Package 'CEdecisiontree'

January 18, 2019

Type Package

Title Cost-effectiveness decision tree analysis

Version 0.1.0

Maintainer The package maintainer <ngreen1@ic.ac.uk>

Description Cost-effectiveness decision tree analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports assertthat,
Rcpp,
heemod,
readr,
dplyr,
reshape2,
tidyr

Suggests testthat,
knitr,
rmarkdown,
covr

VignetteBuilder knitr

LinkingTo Rcpp

BugReports https://github.com/Health-Economics-in-R/CEdecisiontree/issues

R topics documented:

`branch_joint_probs` *Branch Joint Probabilities*

Description

Provides a measure of the chances of following particular paths.

Usage

```
branch_joint_probs(probs)
```

Arguments

`probs`	Branch conditional probabilities (matrix)

Details

These probabilities could be used to weight branch costs or QALYs to indicate the relative contribution to the total expected value.

Value

transition matrix with joint probabilities

Examples

```
data(probs)
data(cost)
branch_joint_probs(probs) * cost
```

`Cdectree_expected_values` *Cdectree_expected_values*

Description

Cdectree_expected_values

Usage

```
Cdectree_expected_values(vals, p)
```

---

dectree_expected_recursive

Cost-effectiveness decision tree using recursive approach

Description

Cost-effectiveness decision tree using recursive approach

Usage

```
dectree_expected_recursive(node, tree, dat)
```

Arguments

`node`	Node at which total expected value is to be calculate at
`tree`	List of children by parents
`vals`	Node labels, branch probabililities and value; dataframe

Value

Expected value at root node

See Also

CEdecisiontree

Examples

```
tree <-
  list("1" = c(2,3),
       "2" =  c(4,5),
       "3" =  c(6,7),
       "4" =  c(),
       "5" =  c(),
       "6" =  c(),
       "7" =  c())
dat <-
  data.frame(node = 1:7,
             prob = c(NA, rep(0.5, 6)),
             vals = c(10,2,3,16,5,6,7))

root <- names(tree)[1]
dectree_expected_recursive(node = root, tree, dat)
```

dectree_expected_values

Cost-effectiveness decision tree expected values

Description

Root node expected value as the weighted mean of probability and edge/node values e.g. costs or QALYS.

Usage

```
dectree_expected_values(model, ...)

## Default S3 method:
dectree_expected_values(vals, p, dat = NA)
```

Arguments

`model`	List as `define_model()` output of type `tree_dat`, `transmat` or `dat_long`
`vals`	Values on each edge/branch e.g. costs or QALYs (array)
`p`	Transition probabilities matrix
`dat`	Long node-edge value array; default: `NA`

Details

The expected value at each node is calculate by

$$\hat{c}_i = c_i + \sum p_{ij} \hat{c}_j$$

The default calculation assumes that the costs are associated with the nodes. An alternative would be to associate them with the edges. For total expected cost this doesn't matter but for the other nodes this is different to assuming the costs are assigned to the nodes. The expected value would then be

$$\hat{c}_i = \sum p_{ij}(c_{ij} + \hat{c}_j)$$

Value

Expected value at each node (vector)

`define_model` *Define model*

Description

Basic constructor for decision tree classes for different data formats.

Usage

```
define_model(transmat, tree_dat, dat_long)
```

Arguments

`transmat`	Transition probability matrix (from-to node)
`tree_dat`	Hierarchical tree structure of parents and children
`dat_long`	Long dataframe with from, to, prob, vals columns

Value

transmat, tree_dat or dat_long class object

Examples

```
define_model(transmat =
              list(prob = matrix(data=c(NA, 0.5, 0.5), nrow = 1),
                   vals = matrix(data=c(NA, 1, 2), nrow = 1)
              ))

define_model(tree_dat =
              list(child = list("1" = c(2, 3),
                                "2" = NULL,
                                "3" = NULL),
                   dat = data.frame(node = 1:3,
                                    prob = c(NA, 0.5, 0.5),
                                    vals = c(0, 1, 2))
              ))

define_model(dat_long = data.frame(from = c(NA, 1, 1),
                                   to = 1:3,
                                   prob = c(NA, 0.5, 0.5),
                                   vals = c(0, 1, 2)))
```

get_children_list *Get tree children list*

Description

Get tree children list by parents from a transition matrix.

Usage

```
get_children_list(transmat)
```

Arguments

transmat	from-to matrix with NA for missing values.

Value

list

is_prob_matrix *Is object a transition probability matrix?*

Description

Is object a transition probability matrix?

Usage

```
is_prob_matrix(probs)
```

Arguments

probs	matrix

Value

logical

Examples

```
## Not run:
probs <- matrix(c(1,0,0,1), nrow = 2)
is_prob_matrix(probs)

probs <- matrix(c(2,0,-1,1), nrow = 2)
assert_that(is_prob_matrix(probs))

## End(Not run)
```

`long_to_transmat` *Long format to transition matrix*

Description

Long format to transition matrix

Usage

```
long_to_transmat(dat)
```

Arguments

`dat`	array of from, to, prob, vals

Value

transition matrix

`trans_binarytree` *Transition matrix to binary tree*

Description

This is adapted from `mstate::trans.illness`. Create a complete binary tree transition matrix.

Usage

```
trans_binarytree(names, depth = 2)
```

Arguments

`names`	Node names
`depth`	Depth of tree

Value

Matrix of TRUE and FALSE

Index