Package: nuggets (via r-universe)

September 11, 2024

Title Extensible Data Pattern Searching Framework

Version 1.0.2

Date 2024-01-08

Maintainer Michal Burda <michal.burda@osu.cz>

Description Extensible framework for subgroup discovery (Atzmueller (2015) <doi:10.1002/widm.1144>), contrast patterns (Chen (2022) <doi:10.48550/arXiv.2209.13556>), emerging patterns (Dong (1999) <doi:10.1145/312129.312191>) and association rules (Agrawal (1994) <https://www.vldb.org/conf/1994/P487.PDF>). Both crisp (binary) and fuzzy data are supported. It generates conditions in the form of elementary conjunctions, evaluates them on a dataset and checks the induced sub-data for interesting statistical properties. Currently, the package searches for implicative association rules and conditional correlations (Hájek (1978) <doi:10.1007/978-3-642-66943-9>). A user-defined function may be defined to evaluate on each generated condition to search for custom patterns.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Language en-US

Imports cli, methods, Rcpp, rlang, stats, tibble, tidyr, tidyselect

LinkingTo Rcpp, testthat

SystemRequirements C++17

Suggests testthat (>= 3.0.0), xml2

Config/testthat/edition 3

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

RemoteUrl https://github.com/beerda/nuggets

RemoteRef HEAD

RemoteSha 6bee040b605f3a9663f9b5474c1b8af1a5526c4b

Contents

`dichotomize` *Create dummy columns from logicals or factors in a data frame*

Description

Create dummy logical columns from selected columns of the data frame. Dummy columns may be created for logical or factor columns as follows:

Usage

```
dichotomize(.data, what = everything(), ..., .keep = FALSE, .other = FALSE)
```

Arguments

`.data`	a data frame to be processed
`what`	a tidyselect expression (see tidyselect syntax) selecting the columns to be processed
`...`	further tidyselect expressions for selecting the columns to be processed
`.keep`	whether to keep the original columns. If FALSE, the original columns are removed from the result.
`.other`	whether to put into result the rest of columns that were not specified for dichotomization in `what` argument.

Details

- for logical column `col`, a pair of columns is created named `col=T` and `col=F` where the former (resp. latter) is equal to the original (resp. negation of the original);
- for factor column `col`, a new logical column is created for each level `l` of the factor `col` and named as `col=l` with a value set to TRUE wherever the original column is equal to `l`.

Value

A tibble with selected columns replaced with dummy columns.

Author(s)

Michal Burda

dig *Search for rules*

Description

This is a general function that enumerates all conditions created from data in `x` and calls the callback function `f` on each.

Usage

```
dig(x, f, ...)

## Default S3 method:
dig(x, f, ...)

## S3 method for class 'matrix'
dig(
  x,
  f,
  condition = everything(),
  focus = NULL,
  disjoint = NULL,
  min_length = 0,
  max_length = Inf,
  min_support = 0,
  min_focus_support = min_support,
  filter_empty_foci = FALSE,
  t_norm = "goguen",
  threads = 1,
  ...
)

## S3 method for class 'data.frame'
dig(
  x,
  f,
  condition = everything(),
  focus = NULL,
  disjoint = NULL,
  min_length = 0,
  max_length = Inf,
  min_support = 0,
  min_focus_support = min_support,
  filter_empty_foci = FALSE,
  t_norm = "goguen",
  threads = 1,
  ...
```

)

