Package: HospitalNetwork (via r-universe)

August 21, 2024

Type Package

Title Building Networks of Hospitals Through Patients Transfers

Version 0.9.3

Description Set of tools to help interested researchers to build hospital networks from data on hospitalized patients transferred between hospitals. Methods provided have been used in Donker T, Wallinga J, Grundmann H. (2010) <doi:10.1371/journal.pcbi.1000715>, and Nekkab N, Crépey P, Astagneau P, Opatowski L, Temime L. (2020) <doi:10.1038/s41598-020-71212-6>.

URL https://pascalcrepey.github.io/HospitalNetwork/

BugReports https://github.com/PascalCrepey/HospitalNetwork/issues

License GPL-3

Encoding UTF-8

LazyLoad true

Imports checkmate, igraph, lubridate, R6, ggplot2, ggraph

Depends data.table

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 2.1.0), shiny, shinyWidgets, shinydashboard, DT, shinyalert, shinyjs, vdiffr, pander, glue, golem, htmltools

VignetteBuilder knitr

Language en-US

Repository https://pascalcrepey.r-universe.dev

RemoteUrl https://github.com/pascalcrepey/hospitalnetwork

RemoteRef HEAD

RemoteSha 923bec40e055f459a12a2de3f7aa899184e24a98

Contents

`adjust_overlapping_stays`

Check and fix overlapping admissions.

Description

This function checks if a discharge (n) is not later than the next (n+1) admission. If this is the case, it sets the date of discharge n to date of discharge n+1, and creates an extra record running from discharge n+1 to discharge n. If the length of stay of this record is negative, it removes it. It is possible that one pass of this algorithm doesn't clear all overlapping admissions (e.g. when one admission overlaps with more than one other admission), it is therefore iterated until no overlapping admissions are found. Returns the corrected database.

Usage

```
adjust_overlapping_stays(
  report,
  maxIteration = 25,
  verbose = FALSE,
  retainAuxData = TRUE,
  ...
)
```

Arguments

report	(list). A list containing the base and in which will be stored reporting variables. The base is a patient discharge database, in the form of a data.table. The data.table should have at least the following columns: sID: subjectID (character) fID: facilityID (character) Adate: admission date (POSIXct, but character can be converted to POSIXct) Ddate: discharge date (POSIXct, but character can be converted to POSIXct)
maxIteration	(integer) the maximum number of times the function will try and remove overlapping admissions.
verbose	(boolean) print diagnostic messages. Default is FALSE.
retainAuxData	(boolean) allow retaining additional data provided in the database. Default is TRUE.
...	other parameters passed on to internal functions

Value

The corrected database as data.table.

---

all_admissions_summary

Summary statistics on entire database

Description

Function that extracts summary statistics from entire database

Usage

```
all_admissions_summary(base, verbose = FALSE, ...)
```

Arguments

base	(data.table). A subject discharge database, in the form of a data.table. The data.table should have at least the following columns: sID: subjectID (character) fID: facilityID (character) Adate: admission date (date) Ddate: discharge date (date)
verbose	(boolean) print diagnostic messages. Default is TRUE.
...	other parameters passed on to internal functions

Value

a list of summary statistics: - meanLOS: The mean length of stay, in days - meanTBA: The mean time between admissions, in days - totalAdmissions: Total number of admissions (i.e. number of records in the database) - numSubjects: Number of unique subjects - numFacilities: Number of unique facilities - LOSdistribution: Distribution of length of stay - TBAdistribution: Distribution of time between admissions

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
all_admissions_summary(myBase)
```

---

checkBase *General check function*

Description

Function that performs various checks to ensure the database is correctly formatted, and adjusts overlapping patient records.

Usage

