Package 'rPref'

August 4, 2014

Version 0.1

Date 2014-08-01

Title Preferences and Skyline Computation in R

Author Patrick Roocks <mail@p-roocks.de>

Maintainer Patrick Roocks <mail@p-roocks.de>

Description Implementing Preferences in R to retrieve Maxima for a given strict partial order. This also includes the Skyline Operator, which allows to retrieve the pareto-optimal values.

URL http://www.p-roocks.de/rpref

Depends R (>= 3.0.0), methods

Imports Rcpp, dplyr, igraph

License GPL (>= 2)

LazyData true

LinkingTo Rcpp

Suggests testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-04 16:48:39

R topics documented:

`base_pref` *Base preferences*

Description

Base preferences are used to describe the different goals of a preference query.

Usage

```
low(expr)

high(expr)

true(expr)
```

Arguments

`expr`	Value which is the term for the current preference and should be minimized/maximized or, for boolean preferences, `TRUE`. The term `expr` can be just a single attribute an arbitrary expression, e.g., `low(a+2*b+f(c))`, where `a`, `b` and `c` are columns of the addressed dataset and `f` is a previously defined function.

Details

All base preferences are mathematically strict weak orders (irreflexive, transitive and negative transitive).

The three fundamental base preferences are:

`low(a), high(a)` Search for minimal/maximal values of `a`, i.e., the induced order is the "smaller than" or "greater than" order on the values of `a`. The values of `a` must be numeric values.

`true(a)` Search for true values in logical expressions, i.e. `TRUE` is better than `FALSE`. The values of `a` must be logical values.

Functions contained in `expr` are evaluated over the entire dataset, i.e., it is possible to use aggregate functions (`min`, `mean`, etc.). Note that all functions (and also variables not occuring in the dataset, where `pref` will be evaluated on) must be defined in the same environment (e.g. function scope) as the base preference.

See Also

See `complex_pref` how to compose complex preferences to retrieve e.g. the Skyline. See `base_pref_macros` for more base preferences.

Examples

```
# Define a preference with a score value combining mpg and hp
p1 <- high(4*mpg + hp)
# Perform the preference selection
psel(mtcars, p1)

# Define a preference with a given function
f <- function(x, y) (abs(x-mean(x))/max(x) + abs(y-mean(y))/max(y))
p2 <- low(f(mpg, hp))
psel(mtcars, p2)
```

---

base_pref_macros *Useful base preference macros*

Description

In addition to the fundamental base preferences, rPref offers some macros to define preferences where a given interval or point is preferred.

Usage

```
around(expr, center)

between(expr, left, right)

pos(expr, pos_value)

layered(expr, ...)
```

Arguments

`expr`	The value which should be in the preferred interval, layer, etc. The same requirements as for `base_pref` apply.
`center`	Preferred value, where a values from the given `expr` should be near by.
`left`	Lower limit for a preferred interval.
`right`	Upper limit for a preferred interval.
`pos_value`	The preferred value or set for a pos-preference.
`...`	Layers (sets) for a layered-Preference, where the first set are the most preferred values.

Definition of the preference macros

`between(expr, left, right)` Those tuples are preferred, where `expr` evaluates to a value between l and r. For values not in this interval, the values nearest to [l, r] are preferred.

`around(expr, center)` Same as `between(expr, center, center)`.

`pos(expr, pos_value)` If `expr` evaluates to a value which is contained in `pos_value`, these tuples are preferred.

`layered(expr, layer1, layer2, ..., layerN)` For the most preferred tuples, `expr` must evaluate to a value in `layer1`. The second-best tuples are those, where `expr` evaluates to a value in `layer2`, and so forth. Values occuring in non of the layers are considered worse than those in `layerN`. Technically, this is realized by a Prioritization (lexicographical order) chain of boolean preferences.

