Package: arulesViz (via r-universe)

October 6, 2026

Title Visualizing Association Rules and Frequent Itemsets

Version 1.5.4

Date 2025-08-20

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, http://michael.hahsler.net/arulesViz/

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

Depends arules (>= 1.6.0)

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

Suggests datasets, graph, htmlwidgets, Rgraphviz, shiny, shinythemes, testthat (>= 3.0.0), tidygraph, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Copyright (C) 2021 Michael Hahsler

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev

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

Date/Publication 2026-10-04 17:01:13 UTC

RemoteUrl https://github.com/mhahsler/arulesViz

RemoteRef HEAD

RemoteSha 1dc960a0f842218fd46ace7f645f66017bad96a9

Contents

associations2igraph 2
inspectDT 3
plot_arulesViz 5
ruleExplorer 12
rules2groupedMatrix 13

Index **16**

`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, associationsAsNodes = TRUE)

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

Arguments

`x`	an object of class "rules" or "itemsets".
`associationsAsNodes`	should associations be translated into nodes or represented by edges?
`file`	file name.
`format`	file format (e.g., "edgelist", "graphml", "dimacs", "gml", "dot"). See `igraph::write_graph()`.
`...`	further arguments are passed on to `associations2igraph()`.

Details

Associations are represented as nodes: All items in the associations are connected to the association node. For itemsets, the wdges are undirected, for rules, the edges are directed towards the rhs

When associations are represented as edges: For rules, each item in the LHS is connected with a directed edge to the item in the RHS. For itemsets, undirected edges for each pair of item in the itemset are created.

Value

`associations2igraph` returns an igraph object.

Author(s)

Michael Hahsler

Examples

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

# convert rules into a graph with rules as nodes
library("igraph")
g <- associations2igraph(rules)
g

plot(g)

# convert the graph into a tidygraph
library("tidygraph")
as_tbl_graph(g)

# convert the generating itemsets of the rules into a graph with itemsets as edges
itemsets <- generatingItemsets(rules)
itemsets
g <- associations2igraph(itemsets, associationsAsNodes = FALSE)
g

plot(g, layout = layout_in_circle)

# 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, ...)

## Default S3 method:
inspectDT(x, ...)

## S3 method for class 'rules'
inspectDT(x, precision = 3, ...)

## S3 method for class 'itemsets'
```

```
inspectDT(x, precision = 3, ...)

## S3 method for class 'data.frame'
inspectDT(x, precision = 3, ...)
```

Arguments

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

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.

See Also

DT::datatable() in **DT**.

Examples

```
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 HTML widget as web page
p <- inspectDT(rules)
htmlwidgets::saveWidget(p, "arules.html", selfcontained = FALSE)
# Note: self-contained seems to make the browser slow.

# inspect the widget
browseURL("arules.html")

# clean up
```

```
unlink(c("arules.html", "arules_files"), recursive = TRUE)
```

---

plot_arulesViz *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.

Usage

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

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

## S3 method for class 'grouped_matrix'
plot(x, ...)
```

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". NULL can be often used to suppress shading.
limit	A limit on the number of associations displayed. The top limit associations according to the measure specified in shading are chosen.
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" (which can be embedded in RMarkdown). 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

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.

Most visualization techniques are described by Bruzzese 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 specift "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 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.

"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 differntly. 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

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

Hahsler, M. and Karpienko, R. (2016), Visualizing Association Rules in Hierarchical Groups. *Journal of Business Economics,* 87(3):17-335. doi: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::scatterplot3d()`, `igraph::plot.igraph()` and `igraph::tkplot()`, `seriation::seriate()`.