```
checkBase(
  base,
  convertDates = FALSE,
  dateFormat = NULL,
  deleteMissing = NULL,
  deleteErrors = NULL,
  subjectID = "sID",
  facilityID = "fID",
  disDate = "Ddate",
  admDate = "Adate",
  maxIteration = 25,
  retainAuxData = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

base	(data.table). A patient discharge database, in the form of a data.table. The data.table should have at least the following columns: sID: patientID (character) fID: facilityID (character) Adate: admission date (POSIXct, but character can be converted to POSIXct) Ddate: discharge date (POSIXct, but character can be converted to POSIXct)
convertDates	(boolean) indicating if dates need to be converted to POSIXct if they are not
dateFormat	(character) giving the input format of the date character string (e.g. "ymd" for dates like "2019-10-30") See parse_date_time for more information on the format.
deleteMissing	(character) How to handle records that contain a missing value in at least one of the four mandatory variables: NULL (default): do not delete. Stops the function with an error message. "record": deletes just the incorrect record. "patient": deletes all records of each patient with one or more incorrect records.

deleteErrors	(character) How incorrect records should be deleted: "record" deletes just the incorrect record "patient" deletes all records of each patient with one or more incorrect records.
subjectID	(character) the columns name containing the subject ID. Default is "sID"
facilityID	(character) the columns name containing the facility ID. Default is "fID"
disDate	(character) the columns name containing the discharge date. Default is "Ddate"
admDate	(character) the columns name containing the admission date. Default is "Adate"
maxIteration	(integer) the maximum number of times the function will try and remove overlapping admissions
retainAuxData	(boolean) allow retaining additional data provided in the database. Default is TRUE.
verbose	(boolean) print diagnostic messages. Default is TRUE.
...	other parameters passed on to internal functions

Value

The adjusted database as a data.table with a new class attribute "hospinet.base" and an attribute "report" containing information related to the quality of the database.

See Also

parse_date_time

Examples

```
## create a "fake and custom" data base
mydb = create_fake_subjectDB(n_subjects = 100, n_facilities = 100)
setnames(mydb, 1:4, c("myPatientId", "myHealthCareCenterID", "DateOfAdmission", "DateOfDischarge"))
mydb[,DateOfAdmission:= as.character(DateOfAdmission)]
mydb[,DateOfDischarge:= as.character(DateOfDischarge)]

head(mydb)
#   myPatientId myHealthCareCenterID DateOfAdmission DateOfDischarge
#1:        s001                 f078      2019-01-26      2019-02-01
#2:        s002                 f053      2019-01-18      2019-01-21
#3:        s002                 f049      2019-02-25      2019-03-05
#4:        s002                 f033      2019-04-17      2019-04-21
#5:        s003                 f045      2019-02-02      2019-02-04
#6:        s003                 f087      2019-03-12      2019-03-19

str(mydb)
#Classes ‘data.table’ and 'data.frame': 262 obs. of  4 variables:
# $ myPatientId         : chr  "s001" "s002" "s002" "s002" ...
# $ myHealthCareCenterID: chr  "f078" "f053" "f049" "f033" ...
# $ DateOfAdmission     : chr  "2019-01-26" "2019-01-18" "2019-02-25" "2019-04-17" ...
# $ DateOfDischarge     : chr  "2019-02-01" "2019-01-21" "2019-03-05" "2019-04-21" ...
#- attr(*, ".internal.selfref")=<externalptr>

my_checked_db = checkBase(mydb,
```

```
      subjectID = "myPatientId",
      facilityID = "myHealthCareCenterID",
      disDate = "DateOfDischarge",
      admDate = "DateOfAdmission",
      convertDates = TRUE,
      dateFormat = "ymd")

#Converting Adate, Ddate to Date format
#Checking for missing values...
#Checking for duplicated records...
#Removed 0 duplicates
#Done.

head(my_checked_db)
#    sID  fID      Adate      Ddate
#1: s001 f078 2019-01-26 2019-02-01
#2: s002 f053 2019-01-18 2019-01-21
#3: s002 f049 2019-02-25 2019-03-05
#4: s002 f033 2019-04-17 2019-04-21
#5: s003 f045 2019-02-02 2019-02-04
#6: s003 f087 2019-03-12 2019-03-19
str(my_checked_db)
#Classes 'hospinet.base', 'data.table' and 'data.frame': 262 obs. of  4 variables:
#$ sID  : chr  "s001" "s002" "s002" "s002" ...
#$ fID  : chr  "f078" "f053" "f049" "f033" ...
#$ Adate: POSIXct, format: "2019-01-26" "2019-01-18" "2019-02-25" "2019-04-17" ...
#$ Ddate: POSIXct, format: "2019-02-01" "2019-01-21" "2019-03-05" "2019-04-21" ...
# ...

## Show the quality report
attr(my_checked_db, "report")
```

