Package 'GUIgems'

May 18, 2017

Type Package

Title Graphical User Interface for Generalized Multistate Simulation Model

Version 0.1

Date 2017-04-30

Author Zofia Baranczuk, Janne Estill, Nello Blaser, Luisa Salazar Vizcaya, Olivia Keiser

Description A graphical user interface for the R package Gems.
Apart from the functionality of Gems package in the Graphical User interface, GUIgems allows adding states to a defined model, merging states for the analysis and plotting progression paths between states based on the simulated cohort.
There is also a module in the GUIgems which allows to compare costs and QALYs between different cohorts.

Maintainer Zofia Baranczuk <zofia.baranczuk@math.uzh.ch>

Imports rpanel, igraph, ggplot2, plyr, tools, msm, MASS, stringr, methods, utils, graphics, stats

License GPL-3

RoxygenNote 6.0.1.9000

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-18 14:14:46 UTC

R topics documented:

`addStateMatrix` *addStateMatrix*

Description

Adds a new state to a transitions functions matrix.

Usage

```
addStateMatrix(newMatrix, oldMatrix,
               stateName=paste("state", after+1, sep=" "),
               after=dim(oldMatrix)[1], newCell="NULL")
```

Arguments

`newMatrix`	an empty matrix with one more row and one more column than the old
`oldMatrix`	a matrix to which we are adding a new state;
`stateName`	the name of the state to be added;
`after`	the number of the state after which a new state should be added
`newCell`	the value in the state to be added;

Value

`newMatrix` a matrix created by adding a new state with the value `newCell` to the old matrix

Examples

```
M <- generateHazardMatrix(5)
largeM <- generateHazardMatrix(6)
oldM <- M@list.matrix
newM <- largeM@list.matrix
M_addedState <- addStateMatrix(newM, oldM, stateName ="newState", after =3, newCell=0)
```

ArtCohort *Class* "ArtCohort"

Description

TestIs a S4 class for the artificial cohort generated by simulateCohort.

Arguments

x	object an ArtCohort i, j, drop same as for data.frame ... passed on to data.frame method newsize size of the updated cohort addbaseline baseline for new part of cohort newInitialStates initial states for new part of cohort

Slots

states.number Object of class "numeric": number of states size Object of class "numeric": cohort size baseline Object of class "matrix": baseline matrix follow.up Object of class "numeric": maximum follow-up time parameters Object of class "transition.structure": input parameters parametersCovariances Object of class "transition.structure": input covariance matrices time-ToTransition Object of class "matrix": input timeToTransition matrix. logical components transitionFunctions Object of class "transition.structure": input hazard functions time.to.state Object of class "data.frame": entry times for each patient into each of the states

Objects from the Class

Objects are created by calls to the function simulateCohort.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

simulateCohort, transition.structure-class, transitionProbabilities, cumulativeIncidence

Examples

```
showClass("ArtCohort")
```

countCost *Counts a cost of the interventions in the cohort*

Description

Counts a cost of the intervention based on the cost data and the times spent in each state.

Usage

```
countCost(cohort, cost)
```

Arguments

cohort	a cohort simulated in simulateCohort()
cost	a list of lists:stateCost, units and qalys defining costs of basic units of cost, cost per state expressed in basic units and QALY for each state.

Value

`c("qaly","cost")` `qaly` an average QALY for an individual in the cohort `cost` an average cost for an individual in the cohort

cumulativeIncidence *Cumulative Incidence*

Description

Calculates the cumulative incidence and prediction intervals after first state

Usage

```
cumulativeIncidence(object, times, M = 100)
```

Arguments

object	either the output of simulateCohort or the matrix with the probabilities slot of that output.
times	a time vector.
M	number of groups for calculating confidence intervals.

Value

an object of class "PosteriorProbabilities", containing the statenames, timepoints and the cumulative incidence with pointwise prediction intervals over time.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

PosteriorProbabilities, ArtCohort, simulateCohort

---

generateHazardMatrix *generate template for transition functions*

Description

This function simplifies generating the matrix of transition functions.

