Package 'rdecision'

September 17, 2019

Title Decision Analytic Modelling In Health Economics

Version 0.1.5

Description A package for decision analytic modelling in health economics.
It includes functions for modelling healthcare interventions using
cohort models (decision trees, Markov models and extended Markov models).
It draws on terminology from Briggs, Claxton and Sculpher, ``Decision
Modelling for Health Economic Evaluation'', Oxford University Press, 2006.

Depends R (>= 3.1.0)

Imports R6

Suggests rmarkdown,
knitr

License GPL-3

LazyData true

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

R topics documented:

ChanceNode *An R6 class to represent a node in a decision tree*

Description

An R6 class for a chance node in a decision tree

Usage

```
ChanceNode
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`p` a list of proportions associated with each chance outcome

`edgelabels` a vector of character stings containing the labels of each chance outcome associated with the node

`costs` a vector of numeric values containing the costs associated with each choice

DecisionNode *An R6 class for a decision node in a decision tree*

Description

An R6 class for a decision node in a decision tree

Usage

```
DecisionNode
```

Format

An R6 class

Fields

`edgelabels` a vector of character stings containing the labels of each choice associated with the decision

`costs` a vector of numeric values containing the costs associated with each choice

des *Discrete event simulation solver of Markov models*

Description

des is solver function for the rdecision package for discrete event simulations. The function is based on the model described by Sonnenberg and Beck. This function solves a Markov state model using a Monte-Carlo method, based on random number generation.

Usage

```
des(nStates, nGroups = 0, nPatients, nCyclesPerYear, nCycles, state,
  group = NA, prevalence, Ip, Tp, Gp = NA, discount = 0, stub = NA,
  continue = F)
```

Arguments

nStates	Number of states in Markov model
nGroups	Number of groups (default zero); states can be grouped together and a maximum cycle limit set on group occupancy.
nPatients	Number of patients to simulate
nCyclesPerYear	Number of cycles per year
nCycles	Number of time cycles to simulate
state	Data frame of length nStates with named fields (see: State)
group	Data frame of length nGroups with named fields (see: Group)
prevalence	Array of length nStates which contains values for the initial state occupancy (prevalence). All values should be in the interval [0,1] and the sum of the elements should be 1.
Ip	Matrix of ANNUAL rates of transition between states. The dimensions are (nStates,nStates) with each entry being the annual rate of transitions from the row state to the column state. Values are in the range [0,1]. Transitions to self (i.e. leading diagonal) should NOT be included.
Tp	Matrix of dimension (nStates,nStates) which contain time independent probabilities of transitions from time limited states to other states after time expiry. Values are in the interval [0,1].
Gp	Matrix of dimension (nGroups, nStates) which contain time independent probabilities of transitions from time limited groups to other states after time expiry. Values are in the interval [0,1]. It is an error if any of the transitions from a group to a state within that group are > 0.
discount	Annual discount rate for costs and benefits as a percentage figure, eg 3.5. This is applied to each time cycle with log correction to convert from annual rate to per-cycle rate if required. Default 0.00.
stub	File stub to be used for output files, eg 'mymarkov' would cause files called 'mymarkov-log.txt' etc to be created. Default NA means no files are written.
continue	If no, states are populated according to the given prevalence; if yes, the simulation restarts in its previous state by reading in the csv files, and the prevalences are ignored. If stub is NA (file output is suppressed) continue=T is disallowed.

Value

A list of 4 matrices, for state populations, costs, entries and utilities.

State

Data frame `state` should have the following fields: `name` string which describes state (used in log output) `hasCycleLimit` value TRUE if cycle limit applies (ie temporary/tunnel state); FALSE otherwise (ie absorbing or normal state) `cycleLimit` (integer) maximum number of cycles for which a single patient can occupy the state, if hasCycleLimit = TRUE. For temporary states use hasCycleLimit=TRUE and cycleLimit=0. `annualCost` cost of being in the state for 1 year `entryCost` one-off cost associated with entering the state `utility` (incremental) utility of being in the state for one year `group` group number to which this state belongs (0 if no group, >= 1 otherwise)

Group

Date frame `group` should have the following fields: `number` (integer) group number as an integer (referenced by state$group); NA if not defined. `name` string which describes the group name (in output) `hasCycleLimit` value TRUE if cycle limit applies to the group, FALSE otherwise. `cycleLimit` maximum number of cycles for which a single patient can occupy the group.

Note

The following files will be created by the function: 'stub'-log.txt: a text file listing inputs, progress and final results of the simulation. 'stub'-pop.csv: a comma separated value file containing the population of each state at the end of each cycle. 'stub'-psp.csv: a comma separated value file containing the cumulative cost associated with entry and occupancy of each state after each cycle. 'stub'-ent.csv: a comma separated value file containing the cumulative number of entries into each state, after each cycle. 'stub'-utl.csv: a comma separated value file containing the cumulative utlity of each state, after each cycle.

References

Sonnenberg FA and Beck JR, "Markov models in medical decision making: a practical guide", Medical Decision Making 1993;13:322-339.

`fprintf` *A version of fprintf for R*

Description

`fprintf` writes a formatted string to the file open on connection `con`. It is intended to be equivalent to the Matlab and C function of the same name.

Usage