---

checkFormat *Check database format*

Description

Function that performs various generic checks to ensure that the database has the correct format

Usage

```
checkFormat(report, convertDates = FALSE, dateFormat = NULL, verbose = TRUE)
```

Arguments

report	(list). A list containing the base and in which will be stored reporting variables. The base is a patient discharge database, in the form of a data.table. The data.table should have at least the following columns: sID: subjectID (character) fID: facilityID (character) Adate: admission date (POSIXct, but character can be converted to POSIXct) Ddate: discharge date (POSIXct, but character can be converted to POSIXct)

convertDates	(boolean) TRUE/FALSE: whether the dates should converted. Default is TRUE.
dateFormat	(boolean) The format of date as a character string (e.g. %y%m%d for 20190524, or %d-%m-%y for 24-05-2019).
verbose	(boolean) print diagnostic messages. Default is FALSE.

Value

Returns either an error message, or the database (modified if need be).

`create_fake_subjectDB` *Create a fake subject database*

Description

Create a fake subject database

Usage

```
create_fake_subjectDB(
  n_subjects = 100,
  n_facilities = 10,
  avg_n_stays = 3,
  days_since_discharge = NULL,
  length_of_stay = NULL,
  start_id_subjects = 1,
  start_id_facilities = 1,
  with_errors = FALSE
)
```

Arguments

n_subjects	the number of different subjects in the database
n_facilities	the number of facility present in the database
avg_n_stays	the average number of stays per subject
days_since_discharge	the number of days between a discharge date and an admission date (default: max(0, rnorm(1, mean = 30, sd = 10)))
length_of_stay	the length of stay (default: max(1, rnorm(1, mean = 5, sd = 3))
start_id_subjects, start_id_facilities	change starting ids (used for clustered network)
with_errors	(boolean) introduce or not random errors in the database. Default to FALSE.

Value

a data.table containing all subjects stays

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
mydb
```

`create_fake_subjectDB_clustered` *Create a fake subject database with clustering*

Description

Create a fake subject database with clustering

Usage

```
create_fake_subjectDB_clustered(
  n_subjects = 50,
  n_facilities = 10,
  avg_n_stays = 3,
  days_since_discharge = NULL,
  length_of_stay = NULL,
  n_clusters = 3
)
```

Arguments

`n_subjects`	the number of different subjects in the database
`n_facilities`	the number of facility present in the database
`avg_n_stays`	the average number of stays per subject
`days_since_discharge`	the number of days between a discharge date and an admission date (default: max(0, rnorm(1, mean = 30, sd = 10)))
`length_of_stay`	the length of stay (default: max(1, rnorm(1, mean = 5, sd = 3))
`n_clusters`	the number of cluster in the network

Value

a data.table containing all subjects stays

Examples

```
mydb <- create_fake_subjectDB_clustered(n_subjects = 100, n_facilities = 10)
mydb
```

create_subject_stay *Create a fake subject stay*

Description

create_subject_stay is an internal function used by create_fake_subjectDB.

Usage

```
create_subject_stay(
  sID,
  fID,
  last_discharge_date = NULL,
  days_since_discharge = NULL,
  length_of_stay = NULL
)
```

Arguments

sID	the subject ID
fID	the facility ID
last_discharge_date	the last discharge date
days_since_discharge	the number of days since last discharge (default: max(0, rnorm(1, mean = 30, sd = 10)))
length_of_stay	the length of stay (default: max(1, rnorm(1, mean = 5, sd = 3))

Value

a one row data.table corresponding to the subject stay.

edgelist_from_base *Compute the edgelist of a network from a database of movements records.*

Description

This function computes the edgelist of a network of facilities across which subjects can be transferred. The edgelist is computed from a database that contains the records of the subjects' stays in the facilities.

Usage