Usage

```
generateHazardMatrix(statesNumber, statesNamesFrom = paste("State",
  1:statesNumber), statesNamesTo = statesNamesFrom)
```

Arguments

statesNumber	the number of states to be considered.
statesNamesFrom	names of the states in the model.
statesNamesTo	names of the states in the model.

Value

a transition.structure of dimension $N \times N$, where N is the number of states and with value "impossible" for all potential transitions.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer, Zofia Baranczuk

See Also

transition.structure-class, simulateCohort

generateParameterCovarianceMatrix
generate a template for parameter covariances

Description

This function simplifies generating the matrix of parameter covariances from a matrix of mean parameters.

Usage

```
generateParameterCovarianceMatrix(muM,
  statesNamesFrom = (dimnames(muM@list.matrix)["from"])[[1]],
  statesNamesTo = statesNamesFrom)
```

Arguments

muM	a transition.structure of dimension $N \times N$, whose components list the mean values for the parameters in the transitionFunctions.
statesNamesFrom	a list of names of states in the model. By default list(State 1, State 2, ..., State N)
statesNamesTo	a list of names of states in the model. By default list(State 1, State 2, ..., State N)

Value

a transition.structure of dimension $N \times N$ of covariance matrices for the parameters.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

transition.structure, generateParameterMatrix, simulateCohort

generateParameterMatrix *generate a template for mean parameters*

Description

This function simplifies generating the matrix of mean parameters from a matrix of transition functions.

Usage

```
generateParameterMatrix(hf,
  statesNamesFrom = (dimnames(hf@list.matrix)["from"])[[1]],
  statesNamesTo = statesNamesFrom)
```

Arguments

hf	a transition.structure of dimension $N \times N$, where N is the number of states.
statesNamesFrom	- a list of names of states in the model. By default list(State 1, State 2, ..., State N)
statesNamesTo	- a list of names of states in the model. By default the same as statesNamesFrom

Value

a transition.structure of dimension $N \times N$, whose components are lists of the right length for the parameters in the corresponding hazard function hf.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

transition.structure-class, simulateCohort

graphHf *graphHf*

Description

Plots the possible progression paths in the stochastic model based on the transition matrix. The endges in the graph exist, if there is a transition function between given two states.

Usage

```
graphHf(hfNames, statesNames = rownames(hfNames))
```

Arguments

hfNames	a matrix with names of hazard functions;
statesNames	a list of names of the states. By default rows names of HfNames

Value

ghf - a graph with statesNames as nodes. There are egdes between states, where the hazard function is not "impossible" nor "NULL"

Examples

```
hfNames <- array(rep("Exponential", 36), dim = c(6,6))
hfNames[col(hfNames)<=row(hfNames)]<-"NULL"
hfNames[3,4:5] <-  rep("impossible",2)
graphHf(hfNames)
```

makeM *makeM*

Description

Builds hazard function matrix based on the hazard function names matrix. If some function is not defined, the name itself is copied to M matrix.

Usage

```
makeM(hfNames)
```

Arguments

hfNames	- a matrix with names of hazard functions. One can use predefined function names: "Impossible", "Exponential", "Weibull", "MultWeibull"

Value

matrix M generated by generateHazardMatrix with hazard functions, which names are in hfNames matrix.

Examples

```
hfNames <- array(rep("Exponential", 36), dim = c(6,6))
hfNames[3,4:5] <-  rep("impossible",2)
hfNames[1:4,6] <- rep("sr.fun", 4)
hfNames[2, 4:5] <- rep("treat.fun", 2)
hfNames[col(hfNames)<=row(hfNames)]<-"NULL"
M <- makeM(hfNames)
```

---

mergeStates *mergeStates*

Description

Merges the states given by the user. The names of merged states are concatenations of names of the states being merged. `merged_cohort\@ time.to.state` is defined as time, when the patient arrives in the first state which was merged into the new state. The function creates also a `hfNames matrix` for the merged states. They are used only to plot a graph of possible transmissions between merged states.