```
fprintf(con, fmt, ...)
```

Arguments

con	an R connection, opened with the open command or one its overloaded methods, such as file or url.
fmt	a format string, of the syntax accepted by the R function sprintf.
...	a list of arguments, matching those specified in the fmt argument,, to be written to the connection.

Value

The number of characters written to the connection.

See Also

sprintf

Examples

```
ofid <- file("marvin.txt", open='wt')
fprintf(ofid, "The answer is %i\n", 42)
close(ofid)
```

LeafNode *An R6 class for a leaf node in a decision tree*

Description

An R6 class for a leaf node in a decision tree

Usage

```
LeafNode
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`pathway` character string; a label for the leaf node which is synonymous with the name of the root-to-leaf pathway

`utility` the preference or value that a user associates with a given health state (tange 0 to 1).

`MarkovModel` *An R6 class for a Markov model*

Description

An R6 class for a Markov model

Usage

```
MarkovModel
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`states` a list of objects of type MarkovState

`Ip` Matrix of *annual* rates of transition between states. The dimensions are (nStates,nStates) with each entry being the annual rate of transitions from the row state to the column state. Values are in the range [0,1]. Transitions to self (i.e. leading diagonal) should be set to zero, and will be adjusted so that the sum of transitions from each row are zero. The matrix should have row names and column names which are the state names.

Methods

cycle Applies one cycle of the model.

cycles Applies `nCycles` cycles of the model. The starting populations are redistributed through the transition probabilities and the state occupancy costs are calculated, using function `cycle`. The end populations are then fed back into the model for a further cycle and the process is repeated. For each cycle, the state populations and the aggregated occupancy costs are saved in one row of the returned dataframe, with the cycle number.

getStatenames Returns a character list of state names

initialize Creates a Markov model. The parameters are `states`, a list of states; `Ip` the matrix of annual state transitions; `discount`, the annual discount rate (percentage); `nCyclesPerYear`, the number of cycles per year.

setPopulations Sets the occupancy of each state. Takes argument `populations` which is a named vector of populations for the start of the state. The names should be the state names. Due to R's implementation of matrix algebra, `populations` must be a numeric type and is not restricted to being an integer.

setTransitions Sets the annual state transition rates. Argument `Ip` must be a numeric square matrix of dimension equal to the number of states, with row and column names equal to Markov state names. Transition rates are from row state to column state.

stateSummary Creates a state summary data frame suitable for printing with kable etc.

transitionSummary Creates a table of the annual transition probabilities.

MarkovState *An R6 class for a state in a Markov model*

Description

An R6 class for a state in a Markov model

Usage

```
MarkovState
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`name` the name of the state (character string)