Arguments

x	a matrix or data frame. The matrix must be numeric (double) or logical. If x is a data frame then each column must be either numeric (double) or logical.
f	the callback function executed for each generated condition. This function may have some of the following arguments. Based on the present arguments, the algorithm would provide information about the generated condition: - condition - a named integer vector of column indices that represent the predicates of the condition. Names of the vector correspond to column names; - foci_supports - a named numeric vector of supports of foci columns (see focus argument to specify, which columns are foci) - names of the vector are foci column names; - support - a numeric scalar value of the current condition's support; - indices - a logical vector indicating the rows satisfying the condition; - weights - (similar to indices) weights of rows to which they satisfy the current condition.
...	Further arguments, currently unused.
condition	a tidyselect expression (see tidyselect syntax) specifying the columns to use as condition predicates
focus	a tidyselect expression (see tidyselect syntax) specifying the columns to use as focus predicates
disjoint	an atomic vector of size equal to the number of columns of x that specifies the groups of predicates: if some elements of the disjoint vector are equal, then the corresponding columns of x will NOT be present together in a single condition.
min_length	the minimum size (the minimum number of predicates) of the condition to be generated (must be greater or equal to 0). If 0, the empty condition is generated in the first place.
max_length	The maximum size (the maximum number of predicates) of the condition to be generated. If equal to Inf, the maximum length of conditions is limited only by the number of available predicates.
min_support	the minimum support of a condition to trigger the callback function for it. The support of the condition is the relative frequency of the condition in the dataset x. For logical data, it equals to the relative frequency of rows such that all condition predicates are TRUE on it. For numerical (double) input, the support is computed as the mean (over all rows) of multiplications of predicate values.
min_focus_support	the minimum support of a focus, for the focus to be passed to the callback function. The support of the focus is the relative frequency of rows such that all condition predicates AND the focus are TRUE on it. For numerical (double) input, the support is computed as the mean (over all rows) of multiplications of predicate values.
filter_empty_foci	a logical scalar indicating whether to skip conditions, for which no focus remains available after filtering by min_focus_support. If TRUE, the condition

is passed to the callback function only if at least one focus remains after filtering. If FALSE, the condition is passed to the callback function regardless of the number of remaining foci.

t_norm — a t-norm used to compute conjunction of weights. It must be one of "goedel" (minimum t-norm), "goguen" (product t-norm), or "lukas" (Lukasiewicz t-norm).

threads — the number of threads to use for parallel computation.

Value

A list of results provided by the callback function f.

Author(s)

Michal Burda

---

dig_correlations *Search for conditional correlations*

Description

Compute correlation between all combinations of xvars and yvars columns of x in subdata corresponding to conditions generated from condition columns.

Usage

```
dig_correlations(
  x,
  condition = where(is.logical),
  xvars = where(is.numeric),
  yvars = where(is.numeric),
  method = "pearson",
  alternative = "two.sided",
  exact = NULL,
  min_length = 0L,
  max_length = Inf,
  min_support = 0,
  threads = 1,
  ...
)
```

Arguments

x — a matrix or data frame with data to search in. The matrix must be numeric (double) or logical. If x is a data frame then each column must be either numeric (double) or logical.

condition	a tidyselect expression (see tidyselect syntax) specifying the columns to use as condition predicates
xvars	a tidyselect expression (see tidyselect syntax) specifying the columns to use for computation of correlations
yvars	a tidyselect expression (see tidyselect syntax) specifying the columns to use for computation of correlations
method	a character string indicating which correlation coefficient is to be used for the test. One of "pearson", "kendall", or "spearman"
alternative	indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less". "greater" corresponds to positive association, "less" to negative association.
exact	a logical indicating whether an exact p-value should be computed. Used for Kendall's *tau* and Spearman's *rho*. See stats::cor.test() for more information.
min_length	the minimum size (the minimum number of predicates) of the condition to be generated (must be greater or equal to 0). If 0, the empty condition is generated in the first place.
max_length	The maximum size (the maximum number of predicates) of the condition to be generated. If equal to Inf, the maximum length of conditions is limited only by the number of available predicates.
min_support	the minimum support of a condition to trigger the callback function for it. The support of the condition is the relative frequency of the condition in the dataset x. For logical data, it equals to the relative frequency of rows such that all condition predicates are TRUE on it. For numerical (double) input, the support is computed as the mean (over all rows) of multiplications of predicate values.
threads	the number of threads to use for parallel computation.
...	Further arguments, currently unused.

Value

A tibble with found rules.

Author(s)

Michal Burda

See Also

dig(), stats::cor.test()

`dig_implications` *Search for implicative rules*

Description

Implicative rule is a rule of the form $A \Rightarrow c$, where A (*antecedent*) is a set of predicates and c (*consequent*) is a predicate.