Usage

```
mergeStates(cohort, statesGroups)
```

Arguments

`cohort`	a `cohort` simulated by `simulateCohort()`
`statesGroups`	a `list of lists` of states to be merged. e.g. `list(list(1,2,3),list(5,6))`

Value

list(cohortGroup, hfNames) `cohortGroup` is an object with a structure like a `cohort` simulated by `simulateCohort`. States names are now concatentations of the names of the states which were merged. Times of transitions (cohortGroup (at) time.to.state) are recomputed for groups of merged states. The other attributes of the cohorts stay unchanged. `hfNames` is a `matrix` with row names beeing concatenated names of the states that were merged, the values in the matrix are "impossible", if the transition between given groups of states is impossible, or "possible", if there is some possible transition between the states in the given two groups of states.

See Also

simulateCohort

Examples

```
hfNames <- array(rep("Exponential", 16), dim = c(4,4))
hfNames[col(hfNames) <= row(hfNames)]<-"NULL"
rownames(hfNames) <- as.list(paste("state", 1:4))
colnames(hfNames) <- as.list(paste("state", 1:4))
M <- makeM(hfNames)
param <- generateParameterMatrix(M)
param[[1,2]] <- list(rate = 1)
param[[1,3]] <- list(rate = 2)
param[[2,3]] <- list(rate = 0.5)

param[[1,4]] <- list(rate = 1)
param[[2,4]] <- list(rate = 2)
param[[3,4]] <- list(rate = 0.5)

cohort <- simulateCohort(
transitionFunctions = M,
parameters = param,
cohortSize = 100,
to=10)

tmp <- mergeStates(cohort, statesGroup = list(list(1,2), list(3,4)))
merged_cohort <- tmp[["cohort"]]
merged_hfNames <- tmp[["hfNames"]]
```

---

plot,PosteriorProbabilities-method

plot

Description

Plots S4 objects

Usage

```
## S4 method for signature 'PosteriorProbabilities'
plot(x, ci = FALSE, main = paste(x@type,
  "after starting in State", x@states[1], "at time 0"),
  states = 1:dim(x@probabilities)[2], lwd = c(2, 2), col = c("blue",
  "green3"), lty = c(1, 2), xlab = "Time", ylab = "Probability", ...)
```

Arguments

x	a cohort simulated by simulateCohort.
ci	a binary variable, cumulative incidence
main	same as in any plot

states	names of states in the plot
lwd	vector of lentht 2, with first component for the point estimate and second component for the confidence interval
col	vector of lentht 2, with first component for the point estimate and second component for the confidence interval
lty	vector of lentht 2, with first component for the point estimate and second component for the confidence interval
xlab	same as in any plot
ylab	same as in any plot
...	arguments passed on to main method

---

plotProgressionPath *plotProgressionPath*

Description

Plots f-th most frequent progression path in the cohort

Usage

```
plotProgressionPath(cohort, hf, f)
```

Arguments

cohort	simulated in simulateCohort()
hf	names of the transition functions between states in the cohort.
f	number of the path to be plotted

Examples

```
hfNames <- array(rep("Exponential", 9), dim = c(3,3))
hfNames[col(hfNames)<=row(hfNames)]<-"NULL"
colnames(hfNames) <- as.list(paste("state", 1:3))
rownames(hfNames) <- as.list(paste("state", 1:3))

M <- makeM(hfNames)
par <- generateParameterMatrix(M)
par[[1,2]] <- list(rate = 1)
par[[1,3]] <- list(rate = 2)
par[[2,3]] <- list(rate = 0.5 )

cohort <- simulateCohort(
transitionFunctions = M,
parameters = par,
cohortSize = 100,
to=10)

plotProgressionPath(cohort, hfNames, 2)
```

PosteriorProbabilities *Class* "PosteriorProbabilities"

Description

This S4 class summarizes the posterior probabilities over time for objects of class "ArtCohort"

Slots

`states` Object of class "character": names of states

`times` Object of class "numeric": time points at which probabilities are evaluated

`probabilities` Object of class "matrix": posterior Probabilities to be in each state at each time

`lower` Object of class "matrix": lower prediction bound to be in each state at each time

`upper` Object of class "matrix": upper prediction bound to be in each state at each time

`type` Object of class "character": describes type of probability

Objects from the Class

Objects are created by calls to the function `simulateCohort`.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

