

Package ‘arulesViz’

May 21, 2021

Version 1.5-0

Date 2021-5-21

Title Visualizing Association Rules and Frequent Itemsets

Depends arules (>= 1.6.0)

Imports graphics, methods, utils, grDevices, stats, seriation,
colorspace, grid, vcd, igraph, scatterplot3d, ggplot2,
ggnetwork, tibble, tidyr, dplyr, DT, plotly, visNetwork

Suggests graph, datasets, Rgraphviz, shiny, shinythemes, htmlwidgets

Description Extends package 'arules' with various visualization techniques for association rules and itemsets. The package also includes several interactive visualizations for rule exploration. Michael Hahsler (2017) <doi:10.32614/RJ-2017-047>.

License GPL-3

URL <https://github.com/mhahsler/arulesViz>

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

Copyright (C) 2018 Michael Hahsler, Tyler Giallanza and Sudheer Chelluboina

NeedsCompilation no

Author Michael Hahsler [aut, cre, cph],
Giallanza Tyler [ctb],
Sudheer Chelluboina [ctb]

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

Repository CRAN

Date/Publication 2021-05-21 19:30:02 UTC

R topics documented:

associations2igraph	2
inspectDT	3
plot	4
ruleExplorer	10
rules2matrix	11

associations2igraph *Convert rules or itemsets into a graph*

Description

Function to convert associations (rules, itemsets) into a igraph object and saves the graph in different formats (e.g., GraphML, dimacs, dot).

Usage

```
associations2igraph(x)
saveAsGraph(x, file, format="graphml")
```

Arguments

x	an object of class "rules" or "itemsets".
file	file name.
format	file format (e.g., "edgelist", "graphml", "dimacs", "gml", "dot"). See <code>write.graph</code> in package igraph .

Value

associations2igraph returns an igraph object.

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))

# convert rules into a graph
g <- associations2igraph(rules)
g

plot(g)

# save rules as a graph so they can be visualized using external tools
saveAsGraph(rules, "rules.graphml")

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

`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` in package **DT**.

Value

A `datatable` `htmlwidget`.

Author(s)

Michael Hahsler

References

Hahsler M (2017). `arulesViz`: Interactive Visualization of Association Rules with R. *R Journal*, 9(2):163-175. ISSN 2073-4859. doi: [10.32614/RJ2017047](https://doi.org/10.32614/RJ2017047).

See Also

[datatable](#) in **DT**.

Examples

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

inspectDT(rules)

# for more control on the data table, you can used DATAFRAME() to convert the rules.
rules_df <- DATAFRAME(rules, setStart = '', setEnd = '', itemSep = ' + ')
rules_df$count <- NULL
head(rules_df)
```

```
inspectDT(rules_df)

