Package 'arulesViz'

October 3, 2016

Version 1.2-0

Date 2016-10-02

Title Visualizing Association Rules and Frequent Itemsets

Author Michael Hahsler and Sudheer Chelluboina

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Depends arules (>= 1.4.1), grid

Imports scatterplot3d, vcd, seriation, igraph (>= 1.0.0), graphics, methods, utils, grDevices, stats, colorspace, DT, plotly

Suggests graph, Rgraphviz, iplots

Description
Extends package arules with various visualization techniques for association rules and itemsets. The package also includes several interactive visualizations for rule exploration.

License GPL-3

URL http://lyle.smu.edu/IDA/arules/

BugReports https://github.com/mhahsler/arulesViz/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-03 08:17:40

R topics documented:

inspectDT *Inspect Associations Interactively using datatable*

Description

Uses **datatable** to create a HTML table widget using the DataTables library. Rules can be interactively filtered and sorted.

Usage

```
inspectDT(x, ...)
```

Arguments

x	an object of class "rules" or "itemsets".
...	additional arguments. `precision` controls the precision used to print the quality measures (defaults to 2). All other arguments are passed on to `datatable`.

Value

A datatable htmlwidget.

Author(s)

Michael Hahsler

See Also

`datatable` in **DT**.

Examples

```
## Not run:
data(Groceries)
rules <- apriori(Groceries, parameter=list(support=0.005, confidence=0.5))
rules

inspectDT(rules)

### save table as a html page.
p <- inspectDT(rules)
htmlwidgets::saveWidget(as.widget(p), "arules.html")
## End(Not run)
```

plot *Plot method to visualize association rules and itemsets*

Description

This is the S3 method to visualize association rules and itemsets. Implemented are several popular visualization methods including scatter plots with shading (two-key plots), graph based visualizations, doubledecker plots, etc.

Interactive plotting with plotly is available with `plotly_arules`.

Usage

```
## S3 method for class 'rules'
plot(x, method = NULL, measure = "support", shading = "lift",
    interactive = FALSE, data = NULL, control = NULL, ...)
## S3 method for class 'itemsets'
plot(x, method = NULL, measure = "support", shading = NA,
    interactive=FALSE, data = NULL, control = NULL, ...)
```

Arguments

`x`	an object of class "rules" or "itemsets".
`method`	a string with value "scatterplot", "two-key plot", "matrix", "matrix3D", "mosaic", "doubledecker", "graph", "paracoord" or "grouped", "iplots" selecting the visualization method (see Details).
`measure`	measure(s) of interestingness (e.g., "support", "confidence", "lift", "order") used in the visualization. Some visualization methods need one measure, others take a vector with two measures (e.g., scatterplot). In some plots (e.g., graphs) `NA` can be used to suppress using a measure.
`shading`	measure of interestingness used for the color of the points/arrows/nodes (e.g., "support", "confidence", "lift"). The default is "lift". `NA` can be often used to suppress shading.
`interactive`	enable interactive exploration (not implemented by all methods).
`control`	a list of control parameters for the plot. The available control parameters depends on the visualization technique (see Details).
`data`	the dataset (class "transactions") used to generate the rules/itemsets. Only "mosaic" and "doubledecker" require the original data.
`...`	further arguments are usually passed on to the low level plotting function. For scatterplot it is added for convenience to the control list (see above).

Details

Most visualization techniques are described by Bruzzese and Davino (2008), however, we added more color shading, reordering and interactive features. Many visualization methods take extra parameters as a list in the control parameter. Some of the parameters are described below. Use verbose mode with, e.g., `plot(rules, method = "graph", control = list(verbose = TRUE))` to get a complete list of parameters and theit default values.

The following visualization method are available:

"scatterplot", "two-key plot" This visualization method draws a two dimensional scatterplot with different measures of interestingness (parameter "measure") on the axes and a third measure (parameter "shading") is represented by the color of the points. There is a special value for shading called "order" which produces a two-key plot where the color of the points represents the length (order) of the rule.

Interactive manipulations are available.

The list of control parameters for this method is:

"main" plot title

"pch" use filled symbols: 20–25

"cex" symbol size

"xlim","ylim" limits

"jitter" a number greater than 0 adds jitter to the points

"col" color palette

"matrix", "matrix3D" Arranges the association rules as a matrix with the itemsets in the antecedents on one axis and the itemsets in the consequent on the other. The interest measure is either visualized by a color (darker means a higher value for the measure) or as the height of a bar (method "matrix3D").

Currently there is no interactive version available.

The list of control parameters for this method is:

"main" plot title

"type" defines the way the data is rendered: "grid", "image" or "3D" (scatterplot3d)

"reorder" if `TRUE` then the itemsets on the x and y-axes are reordered to bring rules with similar values for the interest measure closer together and make the plot clearer.

