Package ‘svisits’

July 15, 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 basing 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 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

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 basing on the assumption that some patients frequently attend the clinical visits together. These pairs are useful for biological and epidemiological research.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Alex Marzel, Teja Turk, Roger Kouyos

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

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

adjust_Rcpp_min — *adjustment function*

Description

This is the C++ adjustment function that is called by adjust_visits

Arguments

`mat`	matrix to adjust
`prob`	probability of sharing a single visit, default 1/75

Details

written in C++

Value

returns the adjusted matrix, the column "Corrected" is 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)
```

`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

Arguments

`mat`	pairs as found by the function get_observed_pairs
`prob`	probability of sharing a single visit, default 1/75

Details

written in C++

Value

returns a matrix with a probability to share the given number of visits for each 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)
# 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)
length(Bonferroni_m_output[,1])
```

`adjust_visits` *Adjust the number of shared visits*

Description

Adjusts the total number of shared visits per pair

Usage

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

Arguments

`unadjusted_pairs`	The pairs
`ids`	patient ids
`prob`	probability of sharing a single visit, depends on the frequency of visits. Default 1/75 ~visit every three months.

Value

returns data.frame with the column "Corrected" representing the adjsuted 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

Makes a unique identifier for each pair. Each member‘s id should be numeric! For example id_1=6758 , id_2=3456 the barcode will be 3456_6578

Usage

```
barcode(raw_for_eval)
```

Arguments

`raw_for_eval`	

Value

string of a form id1_id2

Examples

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

## The function is currently defined as
function (raw_for_eval)
{
    return(as.character(paste0(min(as.numeric(raw_for_eval)),
        "_", max(as.numeric(raw_for_eval)))))
  }
```

`Bonferroni_m` *Detecting the pairs using Bonferroni correction*

Description

much faster than the shuffling

Usage

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

Arguments

`unadjusted_pairs`	pairs as found by the function get_observed_pairs
`ids`	patient ids
`prob`	probability of sharing a single visit, default 1/75
`alpha`	type 1 error
`only_significant`	return only significant pairs above the threshold (default) or all the pairs

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)
length(Bonferroni_m_output[,1])
```

---

chooseC *Binomial coefficient*

Description

Calculates the Binomial coefficient

Arguments

n	size
k	number of unordered outcomes

Value

number

Examples

```
chooseC(n=6,k=2)
# 15
```

---

getPairs *Pairs that collided at a certain date*

Description

Pairs that shared at least a single visit at a certain date

Usage

```
getPairs(ids)
```

Arguments

`ids`

Details

this function is called by get_observed_pairs which goes date by date for each unique date

Examples

```
# 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)
```

`get_observed_pairs` *Find all the pairs that shared at least a single visit*

Description

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

Usage

```
get_observed_pairs(tested_db)
```

Arguments

`tested_db`	visits in a long format, each row a visit date, see data("simulated_data") for an example

Examples

```
# 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

Prepare the database for finding pairs.

Usage

```
prepare_db(your_database = simulated_data,ids_column = "subject",dates_column = "sim_dates")
```

Arguments

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

Details

the dates are ordered in an increasing order within each subject

Value

returns the database with two columns "subject" and "dates"

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, one run, use "replicate" for several, see example

Usage

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

Arguments

`db_dates`	visits to be shuffled (long format) an output from prepare_db, see data("simulated_data)
`ids`	patients ids
`adjusted_observed`	The dataset with adjusted observed number of visits

Details

Due to the multiple comparisons problem and a 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 quarter (such that the original distribution of the number of visits per individual was 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 quarter were randomly re-assigned between the patients that attended during this quarter.

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

Value

returns a numeric vector of size 10 with each value representing the false positive fraction for a cutoff of adjusted shared visits that corresponds to the position of the value in the order of 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$dates));periodMonths<-3
db_dates<-cbind(db_dates,fupdatePeriod=db_dates$year
                +round(floor((db_dates$month-1)/periodMonths) *periodMonths/12,2))

# 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

Contatins 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` just indexing

`subject` a numeric vector with patient identifiers

`sim_dates` a factor with the visit dates

Examples

```
data(simulated_data)
```

Index