```
edgelist_from_base(
  base,
  window_threshold = 365,
  count_option = "successive",
  prob_params = c(0.0036, 1/365, 0.128),
  condition = "dates",
  noloops = TRUE,
  nmoves_threshold = NULL,
  flag_vars = NULL,
  flag_values = NULL,
  verbose = FALSE
)
```

Arguments

base	(data.table) A database of records of stays of subjects in facilities. The table should have at least the following columns: • **subjectID** (character) unique subject identifier • **facilityID** (character) unique facility identifier • **admDate** (POSIXct) date of admission in the facility • **disDate** (POSIXct) date of discharge of the facility
window_threshold	(integer) A number of days. If two stays of a subject at two facilities occurred within this window, this constitutes a connection between the two facilities (given that potential other conditions are met).
count_option	(character) How to count connections. Options are "successive", "probability" or "all". See details.
prob_params	(vector of numeric) Three numerical values to calculate the probability that a movement causes an introduction from hospital A to hospital B. See Donker T, Wallinga J, Grundmann H. (2010) <doi:10.1371/journal.pcbi.1000715> for more details. For use with count_option="probability". prob_params[1] is the rate of acquisition in hospital A (related to LOS in hospital A). Default: 0.0036 prob_params[2] is the rate of loss of colonisation (related to time between admissions). Default: 1/365 prob_params[4] is the rate of transmission to other patients in hospital B (related to LOS in hospital B). Default: 0.128
condition	(character) Condition(s) used to decide what constitutes a connection. Can be "dates", "flags", or "both". See details.
noloops	(boolean). Should transfers within the same nodes (loops) be kept or set to 0. Defaults to TRUE, removing loops (setting matrix diagonal to 0).
nmoves_threshold	(numeric) A threshold for the minimum number of subject transfer between two facilities. Set to NULL to deactivate, default to NULL.
flag_vars	(list) Additional variables that can help flag a transfer, besides the dates of admission and discharge. Must be a named list of two character vectors which are

	the names of the columns that can flag a transfer: the column that can flag a potential origin, and the column that can flag a potential target. The list must be named with "origin" and "transfer". Eg: list("origin" = "var1", "target" = "var2"). See details.
flag_values	(list) A named list of two character vectors which contain the values of the variables in flag_var that are matched to flag a potential transfer. The list must be named with "origin" and "transfer". The character vectors might be of length greater than one. Eg: list("origin" = c("value1", "value2"), "target" = c("value2", "value2")). The values in 'origin' and 'target' are the values that flag a potential origin of a transfer, or a potential target, respectively. See details.
verbose	TRUE to print computation steps

Details

The edgelist contains the information on the connections between nodes of the network, that is the movements of subjects between facilities. The edgelist can be in two different formats: long or aggregated. In long format, each row corresponds to a single movement between two facilities, therefore only two columns are needed, one containing the origin facilities of a movement, the other containing the target facilities. In aggregated format, the edgelist is aggregated by unique pairs of origin-target facilities.

Value

A list of two data.tables, which are the edgelists. One in long format (el_long), and one aggregated by pair of nodes (el_aggr).

See Also

matrix_from_edgelist, matrix_from_base

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
edgelist_from_base(myBase)
```

---

get_betweenness *Compute the betweenness centrality*

Description

Compute the betweenness centrality

Usage

```
get_betweenness(graph)
```

Arguments

graph	an igraph object

Value

a data.table containing the centrality measure

---

get_closeness *Compute closeness*

Description

Compute one or several closeness measure for facility networks.

Usage

```
get_closeness(graph, modes = "total")
```

Arguments

graph	an igraph object
modes	option passed on to igraph::closeness : "out", "in", "all", "total"

Value

a data.table containing the closeness measure

See Also

closeness

---

get_clusters *Compute the clusters*

Description

Compute the clusters

Usage

```
get_clusters(graph, algos, undirected, ...)
```

Arguments

graph	an igraph object
algos	the type of algorithm, single argument describing a cluster function from the igraph package
undirected	either "mutual" or "arbitrary"
...	other arguments to be passed on to the algorithm

Value

a data.table

---

get_degree *Compute the degree of each nodes in the network*

Description

Compute the degree of each nodes in the network

Usage

```
get_degree(graph, modes = c("in", "out", "total"))
```

Arguments

graph	an igraph object
modes	the type of degree: "in", "out", "total"