`transitionProbabilities`, `cumulativeIncidence`, `ArtCohort`

Examples

```
showClass("PosteriorProbabilities")
```

simulateCohort *Simulate cohort*

Description

Simulates a cohort of patients from a set of functions associated to each possible transition in a multistate model. The multistate model is not required to be a Markov model and may take the history of previous events into account. In the basic version, it allows to simulate from transition-specific hazard function, whose parameters are multivariable normally distributed. For each state, all transition-specific hazard functions and their parameters need to be specified. For simulating one transition, all possible event times are simulated and the minimum is chosen. Then simulation continues from the corresponding state until an absorbing state of time to is reached.

Usage

```
simulateCohort(transitionFunctions, parameters, cohortSize = 1000,
  parameterCovariances = generateParameterCovarianceMatrix(parameters),
  timeToTransition = array(FALSE, dim = dim(transitionFunctions@list.matrix)),
  baseline = matrix(NA, nrow = cohortSize),
  baselineFunction = baselineFunction_empty, initialState = rep(1,
  cohortSize), absorbing = transitionFunctions@states.number, to = 100,
  report.every = 100, sampler.steps = 1000)
```

Arguments

transitionFunctions	a transition.structure of dimension $N \times N$ that contains the hazard functions element corresponds to a transition number, 0 if transition does not occur
parameters	a transition.structure of dimension $N \times N$ that contains the parameters
cohortSize	a numeric indicating the number of patients to be simulated.
parameterCovariances	a transition.structure of dimension $N \times N$ of covariance matrices for the parameters.
timeToTransition	a logical matrix; TRUE for all transitions whose transitionFunction is specified as the time until transition instead of as a hazard function or as a character.
baseline	a matrix or data.frame of dimension $cohortSize \times M$ with M baseline characteristics of subjects to be simulated.
baselineFunction	a function generating a data.frame of dimension $cohortSize \times M$ with M baseline characteristics of subjects to be simulated.
initialState	a numeric of length cohortSize with the initial state for each subject simulated.
absorbing	a numeric containing all absorbing states.
to	final time of the simulation.

report.every	a numeric to check progress of simulation.
sampler.steps	a numeric indicating number of steps for discretization of hazard functions

Details

The transitionFunctions contains hazard functions or time to event function associated to each possible transition. The elements of this list can be either expressed as an explicit R function or as a character ("impossible", "Weibull", "multWeibull", "exponential") in order to express impossible transitions or parametric forms for the distributions of time to event. If the functions should depend on time, baseline characteristics or be *history-dependent*, the function arguments *t*, *bl* or *history* can be used. Time *t* refers to the time since entry into the current state. For the time since the initial state, use t+sum(history).

The components of the parameters argument list the mean values for the parameters in the transitionFunction. If the corresponding transitionFunction is a function, the parameters should appear in the same order as in the function, leaving out *t*, *bl* and *history*. If the corresponding transitionFunction is the character "Weibull", the first argument is the shape and the second one the scale. If the corresponding transitionFunction is the character "multWeibull", specify weights, shapes, scales in this order.

Note that when using the parameterCovariances argument it is the users responsibility to ensure that the functions are parametrized such that parameters for each transition are multivariate normally distributed and mutually independent.

Value

an object of class "ArtCohort" with time.to.state slot of dimension $cohortSize \times N$ with entry times for each patient into each of the states.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

generateHazardMatrix,

generateParameterMatrix,

generateParameterCovarianceMatrix,

ArtCohort, transitionProbabilities,

cumulativeIncidence

Examples

```
# Here is an example model with 3 states and 2 possible transitions.

# number of states in the model
statesNumber <- 3

# cohort size
```

```
cohortSize <- 100

# specification of hazard functions
hazardf <- generateHazardMatrix(statesNumber)
hazardf[[1,2]] <- function(t, r1, r2)
{
  ifelse(t<=2, r1 , r2)
}
hazardf[[2,3]] <- "Weibull"

# list of parameters for the hazard functions
mu <- generateParameterMatrix(hazardf)
mu[[1,2]] <- list(r1=0.33,  r2=0.03) # r1, r2
mu[[2,3]] <- list(shape=1,scale=0.84) # shape, scale

# time
maxTime <- 10

# simulate the cohort
cohort <- simulateCohort(
  transitionFunctions = hazardf,
  parameters = mu,
  cohortSize = cohortSize,
  to=maxTime)

# output
head(cohort)

# transition probability
tr <- transitionProbabilities(cohort, times=seq(0,4,.1))
plot(tr, ci=FALSE)

# cumulative incidence
inc <- cumulativeIncidence(cohort, times=seq(0,4,.1))
plot(inc, ci=FALSE, states=c(2,3))
```

