg), size = 0.7) +
```

```
#    geom_vline(aes(xintercept = cut1), linetype = 2, size = 0.7) +
#    geom_vline(aes(xintercept = cut2), linetype = 2, size = 0.7)

## Finding the final reproductive number estimates with confidence intervals
# NOTE should run with bootSamples > 2.
rFinal <- estimateR(nbResults, dateVar = "infectionDate",
             indIDVar = "individualID", pVar = "pScaled",
             timeFrame = "months", rangeForAvg = c(cut1, cut2),
             bootSamples = 2, alpha = 0.05)

rFinal$RtAvgDf
```

`estimateRi` *Estimates individual-level reproductive numbers*

Description

The function `estimateRi` uses relative transmission probabilities to estimate the individual-level reproductive number.

Usage

```
estimateRi(df, indIDVar, dateVar, pVar)
```

Arguments

`df`	The name of the dateset with transmission probabilities
`indIDVar`	The variable name (in quotes) of the individual ID varaibles (data frame `df` must have variables called `<indIDVar>.1` and `<indIDVar>.2`).
`dateVar`	The variable name (in quotes) of the dates that the individuals are observed (data frame `df` must have variables called `<dateVar>.1` and `<dateVar>.2`).
`pVar`	The variable name (in quotes) of the transmission probabilities.

Details

This function is meant to be called by `estimateR` which estimates the individual-level, time-level, and average reproductive numbers, but it can also be run directly.

Value

A data frame with the individual-level reproductive numbers. Column names:

- `<indIDVar>` - the individual ID with name specified.
- `<dateVar>` - the date the individual was observed with name specified.
- `Ri` - the individual-level reproductive number.
- `nInfectees` - the number of possible infectees for this individual.

See Also

estimateR estimateRt estimateRtAvg

---

estimateRt *Estimates time-level reproductive numbers*

Description

The function estimateRt estimates the time-level effective reproductive number from individual-level reproductive numbers.

Usage

```
estimateRt(riData, dateVar, timeFrame = c("days", "weeks", "months", "years"))
```

Arguments

riData	The name of the dateset with individual-level reproductive numbers.
dateVar	The variable name (in quotes) of the dates that the individuals are observed (data frame riData must have a variable called <dateVar>).
timeFrame	The time frame used to calculate Rt (one of "days", "months", "weeks", "years").

Details

This function is meant to be called by estimateR which estimates the individual-level and time-level, and average reproductive numbers, but it can also be run directly.

Value

A data frame with the time-level reproductive numbers. Column names:

- time - the time frame corresponding to the reproductive number estimate (day for "days" and "weeks", month for "months", year for "years").
- timeRank - the rank of the time frame.
- Rt - the time-level reproductive number for this time frame.

See Also

estimateR estimateRi estimateRtAvg

`estimateRtAvg` *Estimates the average effective reproductive number*

Description

Averages the time-level reproductive numbers within a certain range to estimate the overall reproductive number for an oubreak.

Usage

```
estimateRtAvg(rtData, rangeForAvg = NULL)
```

Arguments

`rtData`	The name of the dateset with time-level reproductive numbers.
`rangeForAvg`	A vector with the start and ending time period to be used to calculate the average effective reproductive number.

Details

This function is meant to be called by `estimateR` which estimates the individual-level and time-level, and average reproductive numbers, but it can also be run directly.

Value

A data frame with the average effective reproductive. Column names:

- `RtAvg` - the average time-level reproductive number between the range specified in `rangeForAvg`.

See Also

`estimateR` `estimateRi` `estimateRt`

`estimateSI` *Estimates the generation/serial interval distribution*

Description

The function `estimateSI` uses the relative transmission probabilities to estimate the generation/serial interval distribution assuming a gamma distribution. It uses the PEM algorithm developed by Hens et al. 2012 extending their method to include restricting analysis to the top cluster of possible infectors.

Usage