# save a datatable as a html page.
p <- inspectDT(rules)
htmlwidgets::saveWidget(p, "arules.html", selfcontained = FALSE)
browseURL("arules.html")
## End(Not run)
```

plot

Visualize Association Rules and Itemsets

Description

Methods (S3) 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.

Many plots can use different rendering engines including static standard plots (using base plots, **ggplot2**, **grid**), standard plots with interactive manipulation and interactive HTML widget-based visualizations.

Usage

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

Arguments

x	an object of class "rules" or "itemsets".
method	a string indicating the visualization method. Methods for rules include "scatterplot", "two-key plot", "matrix", "grouped matrix", "graph", "paracoord", etc. Specify "help" to get a complete list of available methods. Note that some methods may only be available for rules or itemsets.
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	deprecated. See parameter engine below.

engine	a string indicating the plotting engine used to render the plot. The "default" engine uses (mostly) ggplot2 . Other engines include "base" (base R plots), "grid", "interactive", "plotly", "visnetwork", "igraph", "graphviz", and "htmlwidget". Note that not all engines are available for all methods. Specify "help" to get a complete list of available engines for the selected visualization method.
data	the dataset (class "transactions") used to generate the rules/itemsets. Only "mosaic" and "doubledecker" require the original data.
control	a list of control parameters for the plot. The available control parameters depend on the used visualization method and engine. Specify "help" to get a complete list of available control parameters and their default values.
...	Further arguments are added for convenience to the control list.

Details

Most visualization techniques are described by Bruzese and Davino (2008), however, we added more color shading, reordering and interactive features (see Hahsler, 2017). Many visualization methods take extra parameters as the control parameter list. Although, we have tried to keep control parameters consistent, the available control parameters vary from visualization method to visualization method. You can specify "help" for method, engine, or control to get a list of available settings.

Note on HTML widgets: HTML widgets tend to get very slow or unresponsive for too many rules. To prevent this situation, the control parameter `max` sets a limit, and the user is warned if the limit is reached.

The following visualization methods 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.

"matrix" Arranges the association rules as a matrix with the itemsets in the antecedents on one axis and the itemsets in the consequents on the other. The measure of interestingness (first element of measure) is either visualized by a color (darker means a higher value for the measure) or as the height of a bar (engine "3d"). The control parameter `reorder` takes the values "none", "measure", "support/confidence", or "similarity" and can be used to reorder LHS and RHS of the rules differently. The default reordering average measure (typically lift) pushing the rules with the highest lift value to the top-left corner of the plot.

"grouped matrix" Grouped matrix-based visualization (Hahsler and Karpienko, 2016; Hahsler 2016). Antecedents (columns) in the matrix are grouped using clustering. Groups are represented by the most interesting item (highest ratio of support in the group to support in all rules) in the group. Balloons in the matrix are used to represent with what consequent the antecedents are connected.

Interactive manipulations (zooming into groups and identifying rules) are available.

The list of control parameters for this method includes:

"main" plot title

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

"rhs_max" maximal number of RHSs to show. The rest are suppressed. (default: 10)
"lhs_items" number of LHS items shown (default: 2)
"aggr.fun" aggregation function can be any function computing a scalar from a vector (e.g., min, mean (default), median, sum, max). It is also used to reorder the balloons in the plot.
"col" color palette (default is 100 heat colors.)

"graph" Represents the rules (or itemsets) as a graph with items as labeled vertices, and rules (or itemsets) represented as vertices connected to items using arrows. For rules, the LHS items are connected with arrows pointing to the vertex representing the rule and the RHS has an arrow pointing to the item.

"doubledecker", **"mosaic"** Represents a single rule as a doubledecker or mosaic plot. Parameter data has to be specified to compute the needed contingency table. No interactive version is available.

"paracoord" Represents the rules (or itemsets) as a parallel coordinate plot. Currently there is no interactive version 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). Engine "htmlwidget" always returns an object of class `htmlwidget`.

Author(s)

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

References

- Hahsler M (2017). `arulesViz`: Interactive Visualization of Association Rules with R. *R Journal*, 9(2):163-175. ISSN 2073-4859. doi: [10.32614/RJ2017047](https://doi.org/10.32614/RJ2017047).
- 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. doi: [10.1007/9783540710806](https://doi.org/10.1007/9783540710806)
- Hahsler, M. and Karpienko, R. (2016), Visualizing Association Rules in Hierarchical Groups. *Journal of Business Economics*, 87(3):17-335. doi: [10.1007/s1157301608228](https://doi.org/10.1007/s1157301608228)
- Hahsler, M. (2016), Grouping association rules using lift. In C. Iyigun, R. Moghaddess, and A. Oztekin, editors, 11th INFORMS Workshop on Data Mining and Decision Analytics (DM-DA 2016).

See Also

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

Examples

```

data(Groceries)
rules <- apriori(Groceries, parameter=list(support = 0.001, confidence = 0.8))
rules

## Getting help
# There are many method, plotting engines and all of them have different control parameters. Use
# "help" to get help. List available methods for the object rules:
plot(rules, method = "help")

# List the available engines for method "scatterplot"
plot(rules, method = "scatterplot", engine = "help")

# List control parameters for scatterplot with engine "ggplot2"
## Not run: plot(rules, method = "scatterplot", engine = "ggplot2", control = "help")

## Scatter plot
# Display a scatter plot using two quality measures
plot(rules)

# Scatter plot with custom measures
plot(rules, measure = c("support", "lift"), shading = "confidence")

# Custom color scale, , labels, theme and no title (ggplot2)
library(ggplot2)
plot(rules, engine = "ggplot2", main = NULL) +
  scale_color_gradient2(low = "red", mid = "gray90", high = "blue",
    midpoint = 1, limits = c(0,12)) +
  labs(x = "Supp.", y = "Conf.", color = "Lift") +
  theme_classic()

# Scatter plot using other engines
plot(rules, engine = "grid")
library(colorspace) # for sequential_hcl
plot(rules, engine = "grid", control = list(col=sequential_hcl(100)))

plot(rules, engine = "base")

# Interactive scatter plot using the grid engine (selected rules are returned)
## Not run: sel <- plot(rules, engine = "interactive")

# Create a html widget for interactive visualization (uses plotly)
plot(rules, engine = "htmlwidget")

# Two-key plot (a scatter plot with shading = "order")
plot(rules, method = "two-key plot")

## Matrix shading
# Display rules as a matrix with RHS itemsets as rows and LHS itemsets as columns

```

```

# works better with small sets of rules
subrules <- subset(rules, lift > 5)
subrules

# 2D matrix with shading (ggplot2). The LHS and RHS are reordered so
# that rules with similar lift are displayed close to each other.
plot(subrules, method = "matrix")