Examples

```
# Search for cars where mpg is near to 25
psel(mtcars, around(mpg, 25))

# cyl = 2 and cyl = 4 are equally good, cyl = 6 is worse
psel(mtcars, layered(cyl, c(2, 4), 6))
```

`complex_pref` *Complex preferences*

Description

Complex preferences are used to compose different preferences orders. For example the Pareto composition (via operator `*`) is the usual operator to compose the preference for a Skyline query. All complex preferences are mathematically strict partial orders (irreflexive and transitive).

Usage

```
## S3 method for class 'preference'
p1 * p2

## S3 method for class 'preference'
p1 & p2

## S3 method for class 'preference'
p1 | p2

## S3 method for class 'preference'
p1 + p2

reverse(p1)
```

Arguments

`p1,p2`	Preferences to be composed (either base preferences via `base_pref` or also complex preferences)

Skylines

The most important preference composition operator is the Pareto-Operator (`p1 * p2`) to formulate a Skyline query. A tuple t1 is better than t2 w.r.t. `p1 * p2` if it is strictly better w.r.t. one of the preferences p1, p2 and is better or equal w.r.t. the other preference.

The syntactical translation from other query languages to rPref is as follows:

- A query in the syntax from Borzsonyi et. al (2001) like
  `"... SKYLINE OF a MAX, b MIN, c MAX"`
  corresponds in rPref to the preference
  `low(a) * high(b) * low(c)`.
- A query in the syntax from Kiessling (2002) like
  `"... PREFERRING a LOWEST and (b HIGHEST PRIOR TO c LOWEST)"`
  corresponds in rPref to
  `low(a) * (high(b) & low(c))`.

Definition of additional preference operators

Additionally, rPref supports the following preference composition operators:

`p1 & p2` Prioritization (Lexicographical order): A tuple t1 is better than t2 w.r.t. `p1 & p2` if it is strictly better w.r.t. `p1` or is euqal w.r.t. `p1` and is better w.r.t. `p2`.

`p1 | p2` Intersection preference: A tuple t1 is better than t2 w.r.t. `p1 | p2` if it is strictly better w.r.t. both preferences.

`p1 + p2` Union preference: A tuple t1 is better than t2 w.r.t. `p1 + p2` if it is strictly better w.r.t. to one of the preferences. Note that this can violate the strict partial order property, if the domains (the tuples on which `p1` and `p2` define better-than-relationships) of the preferences are not disjoint.

`reverse(p1)` **or** `-p1` Reverse preference (converse relation): A tuple t1 is better than t2 w.r.t. `-p1` if t2 is better than t1 w.r.t. `p1`. The unary minus operator, i.e. `-p1`, is a short for `reverse(p1)`.

References

W. Kiessling (2002): Foundations of Preferences in Database Systems. In Very Large Data Bases (VLDB '02), pages 311-322.

S. Borzsonyi, D. Kossmann, K. Stocker (2001): The Skyline Operator. In Data Engineering (ICDE '01), pages 421-430.

See Also

See `base_pref` for the construction of base preferences. See `psel` for the evaluation of preferences.

Examples

```
# Define preference for cars with low consumption (high mpg-value)
# and simultanously high horsepower
p1 <- high(mpg) * high(hp)

# Perform the preference search
psel(mtcars, p1)
```

---

get_btg *Better-Than-Graph*

Description

Returns a Hasse-Diagramm of a preference order (also called the Better-Than-Graph) on a given dataset to be plotted with the igraph package.

Usage

```
get_btg(df, pref)
```

Arguments

df	A dataframe.
pref	A preference on the columns of df, see psel for details.

Details

This function returns a list l with the following list entries:

l$graph An igraph object, created with the igraph package.

l$layout A typical Hasse-diagram layout for plotting the graph, also created with igraph.

To plot the resulting graph, use the plot function as follows: plot(l$graph, layout = l$layout). For more details, see igraph.plotting and the examples below.