`start_gui` *Starts graphical user interface for gems*

Description

This function start rp.panel-based GUI for the package "gems". The GUI can be used to define generalized multistate simulation model, run simulations for this model and analyze simulated cohorts.

start_gui() opens the basic panel of the Graphical User Interface for "gems". This basic panel of GUI_gems is used to define the model for which the cohort will be simulated. I. The basic panel - defining the model.

In GUIgems, there are two ways to define the model for the simulation. One method is loading a pre-defined model, which has been written previously in an R script. There in an examples of pre-defined models in the demo folder: `model.R`

The second method is to define the model directly within the GUI, it can be saved and reused later.

The main part of this panel is a matrix with states and transitions between states. Only transitions above the diagonal are allowed to assure no cycles in the model progression. One can edit names of transition function using the `Edit hf matrix`, It opens a separate window with hazard functions matrix.

There is a menu for every allowed transition. One can choose one of the following options:

1. Choose the transition function for the given pair of states. This lets the user change the defined transition function from state "i" to state "j" to one of the predefined functions: "impossible", "Weibull", "multWeibull", "exponential"

2. Edit the parameters for the transition function. par_i_j The table with the parameters for the current transition functions will be opened in the editor. In the same table there is a place to set the covariance of parameters. By default, it is set to 0. If parameters weren't previously defined by the user, the paremeters for the transition function are set to the default value of the parameter (if it exists) or to 0.

Apart from defining the transition matrix, the user can:

`Set the number of states` It creates a new, empty (all transitions impossible) model with the given number of states.

`Set <start rows> and <start columns>` If the model has more than 12 states, it is impossible to see the whole matrix. This way the user can choose which part of the transition matrix he/she wants to see. It sets the first row and the first column to be shown.

`Set <Cohort size>` Sets the size of the cohort to be simulated.

`Add new state with a given <state_name>, <after> a given state number` The user can write in the textentires the name of the state to be added and the state number, after which the new state should be added. This is possible only, when states after the state to be added do not have history as the parameter in their transition functions.

`Load hazard functions` Opens a dialog box so that the user can choose a directory. Source() will be called on all the files in the directory. All the functions from the files in the directory will be loaded into the current environment.

`Save the model` Save the currently defined model using the dialog box. Important - at this level only model is saved, even if the cohort was simulated. For saving the simulated cohort, save the model from the cohort panel.

`Load baseline` Opens a dialogbox to choose a file. Source() is called on the chosen file. The function from the file will be used to generate baseline with the lenght of cohort size.

`Simulate Cohort` Simulates the cohort for the model described in the panel using the function `simulateCohort()`. After the cohort is simulated, `the basic cohort panel` is opened. See part II.

`Load model` Loads an .RData file or runs the .R file chosen by the user in a dialog box. The file should either be a model saved in Gems session as an .RData file or be an .R file inluding variables: `hf` - names of the hazard functions or `M` - hazard functions defined directly without a name; `parameters` - a matrix generated by `generateParametersMatrix(M)` with lists of names parameters for the hazard functions defined in `hf` or `M`; `cohortSize` - a number of items to be simulated,

It may also include variables:

covariances - a matrix generated using

generateParametersCovariances(parameters). If covariances matrix is not set in the file, all covariances are set to 0. bl_function - a function which will be used to generate a data frame with baseline or baseline - a data frame with baseline.

Loaded model can also have a simulated cohort. Then after loading the model, the basic cohort panel will be opened. See part II.

There are examples of defined models and hazard functions in the directory demo.