Value

a data.table of nodes degree

---

get_hubs_bycluster *Function computing hub scores of nodes by group*

Description

Function computing hub scores of nodes by group

Usage

```
get_hubs_bycluster(graphs, name, ...)
```

Arguments

graphs	A list of igraph graphs, one for each group within which the hub scores will be computed
name	[character (1)] The name of grouping variable (used only for naming the column of the DT)
...	Optional arguments to be passed to igraph function 'hub_score()'

See Also

hub_score

---

get_hubs_global *Function computing hub scores for each node. If bycluster = TRUE, hub scores are computed by cluster*

Description

Function computing hub scores for each node. If bycluster = TRUE, hub scores are computed by cluster

Usage

```
get_hubs_global(graph, ...)
```

Arguments

graph	An igraph graph
...	other arguments to be passed to igraph function hub_score()

See Also

hub_score

---

get_matrix_bycluster *Function returning matrices of transfers within each by clusters*

Description

Function returning matrices of transfers within each by clusters

Usage

```
get_matrix_bycluster(mat, DT, clusters)
```

Arguments

mat	The adjacency matrix of the network
DT	A data table with at least a column 'node' and a factor column identifying the node's cluster
clusters	A unique character vector of the name of the column identifying the nodes' clusters

---

get_metrics *Compute network metrics*

Description

Function computing different network analysis metrics.

Usage

```
get_metrics(
  network,
  mode = "directed",
  weighted = TRUE,
  transfers = TRUE,
  metrics = c("degree", "closeness", "clusters", "betweenness"),
  clusters = c("cluster_fast_greedy", "cluster_infomap"),
  hubs = "all_clusters",
 options = list(degree = list(modes = c("in", "out", "total")), closeness = list(modes =
  "total"), betweenness = list(), cluster_fast_greedy = list(undirected = "collapse"),
    cluster_infomap = list(undirected = "collapse"), clusters = list(algos =
    c("cluster_fast_greedy", "cluster_infomap"), undirected = "collapse"))
)
```

Arguments

network	the network to analyze. Must be an igraph, HospiNet or a square adjacency matrix (n*n).
mode	either "directed" or "undirected" network measures
weighted	TRUE if the network is weighted
transfers	TRUE if metrics specific to subject transfers must be computed
metrics	list of the metrics to compute
clusters	choose between cluster algorithm: cluster_fast_greedy or cluster_infomap
hubs	choose between getting hubs from "all_clusters" or "global"
options	named list of options to be passed to the igraph functions

`HospiNet` *Class providing the HospiNet object with its methods*

Description

Class providing the HospiNet object with its methods

Class providing the HospiNet object with its methods

Format

`R6Class` object.

Value

Object of `R6Class` with methods for accessing facility networks.

Methods

`new(edgelist, window_threshold, nmoves_threshold, noloops)` This method is used to create an object of this class with `edgelist` as the necessary information to create the network. The other arguments `window_threshold`, `nmoves_threshold`, and `noloops` are specific to the `edgelist` and need to be provided. For ease of use, it is preferable to use the function `hospinet_from_subject_database`

`print()` This method prints basic information about the object.

`plot(type = "matrix")` This method plots the network matrix by default. The argument `type` can take the following values:

matrix plot the network matrix,

clustered_matrix identify and plot cluster(s) in the matrix using the infomap algorithm (from igraph),

degree plot the histogram of the number of neighbors by facility,

circular_network plot the network by clusters using a "spaghetti-like" layout. Only works when there are at least 2 clusters.

Active bindings

`edgelist` (data.table) the list of edges (origin, target) and their associated number of movements (N) (read-only)

`edgelist_long` (data.table) edgelist with additional information (read-only)

`matrix` (matrix) the transfer matrix (active binding, read-only)

`igraph` (igraph) the igraph object corresponding to the network (active binding, read-only)

`n_facilities` the number of facilities in the network (read-only)

`n_movements` the total number of subject movements in the network (read-only)

`window_threshold` the window threshold used to compute the network (read-only)

`nmoves_threshold` the nmoves threshold used to compute the network (read-only)

`noloops` TRUE if loops have been removed (read-only)

