Package ‘TreeMineR’

April 5, 2025

Type Package

Title Tree-Based Scan Statistics

Version 1.0.3

Description Implementation of unconditional Bernoulli Scan Statistic developed by Kulldorff et al. (2003) <doi:10.1111/1541-0420.00039> for hierarchical tree structures. Tree-based Scan Statistics are an exploratory method to identify event clusters across the space of a hierarchical tree.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), tidyr (>= 1.3.0), comorbidity (>= 1.0.7), knitr, rmarkdown

Config/testthat/edition 3

Imports data.table, future, future.apply, cli (>= 3.6.1)

URL `https://entjos.github.io/TreeMineR/`, `https://github.com/entjos/TreeMineR`

BugReports `https://github.com/entjos/TreeMineR/issues`

VignetteBuilder knitr

NeedsCompilation no

Author Joshua P. Entrop [aut, cre, cph] (<`https://orcid.org/0000-0003-1614-8096`>), Viktor Wintzell [aut]

Maintainer Joshua P. Entrop <`joshuaentrop@posteo.de`>

Repository CRAN

Date/Publication 2025-04-05 13:10:02 UTC

Contents

`atc_codes` *Hierarchical tree of the ATC system for classifying drugs*

Description

A dataset including the following column:

`pathString` A string identifying all the parents of a node. Each parent is separated by a `/`.

Usage

```
data(atc_codes)
```

`create_tree` *Creating a tree file for further use in `TreeMineR()`.*

Description

Creating a tree file for further use in `TreeMineR()`.

Usage

```
create_tree(x)
```

Arguments

`x`	A data frame that includes two or three columns: `node` A string defining a node `parent` A string defining the partent of the node

Value

A data.frame with one variable `pathString` that describes the full path for each leaf included in the hierarchical tree.

`diagnoses` *Test dataset of ICD diagnoses*

Description

A simulated dataset of hospital diagnoses created with the help of the `comorbidity` package including the following columns:

id Individual identifier,

case Indicator for case status,

diag An ICD-10 diagnosis code.

Usage

```
data(diagnoses)
```

Format

A data frame with 23,144 rows and 3 columns

`drop_cuts` *Remove cuts from your tree. This is, e.g., useful if you would like to remove certain chapters from the ICD-10 tree used for the analysis as some chapters might be a prior deemed irrelevant for the exposure of interest, e.g., chapter 20 (external causes of death) might not be of interest when comparing two drug exposures.*

Description

Remove cuts from your tree. This is, e.g., useful if you would like to remove certain chapters from the ICD-10 tree used for the analysis as some chapters might be a prior deemed irrelevant for the exposure of interest, e.g., chapter 20 (external causes of death) might not be of interest when comparing two drug exposures.

Usage

```
drop_cuts(tree, cuts, delimiter = "/", return_removed = FALSE)
```

Arguments

`tree`	A dataset with one variable `pathString` defining the tree structure that you would like to use. This dataset can, e.g., be created using `create_tree`.
`cuts`	A character vector of cuts to remove. Please make sure that your string uniquely identifies the cut that should be removed. Each string is passed to `base::gsub()` to identify the cuts that should be removed. Hence, strings can include regular expressions for identifying cuts. If you would like to remove a cut on the top level of the hierarchy, it might be helpful to use the regular expression operator `^`. Regular expression are composed as follows: `paste0(cuts, delimiter, "?(.*)")`
`delimiter`	A character defining the delimiter of different tree levels within your `pathString`. The default is `/`.
`return_removed`	A logical value for indicating whether you would like to get a list of removed cuts returned by the function.

Value

If `return_removed = FALSE` a data.frame with a single variable named `pathString` is returned, which includes the updated tree. If `return_removed = TRUE` a list with two elements is return:

tree The updated tree file

removed A list of character vectors including the paths that have been removed from the supplied tree. The list is named using the cuts supplied to `cut`.