# Reorder rules differently
plot(subrules, method = "matrix", control = list(reorder = "none"))
plot(subrules, method = "matrix", control = list(reorder = "support/confidence"))
plot(subrules, method = "matrix", control = list(reorder = "similarity"))

# Use other engines
plot(subrules, method = "matrix", engine = "grid")
plot(subrules, method = "matrix", engine = "base")
plot(subrules, method = "matrix", engine = "3d")

# Interactive matrix plot
# * Engine interactive: identify rules by clicking on them (click outside to end)
# * Engine htmlwidget: hover over rules to identify
## Not run: plot(subrules, method = "matrix", engine = "interactive")
plot(subrules, method = "matrix", engine = "htmlwidget")

## Grouped matrix plot
# Default engine is ggplot2
plot(rules, method="grouped matrix")
plot(rules, method="grouped matrix", shading = "confidence", k = 5)

# Create a htmlwidget
plot(rules, method = "grouped matrix", engine = "htmlwidget")

# Engine grid
plot(rules, method="grouped matrix", engine = "grid")
plot(rules, method="grouped matrix", engine = "grid",
      col = grey.colors(10),
      gp_labels = grid::gpar(col = "blue", cex=1, fontface="italic"))

# Interactive grouped matrix plot
## Not run: sel <- plot(rules, method="grouped matrix", engine = "interactive")

## Graph representation

# Graphs only work well with very few rules (we sample a few of the high-lift rules)
subrules2 <- sample(subset(rules, lift > 5), 25)

# Default engine is ggplot2 with ggnetwork
plot(subrules2, method = "graph")

# Specify layout (from igraph) and edge and node representation
# Note: The measures for size and color shading have to be specified in
# the control list in nodes!

```

```

plot(subrules2, method="graph",
     control = list(
       layout = igraph::with_graphopt(),
       edges = ggnetwork::geom_edges(color = "grey80",
         arrow = arrow(length = unit(5, "pt"), type = "open"), alpha = .5),
       nodes = ggnetwork::geom_nodes(aes(size = support, color = lift), na.rm = TRUE),
       nodetext = ggnetwork::geom_nodetext(aes(label = label), alpha = .6)
     )) +
scale_color_gradient(low = "yellow", high = "red") +
scale_size(range = c(2, 15))

# Engine igraph
plot(subrules2, method = "graph", engine = "igraph")

# Custom colors
plot(subrules2, method = "graph", engine = "igraph",
     nodeCol = grey.colors(10), edgeCol = grey(.7), alpha = 1)

# No shading for lift, set node color to gray and add
# labels for support
plot(subrules2, method = "graph", engine = "igraph",
     shading = NA, nodeCol = grey(.5), measureLabels = TRUE)

# Use plot_options to alter any aspect of the graph
# (see: https://igraph.org/r/doc/plot.common.html)
plot(subrules2, method = "graph", engine = "igraph",
     plot_options = list(
       edge.lty = 2,
       vertex.label.cex = .6,
       margin = c(.1,.1,.1,.1),
       asp = .5))

# igraph layout generators can be used (see ? igraph::layout_)
plot(subrules2, method="graph", engine = "igraph", layout=igraph::in_circle())
plot(subrules2, method="graph", engine = "igraph",
     layout = igraph::with_graphopt(spring.const = 5, mass = 50))

# Graph rendering using engine graphviz
## Not run: plot(subrules2, method = "graph", engine = "graphviz")

# Default interactive plot (using igraph's tkplot)
## Not run: plot(subrules2, method = "graph", engine = "interactive")

# Interactive graph as a html widget (using igraph layout)

plot(subrules2, method = "graph", engine = "htmlwidget")
plot(subrules2, method = "graph", engine = "htmlwidget",
     igraphLayout = "layout_in_circle")

## Parallel coordinates plot
plot(subrules2, method = "paracoord")

```

```
## Doubledecker and mosaic plot
# Uses functions in package vcd
# Notes: doubledecker and mosaic plots only visualize a single rule
# and the transaction set is needed.
oneRule <- sample(rules, 1)
inspect(oneRule)
plot(oneRule, method = "doubledecker", data = Groceries)
plot(oneRule, method = "mosaic", data = Groceries)

## Visualizing itemsets
data(Groceries)
itemsets <- eclat(Groceries, parameter = list(support = 0.02, minlen = 2))

# default is a scatter plot with ggplot2
plot(itemsets)
plot(itemsets, engine = "grid")
plot(itemsets, engine = "base")

plot(itemsets, method = "graph")
plot(itemsets, method = "graph", engine = "igraph")

plot(itemsets, method = "paracoord", alpha = .5)