The Hasse diagram of a preference visualizes all the better-than-relationsships on a given dataset. All the edges which can be retrieved by transitivity of the the order are omitted.

The names of the vertices are characters ranging from "1" to as.character(nrow(df)) and they correspond to the row numbers of df.

See Also

igraph.plotting

Examples

```
# Pick a small data set and create preference / BTG
df <- mtcars[1:10,]
pref <- high(mpg) * high(hp)
btg <- get_btg(df, pref)

# Create labels for the nodes with relevant values
labels <- paste0(df$mpg, "\n", df$hp)

# Plot the graph using igraph
library(igraph)
plot(btg$graph, layout = btg$layout, vertex.label = labels,
     vertex.size = 25)

# Add colors for the maxima nodes and plot again
colors <- rep(rgb(1,1,1), nrow(df))
colors[psel.indices(df, pref)] <- rgb(0,1,0)
plot(btg$graph, layout = btg$layout, vertex.label = labels,
     vertex.size = 25, vertex.color = colors)

# Show lattice structure of 3-dimensional Pareto preference
df <- merge(merge(list(x = 1:3), list(y = 1:3)), list(z = 1:2))
labels <- paste0(df$x, ",", df$y, ",", df$z)
btg <- get_btg(df, low(x) * low(y) * low(z))
plot(btg$graph, layout = btg$layout, vertex.label = labels,
     vertex.size = 20)
```

---

psel *Preference selection*

Description

Evaluate a preference on a given dataset, i.e. return the maximal elements of a dataset for a given preference order.

Usage

```
psel(df, pref, top = NULL)

psel.indices(df, pref, top = NULL)
```

Arguments

df	A dataframe or, for grouped preference selection, a grouped dataframe. See below for details.
pref	The preference order, constructed via complex_pref and base_pref. All variables occuring in pref must be columns of the dataframe df (or can be variables of the environment, where pref was defined).

top — By default NULL, which means that the maxima are returned. For top = k the k-best elements according to the preference are returned.

Details

The difference between the two variants of the preference selection is:

- The psel function returns a subset of the dataset which are the maxima according to the given preference.
- The function psel.indices returns just the row indices of the maxima. Hence psel(df,pref,top) is equivalent to df[psel(df,pref,top),] for non-grouped dataframes. For grouped dataframes, the groups are restored after the preference selection.

For a given top value "k", the k best elements are returned. By definition, this is non-deterministic. A top-1 query of two equivalent tuples (according to pref) can return on both of these tuples. In the rPref implementation, in this case the first occuring tuple in the dataset is picked.

If the top value is greater than the number of elements in df, i.e., top > nrow(df) then all elements of df will be returned without further warning.

Grouped preference selection

With psel it is also possible to perform a preference selection, where the maxima are calculated for every group seperatly. The groups have to be created with group_by from the dplyr package. The preference selection preserves the grouping, i.e., the summarize function from dplyr refers to the set of maxima of each group. This can be used to e.g. calculate the number of maxima in each group, see examples below.

A given top value k in connection with a grouped preference selection returns the k best values for each group. Hence if there are three groups in df, each containing at least 2 elements, and we have top = 2 then 6 tuples will be returned.

See Also

See complex_pref how to construct a Skyline preference.

Examples

```
# Skyline and Top-K skyline
psel(mtcars, low(mpg) * low(hp))
psel(mtcars, low(mpg) * low(hp), top = 5)

# Visualize the skyline in a plot
sky1 <- psel(mtcars, high(mpg) * high(hp))
plot(mtcars$mpg, mtcars$hp)
points(sky1$mpg, sky1$hp, lwd=3)

# Grouped preference with dplyr
library(dplyr)
psel(group_by(mtcars, cyl), low(mpg))

# Return size of each maxima group
summarise(psel(group_by(mtcars, cyl), low(mpg)), n())
```

rPref *The rPref package.*

Description

The rPref package.

Index