```
estimateSI(
  df,
  indIDVar,
  timeDiffVar,
  pVar,
  clustMethod = c("none", "n", "kd", "hc_absolute", "hc_relative"),
  cutoffs = NULL,
  initialPars,
  shift = 0,
  epsilon = 1e-05,
  bootSamples = 0,
  alpha = 0.05,
  progressBar = TRUE
)
```

Arguments

df	The name of the dateset with transmission probabilities (column pVar), individual IDs (columns <indIDVar>.1 and <indIDVar>.2), and difference in time between the pair of cases (column timeDiffVar)
indIDVar	The name (in quotes) of the individual ID columns (data frame df must have variables called <indIDVar>.1 and <indIDVar>.2).
timeDiffVar	The name (in quotes) of the column with the difference in time between infection (generation interval) or symptom onset (serial interval) for the pair of cases. The units of this variable (hours, days, years) defines the units of the resulting distribution.
pVar	The column name (in quotes) of the transmission probabilities.
clustMethod	The method used to cluster the infectors; one of "none", "n", "kd", "hc_absolute", "hc_relative" where "none" or not specifying a value means use all pairs with no clustering (see clusterInfectors for detials on clustering methods).
cutoffs	A vector of cutoffs for clustering (see clusterInfectors). If more than one cutoff is provided, a pooled estimate will also be provided.
initialPars	A vector of length two with the shape and scale to initialize the gamma distribution parameters.
shift	A value in the same units as timeDiffVar that the gamma distribution should be shifted. The default value of 0 is an unmodifed gamma distribution.
epsilon	The difference between successive estimates of the shape and scale parameters that indicates convergence.
bootSamples	The number of bootstrap samples; if 0, then no confidence intervals are calculated.
alpha	The alpha level for the confidence intervals.
progressBar	A logical indicating if a progress bar should be printed (default is TRUE).

Details

The PEM algorithm uses the prior probability that each pair is connected by direct transmission to estimate the generation/serial interval using estimation maximization. This code will provide an estimate of the generation interval if `timeDiffVar` represents the difference in infection dates and the serial interval if it represents the difference in symptom onset dates. The current generation/serial interval distribution parameters are used to update the probabilities which are then used to update the generation/serial interval distribution parameters. The process continues until the parameters converge (indicated by a change of less than `epsilon`) between iterations. *Note: time difference between pairs should not be used to estimate the probabilities*

This function acts as a wrapper around `performPEM` which integrates estimation of the generation/serial interval distribution with clustering the infectors and calculates derived parameters (mean, median, sd) of the distribution. Generally, this function should be called instead of `performPEM` directly.

All pairs of cases can be used in the estimation process by setting `clustMethod = "none"`. However, if the probabilities are from a algorithm such as `nbProbabilities`, then it is recommeneded to use a clustering method and only include the top cluster of infectors for infectees which have such a cluster. This can be specified by using the `clustMethod` and `cutoff` arguments which are passed into `clusterInfectors`. See the details of this function for a description of the different clustering methods.

The method can be performed with any generation/serial interval distribution, but this version of this function assumes that the generation/serial interval has a gamma distribution. The function does allow for a shifted gamma distribution. The `shift` argument defines how much the gamma distribution should be shifted. Any observed generation/serial intervals that are less than this shift will have probability 0. This parameter should be used if there is a clinical lower bound for the possible generation/serial interval. If this argument is not specified then an unmodified gamma function is used. The units of the estimated gamma distribution will be defined by the units of the provided `<timeDiffVar>` column. The value of the `shift` should be in the same units.

The algorithm requires initial parameters which should be specified as a vector: `c(<shape>, <scale>)`. These parameters should result in a gamma distribution that is on the desired scale, set by the `<timeDiffVar>` column.

If `bootSamples > 0`, bootstrap confidence intervals will be estimated for both the shape and scale parameters as well as the mean, median, and mode of the distribution using cluster bootstrapping.

Value

A data frame with one row and the following columns:

- `nIndividuals` - the number of infectees who have intervals included in the estimate.
- `pCluster` - the proportion of cases who have intervals included in the estimate.
- `nInfectors` - the average number of infectors in the top cluster.
- `shape` - the shape of the estimated gamma distribution for the interval
- `scale` - the scale of the estimated gamma distribution for the interval
- `meanSI` - the mean of the estimated gamma distribution for the interval (`shape * scale + shift`)
- `medianSI` - the median of the estimated gamma distribution for the interval (`qgamma(0.5, shape, scale) + shift`))

- sdSI - the standard deviation of the estimated gamma distribution for the interval (shape * scale ^ 2)

If bootSamples > 0, then the data frame also includes the following columns:

- shapeCILB and shapeCIUB - lower bound and upper bounds of the bootstrap confidence interval for the shape parameter
- scaleCILB and scaleCIUB - lower bound and upper bounds of the bootstrap confidence interval for the scale parameter
- meanCILB and meanCIUB - lower bound and upper bounds of the bootstrap confidence interval for the mean of the interval distribution
- medianCILB and medianCIUB - lower bound and upper bounds of the bootstrap confidence interval for the median of the interval distribution
- sdCILB and sdCIUB - lower bound and upper bounds of the bootstrap confidence interval for the sd of the interval distribution

References

Hens N, Calatayud L, Kurkela S, Tamme T, Wallinga J. Robust reconstruction and analysis of outbreak data: influenza A (H1N1) v transmission in a school-based population. *American Journal of Epidemiology*. 2012 Jul 12;176(3):196-203.

See Also

nbProbabilities clusterInfectors performPEM

Examples

```
## First, run the algorithm without including time as a covariate.
orderedPair <- pairData[pairData$infectionDiffY > 0, ]

## Create a variable called snpClose that will define probable links
# (<3 SNPs) and nonlinks (>12 SNPs) all pairs with between 2-12 SNPs
# will not be used to train.
orderedPair$snpClose <- ifelse(orderedPair$snpDist < 3, TRUE,
                        ifelse(orderedPair$snpDist > 12, FALSE, NA))
table(orderedPair$snpClose)

## Running the algorithm
# NOTE should run with nReps > 1.
resGen <- nbProbabilities(orderedPair = orderedPair,
                          indIDVar = "individualID",
                          pairIDVar = "pairID",
                          goldStdVar = "snpClose",
                          covariates = c("Z1", "Z2", "Z3", "Z4"),
                          label = "SNPs", l = 1,
                          n = 10, m = 1, nReps = 1)

## Merging the probabilities back with the pair-level data
nbResultsNoT <- merge(resGen[[1]], orderedPair, by = "pairID", all = TRUE)
```

```
## Estimating the serial interval

# Using hierarchical clustering with a 0.05 absolute difference cutoff
estimateSI(nbResultsNoT, indIDVar = "individualID",
             timeDiffVar = "infectionDiffY", pVar = "pScaled",
             clustMethod = "hc_absolute", cutoff = 0.05, initialPars = c(2, 2))


# Using all pairs
estimateSI(nbResultsNoT, indIDVar = "individualID",
              timeDiffVar = "infectionDiffY", pVar = "pScaled",
              clustMethod = "none", initialPars = c(2, 2))


# # Using a shifted gamma distribution:
# # not allowing serial intervals of less than 3 months (0.25 years)
estimateSI(nbResultsNoT, indIDVar = "individualID",
              timeDiffVar = "infectionDiffY", pVar = "pScaled",
              clustMethod = "hc_absolute", cutoff = 0.05,
              initialPars = c(2, 2), shift = 0.25)