`hist_degrees` histogram data of the number of connections per facility

`LOSPerHosp` the mean length of stay for each facility (read-only)

`admissionsPerHosp` the number of admissions to each facility (read-only)

`subjectsPerHosp` the number of unique subjects admitted to each facility (read-only)

`degrees` number of connections for each facilities (total, in, and out)(read-only)

`closenesss` the closeness centrality of each facility (read-only)

`betweennesss` the betweenness centrality of each facility (read-only)

`cluster_infomap` the assigned community for each facility, based on the infomap algorithm (read-only)

`cluster_fast_greedy` the assigned community for each facility, based on the greedy modularity optimization algorithm (read-only)

`hubs_global` Kleinberg's hub centrality scores, based on the entire network (read-only)

`hubs_infomap` same as hubs_global, but computed per community based on the infomap algorithm (read-only)

`hubs_fast_greedy` same as hubs_global, but computed per community based on the infomap algorithm (read-only)

`metricsTable` (data.table) all of the above metrics for each facility (read-only)

Methods

Public methods:

- `HospiNet$new()`
- `HospiNet$print()`
- `HospiNet$plot()`
- `HospiNet$clone()`

Method `new()`**:** Create a new HospiNet object.

Usage:

```
HospiNet$new(
  edgelist,
  edgelist_long,
  window_threshold,
  nmoves_threshold,
  noloops,
  prob_params,
  fsummary = NULL,
  create_MetricsTable = FALSE
)
```

Arguments:

`edgelist` Short format edgelist

`edgelist_long` Long format edgelist

`window_threshold` The window threshold used to compute the network

`nmoves_threshold` The nmoves threshold used to compute the network

`noloops` TRUE if loops have been removed

`prob_params` Currently unused

`fsummary` A pre-built data.table with the LOSPerHosp, subjectsPerHosp and admissionsPerHosp that don't need to be recomputed.

`create_MetricsTable` all of the metrics for each facility

Returns: A new 'HospiNet' object

Method `print()`**:** Prints a basic description of the number of facilities and movements of a HospiNet object.

Usage:

```
HospiNet$print()
```

Returns: NULL

Method `plot()`**:** Plots various representations of the HospiNet network

Usage:

```
HospiNet$plot(type = "matrix", ...)
```

Arguments:

`type` One of "matrix", "degree", "clustered_matrix", "circular network" Choose what you would like to plot - the connectivity matrix, degree distribution, the clusters, or the network in a circle.

`...` Additional arguments to be provided. Only supported for 'type == 'circular_network''.

Returns: a 'ggplot2' object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HospiNet$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
mydbsmall <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)

hn <- hospinet_from_subject_database(
  base = checkBase(mydbsmall),
  window_threshold = 10,
  count_option = "successive",
  condition = "dates"
)

hn

plot(hn)
plot(hn, type = "clustered_matrix")
```

`hospinet_from_subject_database` *Create HospiNet object from subject database*

Description

This function creates a HospiNet object from the database containing subjects stays.

Usage