Update panel rp.panel after changing a parameter gets refreshed first after the next action. Update panel refreshes the rp.panel and shows current values of variables.

Multiple cohorts This button opens the multiple cohorts panel. See part IV.

Edit ttt matrix Edits mypanel$ttt matrix - the matrix with information about which hazard functions are time to transition functions. mypanel$ttt gets changed.

Edit states names User can edit mypanel$statesNames.

Edit hf matrix User can edit transition functions matrix.

Plot hf graph Plots a graph with all possible transitions between states. Here "possible" means, that the transition between two states is allowed (above the diagonal in the matrix) and not set to "impossible". Compare with plotPath_n() - the function in the cohort panel and the merged cohort panel to plot paths that were present in the simulations.

II. The cohort panel. The basic cohort panel is opened after simulating a cohort or loading a model with a simulated cohort. The panel is used for running analysis on the cohort. There are following actions possible to run in this panel:

Edit Cohort Shows the baseline and the transition times between states in the cohort.

List subcohorts Shows the list of all subcohorts in the model with the attached numbers, including the main subcohort (=the whole simulated cohort), the used defined subcohorts and subcohorts following different progression paths, if the paths were plotted before using the button plot paths.

List states Shows the list of all states in the model

Save the model Opens the save dialog box to choose the name to save the model with the simulated cohort.

Plot paths Simulated elements of the cohorts can follow different progression paths (go through different states). Plot paths marks on the graph of possible transitions the most frequent progression path in the cohort. The less frequent paths may be shown using plot next path

Plot next paths Marks on the graph of possible transitions next most frequent progression path.

Add subcohort <subcohort query> Creates a new subcohort according to the <subcohort query> in the text entry. Subcohorts can be chosen based on the constraints about baseline characteristics or the simulated progression path and times of transmissions.

Examples of the input into the text entry to define a new subcohort:

- (cohortBl["sex"] ==1) & (cohortBl["age"]>40) will return a subcohort of people with gender 1 older than 40.
- is.na(cohortBl["HCC"]) will create a subcohort with simulated patients, who never got into "HCC" state.

`View subcohort <subcohort number>` Shows using View() the subcohort with the number defined in the text entry. The numbers of cohorts with their definitions can be seen after pressing the `list cohorts` button.

`Plot <function name> <states> <subcohorts>`

`Function name` User can choose from the menu one of the functions to be plotted:

- `transitionProbabilities`
- `cumulativeIncidence`

see: `?transitionProbabilities`, `?cumulativeIncidence`

`states` A list of states, for which the chosen function will be plotted.

Eg. `list(1:4)`, `list(c(1,3,5))`. If there is more than 6 states chosen, the plots will be opened in different windows.

`subcohorts` A list of subcohorts to be plotted.

Eg. `list(1:3)`, `list(c(1,3))`

User can check the numbers of defined subcohorts by clicking the button `List subcohorts`

`States groups <list of list of states to be merged>` This option let merging different states of the model. From the group of states to be merged, the time of entry to the earliest of the states counts as the entry time to the meta-state created from this group of states. The time of leaving of the last state from the group counts as the time of leaving the new meta-state. The name of the new state is a convolution of the names of the states composing it connected by a hyphen.

Not all groups of states may be merged - in order to assure lack of cycles, the states to be merged must be sequent ones. States to be merged must be given in the form of a list of lists.

Eg. `list(list(1,2), list(3,4))` will merge the states 1 and 2 into one state, and states 3 and 4.

III. Cost analysis module:

1. Edit cost per unit: User can define in the editor costs per unit during the treatment. Default units are: `visit`, `test`, `medicine`, `fibroscan` The user can define more cost units or change the current costs.

2. Edit QALY per state: User can modify in the editor QALYs for each state of the model.

3. Edit cost per state: User can define cost of the treatment for each state of the model expressed in cost units in a time unit. Eg. `3*medicine + 4* visit + fibroscan`

4. Save cost: Saves cost data in the form cost_list =list(units,qualy,StateCost), where units, qualy and StateCosts are lists.