# # Using multiple cutoffs
estimateSI(nbResultsNoT, indIDVar = "individualID",
              timeDiffVar = "infectionDiffY", pVar = "pScaled",
             clustMethod = "hc_absolute", cutoff = c(0.025, 0.05), initialPars = c(2, 2))



## Adding confidence intervals
# NOTE should run with bootSamples > 2.
estimateSI(nbResultsNoT, indIDVar = "individualID",
             timeDiffVar = "infectionDiffY", pVar = "pScaled",
             clustMethod = "hc_absolute", cutoff = 0.05,
             initialPars = c(2, 2), shift = 0.25, bootSamples = 2)
```

`indData` *Individual-level simulated outbreak dataset*

Description

A dataset of an outbreak of 100 individuals starting from one case. It was simulated with using the package TransPhylo with parameters resembling TB:

- average reproductive number = 1.2
- generation interval = gamma(1.2, 2) years
- outbreak duration = 14 years
- mutation rate 0.5 snps/genome/year

Usage

```
indData
```

Format

A data frame with 100 rows and 8 variables:

individualID An individual-level id for each case.

infector The individualID of the true infector.

infectionDate The date and time of infection.

sampleDate The date and time of sampling.

X1 Covariate with 2 values: a, b (e.g. sex).

X2 Covariate with 4 values: a, b, c, d (e.g. nationality).

X3 Covariate with 2 values: a, b (e.g. homelessness).

X4 Covariate with 10 values: a-j (e.g. county of residence).

References

Didelot X, Fraser C, Gardy J, Colijn C. Genomic Infectious Disease Epidemiology in Partially Sampled and Ongoing Outbreaks. Mol Biol Evol. 2017;34(4):997-1007.

---

indToPair *Transforms a dataset of individuals to a dataset of pairs*

Description

The function `indToPair` takes a dataset of observations (such as individuals in an infectious disease outbreak) and transforms it into a dataset of pairs.

Usage

```
indToPair(
  indData,
  indIDVar,
  separator = "_",
  dateVar = NULL,
  units = c("mins", "hours", "days", "weeks"),
  ordered = FALSE
)
```

Arguments

indData	An individual-level dataframe.
indIDVar	The name (in quotes) of the column with the individual ID.
separator	The character to be used to separate the individual IDs when creating the pairID.
dateVar	The name (in quotes) of the column with the dates that the individuals are observed (optional unless ordered = TRUE). This column must be a date or datetime (POSIXt) object. If supplied, the time difference between individuals will be calculated in the units specified.
units	The units for the time difference, only necessary if dateVar is supplied. Must be one of "mins", "hours", "days", "weeks".
ordered	A logical indicating if a set of ordered pairs should be returned (<dateVar>.1 before <dateVar>.2 or <dateVar>.1 = <dateVar>.2). If FALSE a dataframe of all pairs will be returned

Details

The function requires an id column: indIDVar to identify the individual observations. The resulting pair-level dataframe will have a pairID column which combines the individual IDs for that pair.

The function can either output all possible pairs (ordered = FALSE) or only ordered pairs (ordered = TRUE) where the ordered is determined by a date variable (dateVar). If orded = TRUE, then dateVar must be provided and if ordered = FALSE, it is optional. In both cases, if dateVar is provided, the output will include the time difference between the individuals in the pair in the units specified ("mins", "hours", "days", "weeks").

Value

A dataframe of either all possible pairs of individuals (ordered = FALSE) or ordered pairs of individuals (ordered = TRUE). The dataframe will have all of the original variables with suffixes ".1" and ".2" corresponding to the original values of <indIDVar>.1 and <indIDVar>.2.

Added to the dataframe will be a column called pairID which is <indIDVar>.1 and <indIDVar>.2 separated by separator.

If dateVar is provided the dataframe will also include variables <dateVar>.Diff giving the difference of time of dateVar for <indIDVar>.1 and <indIDVar>.2 in the units specified

Examples

```
## Create a dataset of all pairs with no date variable
pairU <- indToPair(indData = indData, indIDVar = "individualID")

## Create a dataset of all pairs with a date variable
pairUD <- indToPair(indData = indData, indIDVar = "individualID",
                      dateVar = "infectionDate", units = "days")

## Create a dataset of ordered pairs
pairO <- indToPair(indData = indData, indIDVar = "individualID",
                     dateVar = "infectionDate", units = "days", ordered = TRUE)
```

nbHeatmap *Plots a heatmap of the relative transmission probabilities*

Description

The function nbHeatmap plots a heatmap of the transmission probabilities. The rows are the possible infectors and the columns are the possible infectees both ordered by <dateVar>. The darker the square the higher the probability that the pair represented by that square is a transmission link. If a cluster method is specified using clustMethod and cutoff, then stars will be drawn in the squares of the infectors in the top cluster.

Usage