## Add more quality measures to use for the scatter plot

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

## Save HTML widget as web page
## Not run:
p <- plot(rules, engine = "html")
htmlwidgets::saveWidget(p, "arules.html", selfcontained = FALSE)
browseURL("arules.html")
## End(Not run)
# Note: self-contained seems to make the browser slow.
```

ruleExplorer

Explore Association Rules Interactively

Description

Explore association rules using interactive manipulations and visualization using **shiny**.

Usage

```
ruleExplorer(x, sidebarWidth = 2, graphHeight = '600px')
```

Arguments

x	a set of rules, a transactions object or a data.frame.
sidebarWidth	width of the sidebar as a number between 0 (= 0% of the display width) and 12 (= 100% of the display width).
graphHeight	height of the plots in pixels. Increase if you have a larger/higher resolution display.

Author(s)

Tyler Giallanza and Michael Hahsler. Adapted from functions originally created by Andrew Brooks. See <https://github.com/brooksandrew/Rsenal> for the original code.

References

Hahsler M (2017). arulesViz: Interactive Visualization of Association Rules with R. *R Journal*, 9(2):163-175. ISSN 2073-4859. doi: [10.32614/RJ2017047](https://doi.org/10.32614/RJ2017047).

See Also

[plot](#) with engine = "html", [inspectDT](#), [apriori](#).

Examples

```
## Not run:
data(Groceries)

# explore pre-mined rules
rules <- apriori(Groceries, parameter=list(support=0.001, confidence=0.8))
rules

ruleExplorer(rules)

# mine and explore rules on the fly
data(iris)
ruleExplorer(iris)

## End(Not run)
```

rules2matrix

Convert association rules into a matrix

Description

Converts a set of association rules into a matrix with unique LHS itemsets as columns and unique RHS itemsets as rows. The matrix cells contain a quality measure. The LHS itemsets can be grouped.

Usage

```
rules2matrix(rules, measure = "support", reorder = "measure", ...)  
rules2groupedMatrix(rules, measure = "lift", measure2 = "support", k = 10,  
  aggr.fun = mean, lhs_label_items = 2)
```

Arguments

rules	a rules object.
measure	quality measure put in the matrix
reorder	reorder rows and columns? Possible methods are: "none", "measure" (default), "support/confidence", "similarity".
measure2	second quality measure (organized in the same way as measure).
k	number of LHS itemset groups.
aggr.fun	function to aggregate the quality measure for groups.
lhs_label_items	number of top items used to name LHS itemset groups (columns).
...	passed on to DATAFRAME .

Value

rules2matrix returns a matrix with quality values.

rules2groupedMatrix returns a list with elements

m the grouped matrix for measure.

m2 the grouped matrix for measure2.

clustering_rules
vector with group assignment for each rule.

Author(s)

Michael Hahsler

References

Michael Hahsler and Radoslaw Karpienko. Visualizing association rules in hierarchical groups. Journal of Business Economics, 87(3):317–335, May 2016. doi: [10.1007/s1157301608228](https://doi.org/10.1007/s1157301608228).

See Also

[plot](#) for rules using method = 'matrix' and method = 'grouped matrix'.

Examples

```
data(Groceries)
rules <- apriori(Groceries, parameter=list(support = 0.001, confidence = 0.8))
rules

## Matrix
m <- rules2matrix(rules[1:10], measure = "lift")
m
plot(rules[1:10], method = "matrix")

## Grouped matrix
# create a matrix with LHSs grouped in k = 10 groups
m <- rules2groupedMatrix(rules, k = 10)
m$m

# number of rules per group
table(m$clustering_rules)

# get rules for group 1
inspect(rules[m$clustering_rules == 1])

# the corresponding plot
plot(rules, method = "grouped matrix", k = 10)
```

Index

- * **file**
 - associations2igraph, 2
- * **hplot**
 - plot, 4
- * **print**
 - inspectDT, 3

- apriori, 11
- associations2igraph, 2

- DATAFRAME, 12
- datatable, 3
- datatable (inspectDT), 3

- explore (ruleExplorer), 10

- inspect (inspectDT), 3
- inspectDT, 3, 11

- plot, 2, 4, 11, 12
- plot.igraph, 6
- plotly (plot), 4

- ruleExplorer, 10
- rules2groupedMatrix (rules2matrix), 11
- rules2matrix, 11

- saveAsGraph (associations2igraph), 2
- scatterplot3d, 6
- seriate, 6

- tkplot, 6

- write.graph, 2