```
hospinet_from_subject_database(
  base,
  window_threshold = 365,
  count_option = "successive",
  condition = "dates",
  prob_params = c(0.0036, 1/365, 0.128),
  noloops = TRUE,
  nmoves_threshold = NULL,
  flag_vars = NULL,
  flag_values = NULL,
  create_MetricsTable = TRUE,
  verbose = FALSE,
  shinySession = NULL,
  ...
)
```

Arguments

`base`	(hospinet.base) A database of records of stays of subjects in facilities. This can be obtained using the function `checkBase`.
`window_threshold`	(numeric) A threshold for the number of days between discharge and admission to be counted as a transfer. Set to 0 for same day transfer, default is 365 days.
`count_option`	(character) TODO. Default is "successive".
`condition`	(character) TODO. Default is "dates".
`prob_params`	(vector of numeric) Three numerical values to calculate the probability that a movement causes an introduction from hospital A to hospital B. See Donker T, Wallinga J, Grundmann H. (2010) <doi:10.1371/journal.pcbi.1000715> for more details. prob_params[1] is the rate of acquisition in hospital A (related to LOS in hospital A). Default: 0.0036 prob_params[2] is the rate of loss of colonisation (related to time between admissions). Default: 1/365 prob_params[4] is the rate of transmission to other patients in hospital B (related to LOS in hospital B). Default: 0.128
`noloops`	(boolean). Should transfers within the same nodes (loops) be kept or set to 0. Defaults to TRUE, removing loops (setting matrix diagonal to 0).

`nmoves_threshold`
: (numeric) A threshold for the minimum number of subject transfer between two facilities. Set to NULL to deactivate, default to NULL.

`flag_vars`
: (list) Additional variables that can help flag a transfer, besides the dates of admission and discharge. Must be a named list of two character vectors which are the names of the columns that can flag a transfer: the column that can flag a potential origin, and the column that can flag a potential target. The list must be named with "origin" and "transfer". Eg: list("origin" = "var1", "target" = "var2"). See details.

`flag_values`
: (list) A named list of two character vectors which contain the values of the variables in flag_var that are matched to flag a potential transfer. The list must be named with "origin" and "transfer". The character vectors might be of length greater than one. Eg: list("origin" = c("value1", "value2"), "target" = c("value2", "value2")). The values in 'origin' and 'target' are the values that flag a potential origin of a transfer, or a potential target, respectively. See details.

`create_MetricsTable`
: (boolean) Should the metrics table be created along with the network. Setting to FALSE will speed up the results. Default is TRUE.

`verbose`
: TRUE to print computation steps

`shinySession`
: (NULL) internal variable to deal with the progress bar

`...`
: Additional parameters to be sent to checkBase in case the database has not been checked yet.

Details

This function will build a HospiNet object from a line-listed subject database. The HospiNet object has all of the functions stored as active bindings which can be accessed in the usual way. For more info, see `HospiNet`. Note that the subject database will need to be run through `checkBase` before going into this function.

Value

The function returns a HospiNet object.

See Also

`HospiNet`

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
hospinet_from_subject_database(myBase)
```

`matrix_from_base` *Compute the adjacency matrix of a network from a database of movements records.*

Description

This function computes the adjacency matrix of a network of facilities across which subjects can be transferred. The matrix is computed from a database that contains the records of the subjects' stays in the facilities. This function is a simple wrapper around the two functions `edgelist_from_base`, which computes the edgelist of the network from the database, and `matrix_from_edgelist`, which converts the edgelist into the adjacency matrix.

Usage

```
matrix_from_base(
  base,
  window_threshold = 365,
  count_option = "successive",
  prob_params = c(0.0036, 1/365, 0.128),
  condition = "dates",
  noloops = TRUE,
  nmoves_threshold = NULL,
  flag_vars = NULL,
  flag_values = NULL,
  verbose = FALSE
)
```

Arguments

`base` (data.table) A database of records of stays of subjects in facilities. The table should have at least the following columns:

- **subjectID** (character) unique subject identifier
- **facilityID** (character) unique facility identifier
- **admDate** (POSIXct) date of admission in the facility
- **disDate** (POSIXct) date of discharge of the facility

`window_threshold` (integer) A number of days. If two stays of a subject at two facilities occurred within this window, this constitutes a connection between the two facilities (given that potential other conditions are met).

`count_option` (character) How to count connections. Options are "successive", "probability" or "all". See details.

`prob_params` (vector of numeric) Three numerical values to calculate the probability that a movement causes an introduction from hospital A to hospital B. See Donker T, Wallinga J, Grundmann H. (2010) <doi:10.1371/journal.pcbi.1000715> for more details. For use with count_option="probability". prob_params[1] is the

rate of acquisition in hospital A (related to LOS in hospital A). Default: 0.0036 prob_params[2] is the rate of loss of colonisation (related to time between admissions). Default: 1/365 prob_params[4] is the rate of transmission to other patients in hospital B (related to LOS in hospital B). Default: 0.128

`condition` (character) Condition(s) used to decide what constitutes a connection. Can be "dates", "flags", or "both". See details.

`noloops` (boolean). Should transfers within the same nodes (loops) be kept or set to 0. Defaults to TRUE, removing loops (setting matrix diagonal to 0).