```
nbHeatmap(
  df,
  indIDVar,
  dateVar,
  pVar,
  clustMethod = c("none", "n", "kd", "hc_absolute", "hc_relative"),
  cutoff = NA,
  blackAndWhite = FALSE,
  probBreaks = c(-0.01, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1)
)
```

Arguments

df	The name of the dateset with transmission probabilities (column pVar), individual IDs (columns <indIDVar>.1 and <indIDVar>.2), and the dates of observation (columns <dateVar>.1 and <dateVar>.2).
indIDVar	The name (in quotes) of the individual ID columns (data frame df must have variables called <indIDVar>.1 and <indIDVar>.2).
dateVar	The name (in quotes) of the columns with the dates that the individuals are observed (data frame df must have variables called <dateVar>.1 and <dateVar>.2).
pVar	The name (in quotes) of the column with transmission probabilities.
clustMethod	The method used to cluster the infectors; one of "none", "n", "kd", "hc_absolute", "hc_relative" where "none" or not specifying a value means use all pairs with no clustering (see clusterInfectors for detials on clustering methods).
cutoff	The cutoff for clustering (see clusterInfectors).
blackAndWhite	A logical. If TRUE, then the squares are colored in greyscale, if FALSE, then the squares are colored with shades of blue.
probBreaks	A numeric vector containing between 3 and 10 elements specifying the boundaries used to classify the probabilities and color the squares. The first element should be less than 0 and the last should be 1.

Details

Users have the option of specifying how the probabilities should be grouped into different color shades through the argument probBreaks. The probabilities are split into groups by using probBreaks as the breaks argument in cut with the default options. The length of the vector should be between 3 and 10 and the first element should be less than 0 and the last 1 so that all probabilities are guaranteed to be classified. The colors are defined with the code brewer.pal(length(probBreaks) - 1, "Blues") (where "Blues" is replaced by "Greys" if blackAndWhite is set to TRUE).

NOTE: This plot will take long to run and may not look good with larger outbreaks (>200 individuals)

See Also

nbProbabilities clusterInfectors

Examples

```
## Heatmap with no clustering in color with the default probability breaks
par(mar = c(0, 0, 1, 0))
nbHeatmap(nbResults, indIDVar = "individualID", dateVar = "infectionDate",
pVar = "pScaled", clustMethod = "none")
dev.off()


## Adding stars for the top cluster, in black and white, changing the probability breaks
par(mar = c(0, 0, 1, 0))
nbHeatmap(nbResults, indIDVar = "individualID", dateVar = "infectionDate",
          pVar = "pScaled", clustMethod = "hc_absolute", cutoff = 0.05,
          blackAndWhite = TRUE, probBreaks = c(-0.01, 0.01, 0.1, 0.25, 0.5, 1))
dev.off()
```

---

nbNetwork *Plots a network of the relative transmission probabilities*

Description

The function nNetwork plots a network of the transmission probabilities. The nodes are the individuals and the edges represent possible transmission pairs. The darker the edge, the higher the probability that the pair is a transmission link. If a cluster method is specified using clustMethod and cutoff, only edges that are in the high probability cluster of infectors will be drawn.

Usage

```
nbNetwork(
  df,
  indIDVar,
  dateVar,
  pVar,
  clustMethod = c("none", "n", "kd", "hc_absolute", "hc_relative"),
  cutoff = NA,
  blackAndWhite = FALSE,
  probBreaks = c(-0.01, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1)
)
```

Arguments

df	The name of the dateset with transmission probabilities (column pVar), individual IDs (columns <indIDVar>.1 and <indIDVar>.2), and the dates of observation (columns <dateVar>.1 and <dateVar>.2).
indIDVar	The name (in quotes) of the individual ID columns (data frame df must have variables called <indIDVar>.1 and <indIDVar>.2).
dateVar	The name (in quotes) of the columns with the dates that the individuals are observed (data frame df must have variables called <dateVar>.1 and <dateVar>.2).
pVar	The name (in quotes) of the column with transmission probabilities.
clustMethod	The method used to cluster the infectors; one of "none", "n", "kd", "hc_absolute", "hc_relative" where "none" or not specifying a value means use all pairs with no clustering (see clusterInfectors for detials on clustering methods).
cutoff	The cutoff for clustering (see clusterInfectors).
blackAndWhite	A logical. If TRUE, then the edges are colored in greyscale, if FALSE, then the edges are colored with shades of blue.
probBreaks	A numeric vector containing between 3 and 10 elements specifying the boundaries used to classify the probabilities and color the edges. The first element should be less than 0 and the last should be 1.

Details

Users have the option of specifying how the probabilities should be grouped into different color shades through the argument probBreaks. The probabilities are split into groups by using probBreaks as the breaks argument in cut with the default options. The length of the vector should be between 3 and 10 and the first element should be less than 0 and the last 1 so that all probabilities are guaranteed to be classified. The colors are defined with the code brewer.pal(length(probBreaks) - 1, "Blues") (where "Blues" is replaced by "Greys" if blackAndWhite is set to TRUE).

See Also

nbProbabilities clusterInfectors

Examples

```
## Network of all pairs in color with the default probability breaks
par(mar = c(0, 0, 0.2, 0))
nbNetwork(nbResults, indIDVar = "individualID", dateVar = "infectionDate",
pVar = "pScaled", clustMethod = "none")
dev.off()


## Network of just the top cluster of infectors, black and white, changing the probability breaks
par(mar = c(0, 0, 0.2, 0))
nbNetwork(nbResults, indIDVar = "individualID", dateVar = "infectionDate",
          pVar = "pScaled", clustMethod = "hc_absolute", cutoff = 0.05,
          blackAndWhite = TRUE, probBreaks = c(-0.01, 0.01, 0.1, 0.25, 0.5, 1))
dev.off()
```

---

nbProbabilities *Estimates relative transmission probabilities*

Description

The function nbProbabilities uses naive Bayes and an interative estimation procedure to estimate relative transmission probabilities

Usage