Examples

```
# Note: To speed example execution, not all examples are not run when using example("plot").
# Use example("plot") to run all 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")

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

## End(Not run)


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

# Scatter plot with custom measures and limiting the plot to the 100 with the
# largest value for for the shading measure.
plot(rules, measure = c("support", "lift"), shading = "confidence", limit = 100)

## Not run:
# Custom color scale, labels, theme and no title (ggplot2)
```

```
library(ggplot2)
plot(rules, engine = "ggplot2", main = NULL, limit = 100) +
  scale_color_gradient2(
    low = "red", mid = "gray90", high = "blue",
    midpoint = 1, limits = c(0, 12)
  ) +
  labs(x = "Supp.", y = "Conf.", color = "Lift") +
  theme_classic()

# Interactive scatter plot using the grid engine (selected rules are returned)
if (interactive()) {
  sel <- plot(rules, engine = "interactive")

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

## End(Not run)

# Two-key plot (a scatter plot with shading = "order")
plot(rules, method = "two-key plot", limit = 100)


## 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")

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

## End(Not run)


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

## Not run:
# Create a htmlwidget
plot(rules, method = "grouped matrix", engine = "htmlwidget")
```

```
# Interactive grouped matrix plot
if (interactive()) {
  sel <- plot(rules, method = "grouped matrix", engine = "interactive")
}

## End(Not run)

## Graph representation
# Default engine is ggplot2 with ggraph. Associations are represented as nodes.
# We limit the number of rules to the 10 with the larges
# lift (measure used for shading)
plot(subrules, method = "graph", limit = 10)

## Not run:
# Circular layout (see? ggraph for the meaning of the arguments)
plot(subrules, method = "graph", layout = "linear", circular = TRUE, limit = 10)

# Use igraph layouts (algorithm is passes on as ... to ggraph)
plot(subrules,
  method = "graph", layout = "igraph",
  ggraphdots = list(algorithm = "graphopt", spring.const = 1, mass = 10), limit = 10
)

# Specify edge and node representation
library(ggplot2)
plot(subrules,
  method = "graph",
  control = list(
    edges = ggraph::geom_edge_link(
      end_cap = ggraph::circle(4, "mm"),
      start_cap = ggraph::circle(4, "mm"),
      color = "black",
      arrow = arrow(length = unit(2, "mm"), angle = 20, type = "closed"),
      alpha = .2
    ),
    nodes = ggraph::geom_node_point(aes(size = support, color = lift)),
    nodetext = ggraph::geom_node_label(aes(label = label), alpha = .8, repel = TRUE)
  ),
  limit = 10
) +
  scale_color_gradient(low = "yellow", high = "red") +
  scale_size(range = c(2, 10))

# ggplot also can represent associations as edges. Here a rules is represented as a set of
# arrows going from the LHS items to the RHS item.
plot(subrules, method = "graph", asEdges = TRUE, limit = 10)
plot(subrules, method = "graph", asEdges = TRUE, circular = FALSE, limit = 10)

## End(Not run)

# Engine igraph
plot(subrules, method = "graph", engine = "igraph", limit = 10)
```

```
plot(subrules,
  method = "graph", engine = "igraph",
  nodeCol = grey.colors(10), edgeCol = grey(.7), alpha = 1,
  limit = 10
)

# Use plot_options to alter any aspect of the graph
# (see: https://igraph.org/r/doc/plot.common.html)
plot(subrules,
  method = "graph", engine = "igraph",
  plot_options = list(
    edge.lty = 2,
    vertex.label.cex = .6,
    margin = c(.1, .1, .1, .1),
    asp = .5
  ),
  limit = 10
)

# igraph layout generators can be used (see ? igraph::layout_)
plot(subrules, method = "graph", engine = "igraph", layout = igraph::in_circle(), limit = 10)

## Not run:
# Graph rendering using engine graphviz
plot(subrules, method = "graph", engine = "graphviz", limit = 10)

if (interactive()) {
  # Default interactive plot (using igraph's tkplot)
  plot(subrules, method = "graph", engine = "interactive", limit = 10)

  # Interactive graph as a html widget (using igraph layout)
  plot(subrules, method = "graph", engine = "htmlwidget", limit = 10)
  plot(subrules,
    method = "graph", engine = "htmlwidget",
    igraphLayout = "layout_in_circle", limit = 10
  )
}

## End(Not run)

## Parallel coordinates plot
plot(subrules, method = "paracoord", limit = 10)


## 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)
```

```
## 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, method = "graph", limit = 10)

## Not run:
plot(itemsets, method = "graph", asEdges = TRUE, limit = 10)
plot(itemsets, method = "graph", asEdges = TRUE, circular = FALSE, limit = 10) +
  theme(plot.margin = margin(10, 10, 30, 20, "mm"))

## End(Not run)

plot(itemsets, method = "paracoord", alpha = .5, limit = 10)

# 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")

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

# inspect the widget
browseURL("arules.html")

# clean up
unlink(c("arules.html", "arules_files"), recursive = TRUE)

## End(Not run)
```

`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.

Value

returns a shiny app.

Author(s)

Tyler Giallanza and Michael Hahsler. Adapted from functions originally created by Andrew Brooks. See 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.

See Also

`plot()` with `engine = "html"`, `inspectDT()`, `arules::apriori()`.

Examples

```
# this example can only be run manually from the console.

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

ruleExplorer(rules)

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

---

rules2groupedMatrix *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

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

rules2matrix(rules, measure = "support", reorder = "measure", ...)
```

Arguments

rules	a rules object.
measure	quality measure put in the matrix
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).
reorder	reorder rows and columns? Possible methods are: "none", "measure" (default), "support/confidence", "similarity".
...	passed on to arules::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.

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
gm <- rules2groupedMatrix(rules, k = 10)
gm$m

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

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

# create the corresponding grouped matrix plot by passing the grouped matrix as the groups parameter
plot(rules, method = "grouped matrix", groups = gm)
```

Index

∗ **file**
associations2igraph, 2
∗ **hplot**
plot_arulesViz, 5
∗ **print**
inspectDT, 3

arules::apriori(), *13*
arules::DATAFRAME(), *14*
associations2igraph, 2

datatable *(*inspectDT*)*, 3
DT::datatable(), *4*

explore *(*ruleExplorer*)*, 12

graph *(*associations2igraph*)*, 2
guide_edge_colourbar *(*plot_arulesViz*)*, 5

igraph *(*associations2igraph*)*, 2
igraph::plot.igraph(), *8*
igraph::tkplot(), *8*
igraph::write_graph(), *2*
inspect *(*inspectDT*)*, 3
inspectDT, 3
inspectDT(), *13*

plot *(*plot_arulesViz*)*, 5
plot(), *13, 14*
plot.grouped_matrix *(*plot_arulesViz*)*, 5
plot.itemsets *(*plot_arulesViz*)*, 5
plot.rules *(*plot_arulesViz*)*, 5
plot_arulesViz, 5
plotly *(*plot_arulesViz*)*, 5

ruleExplorer, 12
rules2groupedMatrix, 13
rules2matrix *(*rules2groupedMatrix*)*, 13

saveAsGraph *(*associations2igraph*)*, 2
scatterplot3d::scatterplot3d(), *8*
seriation::seriate(), *8*

tidygraph *(*associations2igraph*)*, 2