`hasCycleLimit` (logical) TRUE if the state is a tunnel state

`cycleLimit` (numeric) the maximum length of stay (1 for cohort tunnel states)

`entryCost` the cost to enter the state

`annualCost` the annual cost of state occupancy

Methods

setAnnualCost Sets the annual cost of state occupancy

setEntryCost Sets the entry cost of state occupancy

Node *An R6 class to represent a node in a decision tree*

Description

An R6 class to represent a node in a decision tree

Usage

```
Node
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`children` a list of child nodes.

path.apply — *An apply function for node-to-leaf paths in a decision tree.*

Description

path.apply traverses each node-to-leaf path in the tree, starting with @codenode, and applies function FUN to each list of nodes in each node-to-leaf path.

Usage

```
path.apply(node, FUN)
```

Arguments

node	first node in the tree to which FUN should be applied.
FUN	function to apply to each list of nodes starting with node and ending with one of the leaf nodes. The first agument to FUN must be a list of Nodes.

Value

A list of length equal to the number of node-to-leaf paths in the tree with each member of the list equal to the result of applying FUN to that list of node-to-leaf nodes. For example, if FUN=length, the return value will be a list of the node count of each node-to-leaf traversal of the tree.

pathway.choice — *Returns the choice label associated with the first decision node in a list of nodes.*

Description

Returns the name of the choice arising from the first decision node of a list of nodes. Intended to be used with path.apply to return a list of choices for each node-to-leaf traversal. In many decision trees, where the decision node is at the left, many leaf nodes (pathway names) will be associated with the same choice.

Usage

```
pathway.choice(nodes)
```

Arguments

nodes	list of objects of type 'Node'.

Value

label of the choice; specifically the edgelabel field of the first decision node in nodes.

Note

Uses the first decision node in the list.

pathway.cost *Calculate pathway cost from decision and chance nodes.*

Description

Calculates the sum of the pathways costs in the tree traversal.

Usage

```
pathway.cost(nodes)
```

Arguments

nodes	List of nodes in the node-to-leaf traversal path.

Note

Assumes that the final node is a leaf node.

pathway.name *Return leaf name from a leaf node in a list of nodes.*

Description

Returns the name of the pathway from the (final) leaf node of a list of nodes. Intended to be used with @codepath.apply to return a list of leaf node names for each node-to-leaf traversal.

Usage

```
pathway.name(nodes)
```

Arguments

nodes	list of objects of type 'Node'.

Value

name of pathway; specifically the @codepathway field of the final leaf node in @codenodes.

Note

Assumes the leaf node is the last node in the list.

`pathway.probability` *Calculate pathway probability from conditional probabilities.*

Description

Calculates the product of the conditional P values in the tree traversal.

Usage

```
pathway.probability(nodes)
```

Arguments

`nodes`	List of nodes in the node-to-leaf traversal path.

Note

Assumes the nodes are supplied in order of traversal and that the final node is a leaf node.

`pathway.utility` *Calculate pathway utility.*

Description

Calculates the sum of the pathway utilities in the tree traversal.

Usage

```
pathway.utility(nodes)
```

Arguments

`nodes`	List of nodes in the node-to-leaf traversal path.

Note

Assumes that only the final leaf node has a utility value.

`rdecision` *rdecision: Decision Analytic Modelling In Health Economics.*

Description

A package for decision analytic modelling in health economics. It includes functions for modelling healthcare interventions using cohort models (decision trees, Markov models and extended Markov models). It draws on terminology from Briggs, Claxton and Sculpher, "Decision Modelling for Health Economic Evaluation", Oxford University Press, 2006.

`which.node` *Find the index of a node object in a list of node objects*

Description

Returns the index of node in a list of nodes.

Usage

```
which.node(nodes, node)
```

Arguments

`nodes`	list of objects of type Node
`node`	node of type Node to search for in `nodes`

Value

index of first occurrence of `node` in `nodes` or NA if not present

Index