Package ‘svisits’

July 19, 2016

Type Package

Title Shared Clinic Visit Dates Identify Steady HIV Positive
Partnerships for Research

Version 1.1

Date 2016-07-12

Author Alex Marzel, Teja Turk, Roger Kouyos

Maintainer Alex Marzel <Alex.Marzel@usz.ch>

Description This package helps to find some stable HIV-infected partnerships in cohort studies based on the assumption that some patients frequently attend the clinical visits together. These pairs are useful for biological and epidemiological research.

License GPL (>= 2)

Depends R (>= 2.10), Rcpp (>= 0.12.5), compiler, parallel, chron,
data.table

Imports Rcpp (>= 0.12.5), compiler, parallel, chron, data.table

LinkingTo Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

R topics documented:

`svisits-package` *svisits: finding HIV transmission and serosorting pairs using shared clinic visit dates*

Description

This package helps to find some stable HIV-infected partnerships in cohort studies based on the assumption that some patients frequently attend the clinical visits together. These pairs are useful for biological and epidemiological research.

Author(s)

Alex Marzel, Teja Turk, Roger Kouyos

Maintainer: Alex Marzel <Alex.Marzel@usz.ch>

`adjust_Rcpp_min` *Adjustment function*

Description

This is the C++ adjustment function that is called by adjust_visits. It adjusts the numbers of shared visits for each pair.

Usage

```
adjust_Rcpp_min(mat, prob)
```

Arguments

`mat`	matrix with 3 columns (unadjusted number of shared visits, number of visits of the first member, number of visits of the second member) to adjust
`prob`	probability of sharing a single visit, default 1/75

Details

written in C++

Value

a vector of adjusted number of shared visits

Examples

```
visits <- data.frame(Freq = c(1,1,1,2,3),
                     N_visits.x = c(46,39,10,40,55),
                     N_visits.y = c(68,68,68,24,24))
mat <- as.matrix(visits)
adj_mat <- adjust_Rcpp_min(mat, prob=1/75)
adj_mat
```

`adjust_R_Rcpp_short_binomial` *Probabilites of sharing a given number of visits*

Description

This is the C++ function that is called by `Bonferroni_m`.

Usage

```
adjust_R_Rcpp_short_binomial(mat, prob)
```

Arguments

`mat`	matrix with 3 columns (unadjusted number of shared visits, number of visits of the first member, number of visits of the second member)
`prob`	probability of sharing a single visit, default 1/75

Details

written in C++

Value

a vector of probabilities to share the given number of visits for each pair

Examples

```
visits <- data.frame(Freq = c(1,1,1,2,3),
                     N_visits.x = c(46,39,10,40,55),
                     N_visits.y = c(68,68,68,24,24))
mat <- as.matrix(visits)
Bonferroni_mat <- adjust_R_Rcpp_short_binomial(mat, prob=1/75)
Bonferroni_mat
```

adjust_visits *Adjust the number of shared visits*

Description

The function adjusts the total number of shared visits per pair.

Usage

```
adjust_visits(unadjusted_pairs, ids, prob = 1/75)
```

Arguments

`unadjusted_pairs`
a "table" object containing the pairs and the number of shared visits

`ids` a "data.table" object containing columns "ids" and "N_visits". The "ids" column represents patient identifiers and is the reference (see `setkey` from package `data.table` for more details). The "N_visits" contains the number of visits for each patient.

`prob` probability of sharing a single visit, depending on the frequency of visits. Default `1/75`~visit every three months.

Value

a data frame with the columns:

`"allPairs"` pair identifiers

`"Freq"` unadjusted number of shared visits

`"id_1"` first member of the pair's id

`"id_2"` second member of the pair's id

`"N_visits.x"` first member of the pair's number of visits

`"N_visits.y"` second member of the pair's number of visits

`"Corrected"` the adjusted number of shared visits per pair

Examples

```
# load data
data("simulated_data")
db_dates <- prepare_db(your_database = simulated_data,
                       ids_column = "subject",
                       dates_column = "sim_dates")

# first get unadjusted pairs
unadjusted_observed_pairs <- get_observed_pairs(db_dates)
ids <- data.table(ids = as.character(names(table(db_dates$subject))),
                  N_visits = as.character(as.numeric(table(db_dates$subject))))
```

```
setkey(ids, "ids")

# now adjust the pairs
adjusted_observed <- adjust_visits(unadjusted_pairs = unadjusted_observed_pairs,
                                   ids = ids,
                                   prob = 1/75)
head(adjusted_observed)
```

barcode *Unique pair identifier*

Description

It makes a unique identifier for each pair. Each member‘s id should be numeric! For example for `id_1=6758` and `id_2=3456` the barcode will be `"3456_6578"`.

Usage

```
barcode(raw_for_eval)
```

Arguments

`raw_for_eval` a vector with two ids

Value

string of a form id1_id2

Examples

```
barcode(c(id_2=6578,id_1=3456))
barcode(c(3456, 6578))
```

Bonferroni_m *Detecting the pairs using Bonferroni correction*

Description

This method is much faster than the shuffling method, but less precise.