"orderBy" specifies the measure of interest for reordering (default is the visualized measure)

"reorderMethod","reorderControl","reorderDist" seriation method, control arguments and distance method (default "euclidean") used for reordering (see `seriate()` method in **seriation**)

"col" a vector of n colors used for the plot (default: 100 heat colors)

"xlim","ylim" limits

"grouped" Grouped matrix-based visualization (Hahsler and Chelloboina, 2011; Hahsler and Karpienko, 2016). Antecedents (columns) in the matrix are grouped using clustering. Groups are represented as balloons in the matrix.

Interactive manipulations are available.

The list of control parameters for this method is:

"main" plot title

"k" number of antecedent groups (default: 20)

"aggr.fun" aggregation function can be any function computing a scalar from a vector (e.g., min, mean, median (default), sum, max). It is also used to reorder the balloons in the plot.

"col" color palette (default is 100 heat colors.)

"gp_labels", "gp_main", "gp_labs", "gp_lines" `gpar()` objects used to specify color, font and font size for different elements.

"graph" Represents the rules (or itemsets) as a graph.

Control arguments are:

"main" plot title

"cex" cex for labels

"itemLabels" display item/itemset names instead of ids (`TRUE`)

"edgeCol", "nodeCol" color palettes for edges and nodes.

"measureLabels" display values of interest measures (`FALSE`)

"precision" number of digits for numbers in plot.

"type" vertices represent: "items" (default) or "itemsets"

"engine" graph layout engine: "igraph" (default) or "graphviz"

"layout" layout algorithm defined in **igraph** or **Rgraphviz** (default: igraph::nicely which usually does a Fruchterman-Reingold layout for engine igraph and "dot"/"neato" for graphviz)

"arrowSize" [0,1]

"alpha" alpha transparency value (default .8; set to 1 for no transparency)

For the igraph engine the used plot function is `plot.igraph` in **igraph**. For graphviz the function `plot` in **Rgraphviz** is used. Note that Rgraphviz is available at http://www.bioconductor.org/. For the interactive version `tkplot` in **igraph** is always used.

`...` arguments are passed on to the respective plotting function (use for color, etc.).

"doubledecker", "mosaic" Represents a single rule as a doubledecker or mosaic plot. Parameter data has to be specified to compute the needed contingency table. Available control parameters are:

"main" plot title

"paracoord" Represents the rules (or itemsets) as a parallel coordinate plot. Currently there is no interactive version available. Available control parameters are:

"main" plot title

"reorder" reorder to minimize crossing lines.

"alpha" alpha transparency value

"iplots" Experimental interactive plots (package **iplots**) which support selection, highlighting, brushing, etc. Currently plots a scatterplot (support vs. confidence) and several histograms. Interactive manipulations are available.

Value

Several interactive plots return a set of selected rules/itemsets. Other plots might return other data structures. For example graph-based plots return the graph (invisibly).

Author(s)

Michael Hahsler and Sudheer Chelluboina. Some visualizations are based on the implementation by Martin Vodenicharov.

References

Bruzzese, D. and Davino, C. (2008), Visual Mining of Association Rules, in Visual Data Mining: Theory, Techniques and Tools for Visual Analytics, Springer-Verlag, pp. 103–122.

Hahsler, M. and Chelluboina, S. (2011), Visualizing association rules in hierarchical groups. Presented at the 42nd Symposium on the Interface: Statistical, Machine Learning, and Visualization Algorithms (Interface 2011). The Interface Foundation of North America.

Hahsler, M. and Karpienko, R. (2016) Visualizing Association Rules in Hierarchical Groups. *Journal of Business Economics,* accepted for publication, 2016.

See Also

`plotly_arules`, `scatterplot3d` in **scatterplot3d**, `plot.igraph` and `tkplot` in **igraph**, `seriate` in **seriation**