`nmoves_threshold` (numeric) A threshold for the minimum number of subject transfer between two facilities. Set to NULL to deactivate, default to NULL.

`flag_vars` (list) Additional variables that can help flag a transfer, besides the dates of admission and discharge. Must be a named list of two character vectors which are the names of the columns that can flag a transfer: the column that can flag a potential origin, and the column that can flag a potential target. The list must be named with "origin" and "transfer". Eg: list("origin" = "var1", "target" = "var2"). See details.

`flag_values` (list) A named list of two character vectors which contain the values of the variables in flag_var that are matched to flag a potential transfer. The list must be named with "origin" and "transfer". The character vectors might be of length greater than one. Eg: list("origin" = c("value1", "value2"), "target" = c("value2", "value2")). The values in 'origin' and 'target' are the values that flag a potential origin of a transfer, or a potential target, respectively. See details.

`verbose` TRUE to print computation steps

Details

The edgelist contains the information on the connections between nodes of the network, that is the movements of subjects between facilities. The edgelist can be in two different formats: long or aggregated. In long format, each row corresponds to a single movement between two facilities, therefore only two columns are needed, one containing the origin facilities of a movement, the other containing the target facilities. In aggregated format, the edgelist is aggregated by unique pairs of origin-target facilities. Thus, each row corresponds to a unique connection between two facilities, and the table contains an additional variable which is the count of the number of movements recorded for the pair. If the edgelist is provided in long format, it will be aggregated to compute the matrix.

Value

A square matrix, the adjacency matrix of the network.

See Also

`edgelist_from_base`, `matrix_from_edgelist`

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
matrix_from_base(myBase)
```

`matrix_from_edgelist` *Compute the adjacency matrix of a network from its edgelist*

Description

Compute the adjacency matrix of a network from its edgelist

Usage

```
matrix_from_edgelist(
  edgelist,
  origin_name = "origin",
  target_name = "target",
  count,
  format_long = FALSE
)
```

Arguments

`edgelist`	(data.table) A table containing the edges (or links) of the network, i.e. representing the movements of subjects between facilities. Either in long format with at least two columns (origin and target facilities of a link), each row corresponding to a single movement, or aggregated by unique pairs of origin/target, therefore with an additional variable for movements count (default). See details.
`origin_name`	(character) Column of the origin facilities of the links.
`target_name`	(character) Column of the target facilities of the links.
`count`	(character) Column of the counts of movements by unique pair of facilities.
`format_long`	(logical) Whether the edgelist is in long format, with each row corresponding to a single movement. If TRUE, the edgelist will be aggregated by unique pairs of facilities to compute the matrix.

Details

The edgelist contains the information on the connections between nodes of the network, that is the movements of subjects between facilities. The edgelist can be in two different formats: long or aggregated. In long format, each row corresponds to a single movement between two facilities, therefore only two columns are needed, one containing the origin facilities of a movement, the other containing the target facilities. In aggregated format, the edgelist is aggregated by unique pairs of origin-target facilities. Thus, each row corresponds to a unique connection between two facilities, and the table contains an additional variable which is the count of the number of movements recorded for the pair. If the edgelist is provided in long format, it will be aggregated to compute the matrix.

Value

A square numeric matrix, the adjacency matrix of the network.

See Also

edgelist_from_base, matrix_from_base

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
hospinet <- hospinet_from_subject_database(myBase)
matrix_from_edgelist(hospinet$edgelist, count = "N")
```

---

per_facility_summary *Function that extracts summary statistics from entire database*

Description

Function that extracts summary statistics from entire database

Usage

```
per_facility_summary(base, verbose = FALSE, ...)
```

Arguments

base	(data.table). A subject discharge database, in the form of a data.table. The data.table should have at least the following columns: sID: subjectID (character) fID: facilityID (character) Adate: admission date (date) Ddate: discharge date (date)
verbose	(boolean) print diagnostic messages. Default is TRUE.
...	other parameters passed on to internal functions

Value

a data table with one row per facility, showing mean LOS, number of subjects, and number of admissions

Examples

```
mydb <- create_fake_subjectDB(n_subjects = 100, n_facilities = 10)
myBase <- checkBase(mydb)
per_facility_summary(myBase)
```

Index