```
nbProbabilities(
  orderedPair,
  indIDVar,
  pairIDVar,
  goldStdVar,
  covariates,
  label = "",
  l = 1,
  n = 10,
  m = 1,
  nReps = 10,
  orType = "univariate",
  nBS = 100,
  pSampled = 1,
  progressBar = TRUE
)
```

Arguments

orderedPair	The name of the ordered pair-level dataset with the covariates.
indIDVar	The name (in quotes) of the column with the individual ID. (data frame orderedPair must have columns called <indIDVar>.1 and <indIDVar>.2).
pairIDVar	The name (in quotes) of the column with the unique pair ID variable.
goldStdVar	The name (in quotes) of the column with a logical vector defining training links/non-links
covariates	A character vector containing the covariate column names (in quotes). All covariates need to be categorical factor variables.
label	An optional label string for the run.
l	Laplace smoothing parameter that is added to each cell.
n	The number of folds for nxm cross validation (should be at least 10).
m	The number of times to create n folds in nxm cross validation.
nReps	The number of times to randomly select the "true" infector (should be at least 10).
orType	Takes value "univariate" or "adjusted". "univariate" produces contingency table odds ratios and "adjusted" produces adjusted odds ratios from a bootstrapped multivariable logistic regression.
nBS	Number of bootstrap samples to run in each cross-validation fold/iteration (default is 100). Only relevant when orType = "adjusted".
pSampled	Proportion of unlinked cases to include in bootstrap sample (default is 1, i.e.a true bootstrap). Only relevant when orType = "adjusted".
progressBar	A logical indicating if a progress bar should be printed (default is TRUE).

Details

This algorithm takes a dataset of ordered possible infector-infectee pairs in an infectious disease outbreak or cluster and estimates the relative probability the cases are linked by direct transmission using a classification technique called naive Bayes (NB). NB is a simple machine learning algorithm that uses Bayes rule to estimate the probability of an outcome in a prediction dataset given a set of covariates from the observed frequencies in a training dataset.

The input dataset - orderedPair - should represent ordered pairs of cases (where the potential infector was observed before the infectee) and have a unique identifier for each pair (pairIDVar) as well as the individual ids that are included in the pair (<indIDVar>.1 and <indIDVar>.2). If cases are concurrent (meaning the order cannot determined) both orders can be included.

A subset of pairs should also have pathogen WGS, contact investigation, or some other 'gold standard' defined by goldStdVar which should be a logical vector with TRUE indicating links, FALSE nonlinks, and NA if the pair cannot be used to train (does not have the information or is indeterminate). These pairs will be used to a training dataset of probable links and non/links. The covariates can be any categorical variables and could represent spatial, clinical, demographic, and temporal characteristics of the case pair.

Because the outcomes in the training set represent probable and not certain transmission events and a given case could have mulitple probable infectors, the algorithm uses an iterative estimation

procedure. This procedure randomly chooses one link of all of the possible links to include in the training dataset nReps times, and then uses mxn cross prediction to give all pairs a turn in the prediction dataset.

The output of this function is a list of two dataframes: one with the estimates of the transmission probabilities (probabilities) and the other with the contribution of the covariates to the probabilities in the form of odds ratios (estimates). The 95% confidence intervals reported for these odds ratios use Rubin's Rules, a technique developed for multiple imputation, to pool the error across all iterations.

This function generates odds ratios describing the associations between covariates in the training data and outcome defined in the gold standard variable (goldStdVar) argument. Unadjusted odds ratios are the default. These odds ratios are produced using contingency table methods. Adjusted odds ratios are calculated via bootstrapped logistic regression to produce non-parametric standard errors. The bootstrap is controlled by parameters nBS, the number of bootstrap samples to run, and pSampled, the proportion of unlinked cases to include in the bootstrap sample. pSampled is recommended only for large datasets in which it is computationally unfeasible to run a full bootstrap. Sensitivity analyses should be run to determine an adequate value for pSampled.

Value

List containing two data frames:

1. probabilities - a data frame of transmission probabilities. Column names:
   - label - the optional label of the run.
   - <pairIDVar> - the pair ID with the name specified.
   - pAvg - the mean transmission probability for the pair over all iterations.
   - pSD - the standard deviation of the transmission probability for the pair over all iterations.
   - pScaled - the mean relative transmission probability for the pair over. all iterations: pAvg scaled so that the probabilities for all infectors per infectee add to 1.
   - pRank - the rank of the probability of the the pair out of all pairs for that infectee (in case of ties all values have the minimum rank of the group).
   - nEstimates - the number of probability estimates that contributed to pAvg. This represents the number of prediction datasets this pair was included in over the nxm cross prediction repeated nReps times.
2. estimates - a data frame with the contribution of covariates. Column names:
   - label - the optional label of the run
   - level - the covariate name and level
   - nIter - the number of iterations included in the estimates: n*m*nReps
   - logorMean - the mean value of the log odds ratio across iterations
   - logorSE - the standard error of the log odds ratio across iterations
   - logorCILB - the lower bound of the 95 across iterations
   - logorCIUB - the upper bound of the 95 across iterations

References

Barnard J. and Rubin D. Small-Sample Degrees of Freedom with Multiple Imputation *Biometrika*. 1999 Dec;86(4):948-55.

Examples

```
## Use the pairData dataset which represents a TB-like outbreak
# First create a dataset of ordered pairs
orderedPair <- pairData[pairData$infectionDiffY >= 0, ]

## Create a variable called snpClose that will define probable links
# (<3 SNPs) and nonlinks (>12 SNPs) all pairs with between 2-12 SNPs
# will not be used to train.
orderedPair$snpClose <- ifelse(orderedPair$snpDist < 3, TRUE,
                        ifelse(orderedPair$snpDist > 12, FALSE, NA))
table(orderedPair$snpClose)

## Running the algorithm
#NOTE should run with nReps > 1.
resGen <- nbProbabilities(orderedPair = orderedPair,
                          indIDVar = "individualID",
                          pairIDVar = "pairID",
                          goldStdVar = "snpClose",
                          covariates = c("Z1", "Z2", "Z3", "Z4", "timeCat"),
                          label = "SNPs", l = 1,
                          n = 10, m = 1, nReps = 1)