Usage

```
Bonferroni_m(unadjusted_pairs,
             ids=ids,
             prob = 1/75,
             alpha = 0.01,
             only_significant = TRUE)
```

Arguments

unadjusted_pairs
: a "table" object containing the pairs and the number of shared visits. See get_observed_pairs.

ids
: a "data.table" object with columns "ids" and "N_visits". The "ids" column represents patient identifiers and is the reference (see setkey from package data.table for more details). The "N_visits" contains the number of visits for each patient.

prob
: probability of sharing a single visit. Default 1/75.

alpha
: type I error

only_significant
: returns only significant pairs above the threshold (default) or all the pairs

Value

a data frame with

"allPairs" pair identifier

"Freq" the unadjusted number of shared visits

"id_1" identifier of the first pair member

"id_2" identifier of the second pair member

"N_visits.x" total number of visits of the first pair member

"N_visits.y" total number of visits of the second pair member

"Prob_for_Bonferr" probability of sharing the given number of shared visits

"BP" row number

"ltp" the log-transformed probablity of sharing the given number shared visits

Examples

```
# load data
data("simulated_data")
db_dates <- prepare_db(your_database = simulated_data,
                       ids_column = "subject",
                       dates_column = "sim_dates")

# first get unadjusted pairs
unadjusted_observed_pairs <- get_observed_pairs(db_dates)
# prepare ids
ids <- data.table(ids = as.character(names(table(db_dates$subject))),
                  N_visits = as.character(as.numeric(table(db_dates$subject))))
setkey(ids, "ids")

# run
Bonferroni_m_output <- Bonferroni_m(unadjusted_observed_pairs,
                                    ids = ids, prob = 1/75, alpha = 0.01)
# number of significant pairs
nrow(Bonferroni_m_output)
```

chooseC *Binomial coefficient*

Description

It calculates the binomial coefficient.

Usage

```
chooseC(n, k)
```

Arguments

n	set size
k	number of unordered outcomes

Details

implemented in C++

Value

numeric

Examples

```
chooseC(n=6, k=2)
```

find_pairs *Find significant pairs*

Description

Find significant pairs using the Bonferroni method and (optionally) with the shuffling algorithm to avoid False-Positives. This is the wrap-up of the following functions:

- prepare_db
- get_observed_pairs
- Bonferroni_m
- adjust_visits (only if the shuffling algorithm is performed)
- Shuffling_simulation (optional)

Usage

```
find_pairs(your_database,
           ids_column,
           dates_column,
           prob = 1/75,
           alpha = 0.01,
           only_significant = TRUE,
           shuffling = FALSE,
           n_shuffl = 100,
           period_months = 3,
           FP_threshold = max,
           max_threshold = NULL)
```

Arguments

`your_database` a data frame with the clinic visit dates in a long format.

`ids_column` character, column name with the person identifiers

`dates_column` character, column name with the visit dates

`prob` probability of sharing a single visit, default `1/75`

`alpha` type I error

`only_significant`
returns only the significant pairs above the threshold (default) or all the pairs with the corresponding p-values

`shuffling` logical indicating if the shuffling should be performed. By default (`FALSE`) only the Bonferroni method is used.

`n_shuffl` number of repetitions of shuffling algorithm (default 100)

`period_months` the length of the follow-up period in months (by default `3`)

`FP_threshold` the function used to get the False-Positive fraction from the shuffling repetitions at each position. By default the `max` is called at each threshold. Another possibilities are `mean` or `median`.

`max_threshold` maximum false-positive threshold to be inspected. If it is `NULL`, the 99.995%-quantile of the distribution of the unadjusted number of shared visits is used.

Value

A list containing

`"unadjusted_pairs"` pairs with unadjusted number of shared visits

`"Bonferroni_pairs"` pairs obtained with the Bonferroni correction. If `significant_only==TRUE` only those above the threshold are returned.

`"adjusted_pairs"` pairs with adjusted number of shared visits returned only if `shuffling==TRUE`.

`"Shuffled_pairs"` pairs obtained by the shuffling algorithm (returned only if `shuffling==TRUE`).

`"shuffling_threshold"` threshold determined by the shuffling algorithm

`"shuffling_simulation_output"` output from the shuffling function if `shuffling==TRUE`

See Also

prepare_db, get_observed_pairs, Bonferroni_m, ajudst_visits, Shuffling_simulation

Examples

```
data("simulated_data")
pairs <- find_pairs(simulated_data,
                    ids_column = "subject",
                    dates_column = "sim_dates",
                    shuffling = TRUE, n_shuffl = 2)
```

---

getPairs *Pairs with shared visit on a certain date*

Description

From a given list of patient with the clinic visit on a certain date the function returns all the pairs.

Usage

```
getPairs(ids)
```

Arguments

ids	a vector of patient identifiers

Details

This function is called by get_observed_pairs which date-wise finds all the pairs with at least one shared visit (for all the dates that appear in your database).