Examples

```
drop_cuts(icd_10_se, c("B35-B49", "F41")) |>
  head()
```

`icd_10_se` *Swedish version of the ICD-10 diagnoses code tree*

Description

A dataset including the following column:

`pathString` A string identifying all the parents of a node. Each parent is separated by a `/`.

Usage

```
data(icd_10_se)
```

`icd_10_se_dict` *Dictionary for the Swedish version of the ICD-10 diagnoses code tree*

Description

A dataset including the following column:

`node` A string identidying a node

`title` A label for the node

Usage

```
data(icd_10_se_dict)
```

`TreeMineR` *Unconditional Bernoulli Tree-Based Scan Statistics for R*

Description

Unconditional Bernoulli Tree-Based Scan Statistics for R

Usage

```
TreeMineR(
  data,
  tree,
  p = NULL,
  n_exposed = NULL,
  n_unexposed = NULL,
  dictionary = NULL,
  delimiter = "/",
  n_monte_carlo_sim = 9999,
  random_seed = FALSE,
  return_test_dist = FALSE,
  future_control = list(strategy = "sequential")
)
```

Arguments

`data`	The dataset used for the computation. The dataset needs to include the following columns: `id` An integer that is unique to every individual. `leaf` A string identifying the unique diagnoses or leafs for each individual. `exposed` A 0/1 indicator of the individual's exposure status.

See below for the first and last rows included in the example dataset.

```
   id leaf exposed
    1 K251       0
    2 Q702       0
    3  G96       0
    3 S949       0
    4 S951       0
 ---
  999 V539       1
  999 V625       1
  999 G823       1
 1000  L42       1
 1000 T524       1
```

`tree`	A dataset with one variable `pathString` defining the tree structure that you would like to use. This dataset can, e.g., be created using `create_tree`.
`p`	The proportion of exposed individuals in the dataset. Will be calculated based on `n_exposed`, and `n_unexposed` if both are supplied.
`n_exposed`	Number of exposed individuals (Optional).
`n_unexposed`	Number of unexposed individuals (Optional).
`dictionary`	A `data.frame` that includes one `node` column and a `title` column, which are used for labeling the cuts in the output of `TreeMineR`.
`delimiter`	A character defining the delimiter of different tree levels within your `pathString`. The default is `/`.
`n_monte_carlo_sim`	The number of Monte-Carlo simulations to be used for calculating P-values.
`random_seed`	Random seed used for the Monte-Carlo simulations.
`return_test_dist`	If `true`, a data.frame of the maximum log-likelihood ratios in each Monte Carlo simulation will be returned. This distribution of the maximum log-likelihood ratios is used for estimating the P-value reported in the result table.
`future_control`	A list of arguments passed `future::plan`. This is useful if one would like to parallelise the Monte-Carlo simulations to decrease the computation time. The default is a sequential run of the Monte-Carlo simulations.

Value

A `data.frame` with the following columns:

`cut` The name of the cut G.

`n1` The number of exposed events belonging to cut G.

`n1` The number of inexposed events belonging to cut G.

`risk1` The absolute risk of getting an event belonging to cut G among the exposed.

`risk0` The absolute risk of getting an event belonging to cut G among the unexposed.

`RR` The risk ratio of the absolute risk among the exposed over the absolute risk among the unexposed

`llr` The log-likelihood ratio comparing the observed and expected number of exposed events belonging to cut G.

`p` The P-value that cut G is a cluster of events.

If `return_test_dist` is `true` the function returns a list of two data.frame.

`result_table` A data.frame including the results as described above.

`test_dist` A data.frame with two columns: `iteration` the number of the Monte Carlo iteration. Note that iteration is the calculation based on the original data and is, hence, not included in this data.fame. `max_llr`: the highest observed log-likelihood ratio for each Monte Carlo simulation

References

Kulldorff et al. (2003) A tree-based scan statistic for database disease surveillance. Biometrics 56(2): 323-331. DOI: 10.1111/1541-0420.00039.

Examples

```
TreeMineR(data = diagnoses,
          tree  = icd_10_se,
          p = 1/11,
          n_monte_carlo_sim = 99,
          random_seed = 1234) |>
  head()
```

Index