## Merging the probabilities back with the pair-level data
nbResults <- merge(resGen[[1]], orderedPair, by = "pairID", all = TRUE)
```

---

nbResults *Dataset with results of* nbProbabilities

Description

A ordered dataset created from pairData of the outbreak of 100 individuals including the relative transmission probabilities for each pair estimated using the function nbProbabilities. The code to recreate this dataset from pairData is shown below.

Usage

```
nbResults
```

Format

A data frame with 9900 rows and 24 variables:

pairID A pair-level ID variable (the individual IDs separated by an '_').

label The label for the run, here "SNPs".

pAvg The mean transmission probability for the pair over all runs.

pSD The standard deviation of the transmission probability for the pair over all runs.

pScaled The mean relative transmission probability for the pair over all runs: pAvg scaled so that the probabilities for all infectors per infectee add to 1.

pRank The rank of the probability of the the pair out of all pairs for that infectee (in case of ties all values have the minimum rank of the group).

nSamples The number of probability estimates that contributed to pAvg. This represents the number of prediction datasets this pair was included in over the 10x1 cross prediction repeated 50 times.

individualID.1 The ID of the potential "infector".

individualID.2 The ID of the potential "infectee".

transmission Did individual.1 truly infect individual.2?

snpDist The number of SNPs between the individuals.

infectionDate.1 The date and time of infection of individualID.1.

infectionDate.2 The date and time of infection of individualID.2.

sampleDate.1 The date and time of sampling of individualID.1.

sampleDate.2 The date and time of sampling of individualID.2.

sampleDiff The number of days between sampleDate.1 and sampleDate.2.

infectionDiff The number of days between infectionDate.1 and infectionDate.2.

infectionDiffY The number of years between infectionDate.1 and infectionDate.2.

timeCat A categorical representation of infectionDiff: <1y, 1-2y, 2-3y, 3-4y, 4-5y, >5y.

Z1 Pair-level covariate derived from X1: 1 if match, 0 if not match.

Z2 Pair-level covariate derived from X2: 1 if match, 0 if not match.

Z3 Pair-level covariate derived from X3: 1 if a-a, 2 if b-b, 3 if a-b, 4 if b-a.

Z4 Pair-level covariate derived from X4: 1 if match, 2 if adjacent, 2 otherwise.

snpClose Logical value indicating if a pair is a probable link. TRUE if the pair has fewer than 3 SNPs, FALSE if the pair has more than 12 SNPs, NA otherwise

Examples

```
# ## NOT RUN ##
# ## This is the code used to create this dataset ##
# orderedPair <- pairData[pairData$infectionDiff > 0, ]
# orderedPair$snpClose <- ifelse(orderedPair$snpDist < 3, TRUE,
#                          ifelse(orderedPair$snpDist > 12, FALSE, NA))
# set.seed(0)
# covariates = c("Z1", "Z2", "Z3", "Z4", "timeCat")
# resGen <- nbProbabilities(orderedPair = orderedPair,
#                               indIDVar = "individualID",
#                               pairIDVar = "pairID",
#                               goldStdVar = "snpClose",
#                               covariates = covariates,
#                               label = "SNPs", l = 1,
#                               n = 10, m = 1, nReps = 50)
# nbResults <- merge(resGen[[1]], orderedPair, by = "pairID", all = TRUE)
```

pairData *Pair-level simulated outbreak dataset*

Description

A dataset of all pairs (unordered) of the outbreak of 100 individuals described in indData with SNP distance between each pair. Genomes were simulated using the package phangorn from the phylogentic tree created during the outbreak simulation.

Usage

```
pairData
```

Format

A data frame with 9900 rows and 17 variables:

pairID A pair-level ID variable (the individual IDs separated by an '_').

individualID.1 The ID of the potential "infector".

individualID.2 The ID of the potential "infectee".

transmission Did individual.1 infect individual.2.

snpDist What is the number of SNPs between the individuals.

infectionDate.1 The date and time of infection of individualID.1.

infectionDate.2 The date and time of infection of individualID.2.

sampleDate.1 The date and time of sampling of individualID.1.

sampleDate.2 The date and time of sampling of individualID.2.

sampleDiff The number of days between sampleDate.1 and sampleDate.2.

infectionDiff The number of days between infectionDate.1 and infectionDate.2.

infectionDiffY The number of years between infectionDate.1 and infectionDate.2.

timeCat A categorical representation of infectionDiff: <1y, 1-2y, 2-3y, 3-4y, 4-5y, >5y.

Z1 Pair-level covariate derived from X1: 1 if match, 0 if not match.

Z2 Pair-level covariate derived from X2: 1 if match, 0 if not match.

Z3 Pair-level covariate derived from X3: 1 if a-a, 2 if b-b, 3 if a-b, 4 if b-a.

Z4 Pair-level covariate derived from X4: 1 if match, 2 if adjacent, 2 otherwise.

References

Schliep KP. phangorn: Phylogenetic analysis in R. Bioinformatics. 2011;27(4):592-3.

performNB *Performs naive bayes classification*

Description

The function performNB calculates the posterior probabilities of a dichotomous class variable given a set of covariates using Bayes rule and either a univariate (default, orType = "univariate" odds ratio or a bootstrapped adjusted odds ratio via logistic regression.

Usage