5. Load cost: Loads the cost from the .RData file in the form as described in `Save cost`. 6. Count cost: It counts cost and QALY for the given cohort. Units cost multiplied by the averaged time in a given state are put into the equations in `cost per state`. At the level of one cohort, the cost and the QALYs are only shown in the new tab of R. For a plot of cost vs. QALY, see the cost analysis in the `multiple cohort panel`.

IV. The merged cohort panel. The `merged cohort panel` is used to analyse the cohort after merging states. There are possible actions to be chosen:

`Edit cohort` Shows the cohort in the new R Tab

`List subcohorts` Shows the list of all subcohorts in the model with the attached numbers, including the main subcohort (=the whole simulated cohort) and subcohorts following different progression paths, if the paths were plotted before (using the button `plot paths`)

`List states` Shows the list of all states in the model.

`Plot paths` Simulated elements of the cohorts can follow different progression paths. `Plot paths` marks on the graph of possible transitions the most frequent progression path in the cohort. The less frequent paths may be shown using `plot next path`

`Plot next paths` Marks on the graph of possible transitions next most frequent progression path.

`Add subcohort <subcohort query>` Creates a new subcohort according to the <subcohort query> in the text entry. Subcohorts can be chosen based on the constraints about baseline characteristics or the simulated progression path and times of transmissions.

Examples of the input into the text entry to define a new subcohort:

- `(cohortBl["sex"] ==1) & (cohortBl["age"]>40)` will return a subcohort of people with gender 1 older than 40.
- `is.na(cohortBl["HCC"])` will create a subcohort with simulated patients, who never got into "HCC" state.

`View subcohort <subcohort number>` Shows using View() the subcohort with the number defined in the text entry. The numbers of cohorts with their definitions can be seen after pressing the `list cohorts` button.

V. The multicohort panel. While working with multiple cohorts, the user has similar options as with multiple subcohorts. Different is adding a new cohort. The user can also analyse the cost for different cohorts.

`Add cohort` Adds a new cohort created by `simualteCohort` from the .RData file chosen by the user. Cohort's name is then the full path of the file with the cohort.

`Edit cohort names` User can change in the editor names of the cohorts.

`View cost` Shows the cost data - unit costs, QALYs for each state and the costs expressed in units for each state.

`View cohort <cohort number>` Shows the cohort with the <cohort number>

`List cohorts` Shows the list of all cohorts in the model with the attached numbers

`Plot <function name> <states> <cohorts>` The same as plotting subcohorts in the cohort panel.

The user can check the numbers of defined cohorts by clicking the button `List cohorts`

Usage

```
start_gui() # start the rp.panel  with GUI for the package "gems"
```

Author(s)

Zofia Baranczuk

timeInStates — *Builds hazard function matrix based on the hazard function names matrix.*

Description

Builds hazard function matrix based on the hazard function names matrix.

Usage

```
timeInStates(mydata, end_time = 100)
```

Arguments

mydata	- cohort@time.to.state created in simulateCohort()
end_time	- time of the simulation

Value

time spent in each of the states by each simulated unit

transition.structure — *Class* "transition.structure"

Description

This S4 class provides a structure to specify different characteristics of transitions, such as transition functions functions, parameters or parameter covariances.

Objects from the Class

Objects are created by calls to the functions `generateHazardMatrix`, `generateParameterMatrix`, `generateParameterCovarianceMatrix`.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

`generateHazardMatrix`, `generateParameterMatrix`, `generateParameterCovarianceMatrix`

Examples

```
showClass("transition.structure")
```

`transitionProbabilities` *transition probabilities*

Description

Calculates the probabilities and prediction intervals after first state

Usage

```
transitionProbabilities(object, times, M = 100)
```

Arguments

`object`	either the output of `simulateCohort` or the `matrix` with the `probabilities` slot of that output.
`times`	a time vector.
`M`	number of groups for calculating confidence intervals.

Value

an object of class `"PosteriorProbabilities"`, containing the statenames, timepoints and the transition probabilities with pointwise prediction intervals over time.

Author(s)

Luisa Salazar Vizcaya, Nello Blaser, Thomas Gsponer

See Also

`PosteriorProbabilities`, `ArtCohort-class`, `simulateCohort`

Index