Value

a character vector containing the barcodes of the found pairs on a certain date

Examples

```
dates_long <- data.frame(subject=c(12,15,2,202,31,3),
                         dates=as.Date(c(rep("2001-03-16", 5), "2011-03-17")))
# subjects with a visit on "2001-03-16"
id_s <- dates_long$subject[which(dates_long$dates==as.Date("2001-03-16"))]
getPairs(id_s)
```

`get_observed_pairs` *Pairs with at least one shared visit*

Description

FInd all the pairs that shared at least a single visit.

Usage

```
get_observed_pairs(tested_db)
```

Arguments

`tested_db` visits in long format with each row representing a visit date. See `simulated_data` for an example. It should contain columns `"subject"` with (numeric) patient identifiers and `"dates"` with the dates of clinic visits.

Value

a table object containing the number of shared visits for all pairs with at least one shared visit

Examples

```
# a short example (with artificial data)
dates_long <- data.frame(subject=c(12,15,2,202,31,15, 202),
                         dates=as.factor(c(rep("2001-03-16",5),rep("2011-06-20",2))))
get_observed_pairs(dates_long)

# example with the simulated dates based on the real distribution
# load data
data("simulated_data")
db_dates <- prepare_db(your_database = simulated_data,
                       ids_column = "subject",
                       dates_column = "sim_dates")

# get unadjusted pairs
unadjusted_observed_pairs <- get_observed_pairs(db_dates)
```

`prepare_db` *Prepare the database*

Description

Transform the database to the appropriate format for finding pairs.

Usage

```
prepare_db(your_database, ids_column, dates_column)
```

Arguments

`your_database`	the data frame to be analyzed
`ids_column`	character, column name with the person identifiers
`dates_column`	character, column name with the visit dates

Details

The dates are ordered in an increasing order within each subject.

Value

a data frame with two columns:

`"subject"` person identifiers

`"dates"` dates of the clinic visits

Examples

```
data("simulated_data")
db_dates<-prepare_db(your_database = simulated_data,
                     ids_column = "subject",
                     dates_column = "sim_dates")
```

`Shuffling_simulation` *Shuffling simulation to determine shared visits false-positive threshold*

Description

Shuffling simulation to determine shared visits false-positive threshold. It performs only one run. Please use `"replicate"` for several (see Example).

Usage

```
Shuffling_simulation(db_dates, ids, adjusted_observed, max_threshold=NULL)
```

Arguments

`db_dates`	a data frame with visits to be shuffled (in long format) which column "fupdate-Period" containing the time period which each clinic visit belongs to.
`ids`	a `"data.table"` object with columns "ids" and "N_visits". The "ids" column represents patient identifiers and it is the reference (see `setkey` from package `data.table` for more details). The "N_visits" contains the number of visits for each patient.
`adjusted_observed`	a data frame with adjusted observed number of visits.
`max_threshold`	a maximum false-positive threshold to be inspected. If it is `NULL`, the 99.995%-quantile of the distribution of the unadjusted number of shared visits is used.

Details

Due to the multiple comparisons problem and strongly simplifying assumptions about the uniform distribution of visits, the corrected number of shared visits is not an optimal test statistic. Instead, the visits are first shuffled within each period (such that the original distribution of the number of visits per individual is preserved) and the number of randomly collided shared visits per pair is counted and penalized using "adjust_visits". In other words, the observed visit dates from a given period (by default a quater) are randomly re-assigned between the patients that attended the clinic during this period.

Define the number of desired shuffling simulations (ideally > 100, but it will take some time) and run.

Value

a numeric vector of size `max_threshold`. Each value represents the false positive fraction for a cutoff of adjusted shared visits that corresponds to the position (index) of the value in the vector.

Examples

```
# load data
data("simulated_data")
db_dates <- prepare_db(your_database = simulated_data,
                      ids_column = "subject",
                      dates_column = "sim_dates")

# prepare 3 months periods for later shuffling
db_dates <- cbind(db_dates,month.day.year(db_dates$dates))
periodMo <- 3
db_dates <- cbind(db_dates,
                  fupdatePeriod=with(db_dates, year+round(floor((month-1)/periodMo)
                                                           *periodMo/12,
                                                          digits=2)))

# first get unadjusted pairs
unadjusted_observed_pairs <- get_observed_pairs(db_dates)
ids <- data.table(ids=as.character(names(table(db_dates$subject))),
                  N_visits=as.character(as.numeric(table(db_dates$subject))))
setkey(ids, "ids")

# now adjust the pairs
adjusted_observed <- adjust_visits(unadjusted_pairs = unadjusted_observed_pairs,
                                  ids = ids,
                                  prob = 1/75)
# number of simulations
n_shuffl <- 2 # two
Shuffling_simulation_output <- replicate(n_shuffl,
                                         Shuffling_simulation(db_dates,
                                                              ids=ids,
                                                              adjusted_observed),
                                         simplify=FALSE)
head(Shuffling_simulation_output)
```

`simulated_data` *Simulated dataset of visit dates based on the SHCS*

Description

It contains 20 pseudo transmission pairs that should be detected with the method.

Usage

```
data("simulated_data")
```

Format

A data frame with 90266 observations on the following 3 variables:

`X` row's indexing

`subject` a numeric vector with patient identifiers

`sim_dates` a factor with the visit dates

Examples

```
data("simulated_data")
head(simulated_data)
```

Index