```
performNB(
  training,
  prediction,
  obsIDVar,
  goldStdVar,
  covariates,
  l = 1,
  orType = "univariate",
  nBS = 100,
  pSampled = 1
)
```

Arguments

training	The training dataset name.
prediction	The prediction dataset name.
obsIDVar	The variable name (in quotes) of the observation ID variable.
goldStdVar	The variable name (in quotes) of the outcome in the training dataset (needs to be a logical variable with value TRUE for observations with the outcome of interest.)
covariates	A character vector containing the covariate variable names. All covariates need to be categorical factor variables.
l	Laplace smoothing parameter that is added to each cell (a value of 0 indicates no smoothing).
orType	Takes value "univariate" or "adjusted". "univariate" produces contingency table odds ratios and "adjusted" produces adjusted odds ratios from a bootstrapped multivariable logistic regression.
nBS	Number of bootstrap samples to run in each cross-validation fold/iteration (default is 100). Only relevant when orType = "adjusted".
pSampled	Proportion of unlinked cases to include in bootstrap sample (default is 1, i.e.a true bootstrap). Only relevant when orType = "adjusted".

Details

The main purpose of this function is to be used by nbProbabilities to estimate the relative transmission probability between individuals in an infectious disease outbreak. However, it can be used more generally to estimate the probability of any dichotomous outcome given a set of categorical covariates and adjusted odds ratios of such dichotomous outcome.

This function also generates odds ratios describing the associations between covariates in the training data and outcome defined in the gold standard variable (goldStdVar) argument. Unadjusted odds ratios are the default. These odds ratios are produced using contingency table methods. Adjusted odds ratios are calculated via bootstrapped logistic regression to produce non-parametric standard errors. The bootstrap is controlled by parameters nBS, the number of bootstrap samples to run, and pSampled, the proportion of unlinked cases to include in the bootstrap sample. pSampled is recommended only for large datasets in which it is computationally unfeasible to run a full bootstrap. Sensitivity analyses should be run to determine an adequate value for pSampled.

The function needs a training dataset with the outcome variable (goldStdVar) which is TRUE for those who have the value of interest and FALSE for those who do not. The probability of having the outcome (<goldStdVar> = TRUE) is predicted in the prediction dataset.

Value

List containing two dataframes:

1. probabilities - a dataframe combining training and prediction with predictied probabilities for the prediction dataframe. Column names:
   - <obsIDVar> - the observation ID with the name specified
   - p - the probability that <goldStdVar> = TRUE for observations in the prediction dataset.
2. estimates - a dataframe with the effect estimates derived from the training dataset. Column names:
   - level - the covariate name and level
   - est - the log odds ratio for this covariate and level
   - se - the standard error of the log odds ratio

See Also

nbProbabilities

Examples

```
## Use iris dataset and predict if a flower is of the specices "virginica".

data(iris)
irisNew <- iris
## Creating an id variable
irisNew$id <- seq(1:nrow(irisNew))
## Creating logical variable indicating if the flower is of the species virginica
irisNew$spVirginica <- irisNew$Species == "virginica"

## Creating categorical/factor versions of the covariates
irisNew$Sepal.Length.Cat <- factor(cut(irisNew$Sepal.Length, c(0, 5, 6, 7, Inf)),
```

```
                                labels = c("<=5.0", "5.1-6.0", "6.1-7.0", "7.1+"))

irisNew$Sepal.Width.Cat <- factor(cut(irisNew$Sepal.Width, c(0, 2.5, 3, 3.5, Inf)),
                                labels = c("<=2.5", "2.6-3.0", "3.1-3.5", "3.6+"))

irisNew$Petal.Length.Cat <- factor(cut(irisNew$Petal.Length, c(0, 2, 4, 6, Inf)),
                                labels = c("<=2.0", "2.1-4.0", "4.1-6.0", "6.0+"))

irisNew$Petal.Width.Cat <- factor(cut(irisNew$Petal.Width, c(0, 1, 2, Inf)),
                              labels = c("<=1.0", "1.1-2.0", "2.1+"))
## Using NB to predict if the species is virginica
## (training and predicting on same dataset)
pred <- performNB(irisNew, irisNew, obsIDVar = "id",
                  goldStdVar = "spVirginica",
                  covariates = c("Sepal.Length.Cat", "Sepal.Width.Cat",
                                 "Petal.Length.Cat", "Petal.Width.Cat"), l = 1)
irisResults <- merge(irisNew, pred$probabilities, by = "id")
tapply(irisResults$p, irisResults$Species, summary)
```

`performPEM` *Executes the PEM algorthim to estimate the generation/serial interval distribution*

Description

The function `performPEM` uses relative transmission probabilities to estimate the generation/serial interval distribution

Usage

```
performPEM(
  df,
  indIDVar,
  timeDiffVar,
  pVar,
  initialPars,
  shift = 0,
  epsilon = 1e-05,
  plot = FALSE
)
```

Arguments

`df`	The name of the dateset with transmission probabilities.
`indIDVar`	The name (in quotes) of the individual ID columns (data frame `df` must have variables called `<indIDVar>.1` and `<indIDVar>.2`).

timeDiffVar	The name (in quotes) of the column with the difference in time between infection (generation interval) or symptom onset (serial interval) for the pair of cases. The units of this variable (hours, days, years) defines the units of the resulting distribution.
pVar	The column name (in quotes) of the transmission probabilities.
initialPars	A vector of length two with the shape and scale to initialize the gamma distribution parameters.
shift	A value in the same units as timeDiffVar that the gamma distribution should be shifted. The Default value of 0 is an unmodifed gamma distribution.
epsilon	The difference between successive estimates of the shape and scale parameters that indicates convergence.
plot	A logical indicating if a plot should be printed showing the parameter estimates at each iteration.

Details

This function is meant to be called by estimateSI which estimates the generation/serial interval distribution as well as clustering the probabilities, but can be called directly. The main reason to call performPEM directly is for the plot argument. Setting this argument to TRUE will produce a plot of the shape and scale parameters at each iteration. For more details on the PEM algorithm see estimateSI.

Value

A data frame with one row and the following columns:

- nIndividuals - the number of infectees who have intervals included in the SI estimate.
- shape - the shape of the estimated gamma distribution for the interval.
- scale - the scale of the estimated gamma distribution for the interval.
- meanSI - the mean of the estimated gamma distribution for the interval (shape * scale + shift).
- medianSI - the median of the estimated gamma distribution for the interval (qgamma(0.5, shape, scale) + shift)).
- sdSI - the standard deviation of the estimated gamma distribution for the interval (shape * scale ^ 2)

References

Hens N, Calatayud L, Kurkela S, Tamme T, Wallinga J. Robust reconstruction and analysis of outbreak data: influenza A (H1N1) v transmission in a school-based population. *American Journal of Epidemiology*. 2012 Jul 12;176(3):196-203.

See Also

nbProbabilities clusterInfectors performPEM

Examples