Usage

```
dig_implications(
  x,
  antecedent = everything(),
  consequent = everything(),
  disjoint = NULL,
  min_length = 0L,
  max_length = Inf,
  min_coverage = 0,
  min_support = 0,
  min_confidence = 0,
  t_norm = "goguen",
  threads = 1,
  ...
)
```

Arguments

`x`	a matrix or data frame with data to search in. The matrix must be numeric (double) or logical. If `x` is a data frame then each column must be either numeric (double) or logical.
`antecedent`	a tidyselect expression (see tidyselect syntax) specifying the columns to use in the antecedent (left) part of the rules
`consequent`	a tidyselect expression (see tidyselect syntax) specifying the columns to use in the consequent (right) part of the rules
`disjoint`	an atomic vector of size equal to the number of columns of `x` that specifies the groups of predicates: if some elements of the `disjoint` vector are equal, then the corresponding columns of `x` will NOT be present together in a single condition.
`min_length`	the minimum length, i.e., the minimum number of predicates in the antecedent, of a rule to be generated. Value must be greater or equal to 0. If 0, rules with empty antecedent are generated in the first place.
`max_length`	The maximum length, i.e., the maximum number of predicates in the antecedent, of a rule to be generated. If equal to Inf, the maximum length is limited only by the number of available predicates.
`min_coverage`	the minimum coverage of a rule in the dataset `x`. (See Description for the definition of *coverage*.)

`min_support`	the minimum support of a rule in the dataset `x`. (See Description for the definition of *support*.)
`min_confidence`	the minimum confidence of a rule in the dataset `x`. (See Description for the definition of *confidence*.)
`t_norm`	a t-norm used to compute conjunction of weights. It must be one of `"goedel"` (minimum t-norm), `"goguen"` (product t-norm), or `"lukas"` (Lukasiewicz t-norm).
`threads`	the number of threads to use for parallel computation.
`...`	Further arguments, currently unused.

Details

For the following explanations we need a mathematical function $supp(I)$, which is defined for a set I of predicates as a relative frequency of rows satisfying all predicates from I. For logical data, $supp(I)$ equals to the relative frequency of rows, for which all predicates $i_1, i_2, \ldots, i_n$ from I are TRUE. For numerical (double) input, $supp(I)$ is computed as the mean (over all rows) of truth degrees of the formula `i_1 AND i_2 AND ... AND i_n`, where `AND` is a triangular norm selected by the `t_norm` argument.

Implicative rules are characterized with the following quality measures.

Length of a rule is the number of elements in the antecedent.

Coverage of a rule is equal to $supp(A)$.

Support of a rule is equal to $supp(A \cup \{c\})$.

Confidence of a rule is the fraction $supp(A)/supp(A \cup \{c\})$.

Value

A tibble with found rules and computed quality measures.

Author(s)

Michal Burda

See Also

`dig()`

`format_condition` *Format condition - convert a character vector to character scalar*

Description

Function takes a character vector of predicates and returns a formatted condition.

Usage

```
format_condition(condition)
```

Arguments

condition	a character vector

Value

a character scalar

Author(s)

Michal Burda

Examples

```
format_condition(NULL)              # returns {}
format_condition(c("a", "b", "c"))  # returns {a,b,c}
```

---

is_subset *Determine whether the first vector is a subset of the second vector*

Description

Determine whether the first vector is a subset of the second vector

Usage

```
is_subset(x, y)
```

Arguments

x	the first vector
y	the second vector

Value

TRUE if x is a subset of y or FALSE otherwise.

Author(s)

Michal Burda

`which_antichain` *Return indices of first elements of the list, which are incomparable with preceding elements.*

Description

The function returns indices of elements from the given list `x`, which are incomparable (i.e., it is neither subset nor superset) with any preceding element. The first element is always selected. The next element is selected only if it is incomparable with all previously selected elements.

Usage

```
which_antichain(x, distance = 0)
```

Arguments

`x`	a list of integerish vectors
`distance`	a non-negative integer, which specifies the allowed discrepancy between compared sets

Value

an integer vector of indices of selected (incomparable) elements.

Author(s)

Michal Burda

Index