Examples

```
data(Groceries)
rules <- apriori(Groceries, parameter=list(support=0.005, confidence=0.5))
rules

## Scatterplot
plot(rules)
## try: sel <- plot(rules, interactive=TRUE)

## Scatterplot with custom colors
library(colorspace) # for sequential_hcl
plot(rules, control = list(col=sequential_hcl(100)))
plot(rules, control = list(col=grey.colors(50, alpha =.8)))

## see all control options
plot(rules, control = list(verbose = TRUE))

## Two-key plot is a scatterplot with shading = "order"
plot(rules, shading="order", control = list(main = "Two-key plot",
  col=rainbow(max(size(rules))-1L)))

## The following techniques work better with fewer rules
subrules <- subset(rules, lift>2.5)
subrules

## 2D matrix with shading
plot(subrules, method="matrix", measure="lift")
plot(subrules, method="matrix", measure="lift", control=list(reorder=TRUE))

## 3D matrix
```

```
plot(subrules, method="matrix3D", measure="lift")
plot(subrules, method="matrix3D", measure="lift", control=list(reorder=TRUE))

## matrix with two measures
plot(subrules, method="matrix", measure=c("lift", "confidence"))
plot(subrules, method="matrix", measure=c("lift", "confidence"),
control=list(reorder=TRUE))

## try: plot(subrules, method="matrix", measure="lift", interactive=TRUE,
## control=list(reorder=TRUE))

## grouped matrix plot
plot(rules, method="grouped")
plot(rules, method="grouped",
  control = list(col = grey.colors(10),
    gp_labels= gpar(col = "blue", cex=1, fontface="italic")))
## try: sel <- plot(rules, method="grouped", interactive=TRUE)

## graphs only work well with very few rules
subrules2 <- sample(rules, 10)
plot(subrules2, method="graph")
plot(subrules2, method="graph",
  control = list(nodeCol = grey.colors(10), edgeCol = grey(.7), alpha = 1))
## igraph layout generators can be used (see ? igraph::layout_)
plot(subrules2, method="graph", control=list(layout=igraph::in_circle()))
plot(subrules2, method="graph", control=list(
  layout=igraph::with_graphopt(spring.const=5, mass=50)))

plot(subrules2, method="graph", control=list(type="itemsets"))
## try: plot(subrules2, method="graph", interactive=TRUE)
## try: plot(subrules2, method="graph", control=list(engine="graphviz"))


## parallel coordinates plot
plot(subrules2, method="paracoord")
plot(subrules2, method="paracoord", control=list(reorder=TRUE))

## Doubledecker plot only works for a single rule
oneRule <- sample(rules, 1)
inspect(oneRule)
plot(oneRule, method="doubledecker", data = Groceries)

## use iplots (experimental)
## try: sel <- plot(rules, method="iplots", interactive=TRUE)


## for itemsets
itemsets <- eclat(Groceries, parameter = list(support = 0.02, minlen=2))
plot(itemsets)
plot(itemsets, method="graph")
plot(itemsets, method="paracoord", control=list(alpha=.5, reorder=TRUE))

## add more quality measures to use for the scatterplot
```

```
quality(itemsets) <- interestMeasure(itemsets, trans=Groceries)
head(quality(itemsets))
plot(itemsets, measure=c("support", "allConfidence"), shading="lift")
```

`plotly_arules` *Interactive Scatter Plot for Association Rules using plotly*

Description

Plot an interactive scatter plot for association rules using **plotly**.

Usage

```
plotly_arules(x, method = "scatterplot", measure = c("support", "confidence"),
  shading = "lift", max = 1000, ...)
```

Arguments

`x`	an object of class "rules".
`method`	currently the methods "scatterplot" and "matrix" are supported.
`measure`	measure(s) of interestingness (e.g., "support", "confidence", "lift", "order") used in the visualization as x and y-axis.
`shading`	measure of interestingness used for color shading.
`max`	client side processing in plotly is expensive. We restrict the number of rules to the max best rules (according to the measure used for shading.)
`...`	colors can be used to specify a color palette. Further arguments are passed on to `plot_ly()` as markter attributes (e.g., size, symbol and opacity).

Value

The plotly object (plotly_hash).

Examples

```
## Not run:
library(plotly)
data(Groceries)
rules <- apriori(Groceries, parameter=list(support=0.005, confidence=0.5))
rules

### interactive scatter plot visualization
plotly_arules(rules)
plotly_arules(rules, measure = c("support", "lift"), shading = "order")

# add a title
plotly_arules(rules)
```

```
# save a plot as a html page
p <- plotly_arules(rules)
htmlwidgets::saveWidget(as.widget(p), "arules.html")

### Interactive matrix visualization
plotly_arules(rules, method = "matrix")

## End(Not run)
```

---

saveAsGraph *Save rules or itemsets as a graph description.*

Description

This function saves a rule sat as a graph description in different formats (e.g., GraphML, dimacs, dot).

Usage

```
saveAsGraph(x, file, type="items", format="graphml")
```

Arguments

x	an object of class "rules" or "itemsets".
file	file name
type	see type in plot with method "graph" (e.g., "itemsets", "items").
format	file format (e.g., "edgelist", "graphml", "dimacs", "gml", "dot"). See `write.graph` in package **igraph**.

Author(s)

Michael Hahsler

See Also

`plot`, `write.graph` in **igraph**

Examples

```
data("Groceries")
rules <- apriori(Groceries, parameter=list(support=0.01, confidence=0.5))

saveAsGraph(rules, "rules.graphml")

## clean up
unlink("rules.graphml")
```

Index