```
## First, run the algorithm without including time as a covariate.
orderedPair <- pairData[pairData$infectionDiffY > 0, ]

## Create a variable called snpClose that will define probable links
# (<3 SNPs) and nonlinks (>12 SNPs) all pairs with between 2-12 SNPs
# will not be used to train.
orderedPair$snpClose <- ifelse(orderedPair$snpDist < 3, TRUE,
                        ifelse(orderedPair$snpDist > 12, FALSE, NA))
table(orderedPair$snpClose)

## Running the algorithm
#NOTE should run with nReps > 1.
resGen <- nbProbabilities(orderedPair = orderedPair,
                          indIDVar = "individualID",
                          pairIDVar = "pairID",
                          goldStdVar = "snpClose",
                          covariates = c("Z1", "Z2", "Z3", "Z4"),
                          label = "SNPs", l = 1,
                          n = 10, m = 1, nReps = 1)

## Merging the probabilities back with the pair-level data
nbResultsNoT <- merge(resGen[[1]], orderedPair, by = "pairID", all = TRUE)

## Estimating the serial interval


# Using all pairs and plotting the parameters
  performPEM(nbResultsNoT, indIDVar = "individualID", timeDiffVar = "infectionDiffY",
  pVar = "pScaled", initialPars = c(2, 2), shift = 0, plot = TRUE)



# Clustering the probabilities first
allClust <- clusterInfectors(nbResultsNoT, indIDVar = "individualID", pVar = "pScaled",
                          clustMethod = "hc_absolute", cutoff = 0.05)

performPEM(allClust[allClust$cluster == 1, ], indIDVar = "individualID",
           timeDiffVar = "infectionDiffY", pVar = "pScaled",
           initialPars = c(2, 2), shift = 0, plot = TRUE)


# The above is equivalent to the following code using the function estimateSI()
# though the plot will not be printed and more details will be added
estimateSI(nbResultsNoT, indIDVar = "individualID", timeDiffVar = "infectionDiffY",
           pVar = "pScaled", clustMethod = "hc_absolute", cutoff = 0.05,
           initialPars = c(2, 2))
```

plotRt *Creates a plot of the effective reproductive number*

Description

The function plotRt creates a plot of the effective reproductive number (Rt) over the course of the outbreak. Using various options, the plot can include the overall average Rt value for the outbreak and the confidence intervals.

Usage

```
plotRt(
  rData,
  includeRtAvg = FALSE,
  includeRtCI = FALSE,
  includeRtAvgCI = FALSE
)
```

Arguments

rData	A list that is the output of estimateR. It should contain the dataframes RtDf, RtAvgDf, and vectors timeFrame and rangeForAvg
includeRtAvg	A logical. If TRUE, a horizontal line will be drawn for the average Rt value over rangeForAvg and verticle lines will be drawn at the rangeForAvg values.
includeRtCI	A logical. If TRUE, error bars will be added to the Rt values representing the bootstrap confidence intervals.
includeRtAvgCI	A logical. If TRUE, horizontal lines will be drawn around the Rt average line representing the bootstrap confidence interval.

Details

The main input rData should be the output of estimateRt with the time-level reproductive numbers, overall average, range used to calculate that average, and time frame.

The options includeRtCI and includeRtAvgCI add confidence interval bounds to the plot. If set to true, rData should be from a call of estimateRt with bootSamples > 0 so that confidence intervals are available. If includeRtAvgCI is set to TRUE, a line for the point estimate of the average Rt value will be drawn even if includeRtAvg is set to FALSE.

See Also

nbProbabilities estimateR

Examples

```
## Use the nbResults data frame included in the package which has the results
# of the nbProbabilities() function on a TB-like outbreak.

## Getting initial estimates of the reproductive number
# (without specifying nbResults and without confidence intervals)
rInitial <- estimateR(nbResults, dateVar = "infectionDate",
               indIDVar = "individualID", pVar = "pScaled",
               timeFrame = "months")

## Finding the stable portion of the outbreak for rangeForAvg using the plot
plotRt(rInitial)
cut1 <- 25
cut2 <- 125

## Finding the final reproductive number estimates with confidence intervals
# NOTE should run with bootSamples > 10.
rFinal <- estimateR(nbResults, dateVar = "infectionDate",
             indIDVar = "individualID", pVar = "pScaled",
             timeFrame = "months", rangeForAvg = c(cut1, cut2),
             bootSamples = 10, alpha = 0.05)

## Ploting the final result
plotRt(rFinal, includeRtAvg = TRUE, includeRtCI = TRUE, includeRtAvgCI = TRUE)
```

Index