Package: quickdag (via r-universe)

March 12, 2025

Title Make Beautiful Directed Acyclic Graphs

Version 0.2.0

Description This package allows the user the save directed acyclic graphs (DAGs) generated in DiagrammeR and export them to pdf, png, or svg format. Users may optionally choose to view the DAG without saving or to embed it in an RMarkdown document. In addition, quickDAG allows the user easily to reformat DAGs as single-world intervention graph (SWIG) templates.

Depends R (>= 3.5), DiagrammeR

Imports DiagrammeRsvg, rsvg, knitr, dagitty, stringr, purrr, dplyr, htmlTable, messaging

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.2

URL `https://github.com/jrgant/quickDAG`

BugReports `https://github.com/jrgant/quickDAG/issues`

Suggests testthat

Config/pak/sysreqs libglpk-dev make libicu-dev librsvg2-dev libxml2-dev libssl-dev libnode-dev libx11-dev

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

RemoteUrl https://github.com/jrgant/quickdag

RemoteRef HEAD

RemoteSha 6ea3c3baf82d8e1027db0806ac745a4c5b28a927

Contents

makeDAG 2
qd_dag 3
qd_embed 5

qd_save 5
qd_swig 6
qd_themes 7
qd_todagitty 7
sep_opts 9

Index **10**

makeDAG *Output, view, or embed DAGs.*

Description

Save directed acyclic graphs (DAGs) generated in DiagrammeR and export them to pdf, png, or svg format. Users may optionally choose to view the DAG without saving or to embed it in an RMarkdown document.

Usage

```
makeDAG(
  graphcode = NULL,
  dagname = NULL,
  filetype = "pdf",
  text.nodes = NULL,
  box.nodes = NULL,
  solid.edges = NULL,
  dashed.edges = NULL,
  footnotes = NULL,
  direction = "LR",
  embed = FALSE,
  ...
)
```

Arguments

`graphcode`	Partial graphViz code object, which will give you the most control over the appearance of your DAG.
`dagname`	A path with which to name your DAG file, if `filetype` set to output format (see below).
`filetype`	Output file format. Select from `pdf`, `png`, `svg`, or `view`. Setting the option to `view` will not save a file but will generate the diagram in your viewer. Defaults to `pdf`.
`text.nodes`	A string containing the plain text nodes, separated by spaces.
`box.nodes`	A string containing the boxed nodes, separated by spaces.
`solid.edges`	A string specifying the paths you want to draw between measured covariates. Example: `"Alcohol -> Smoking Smoking -> Cancer"`.

`dashed.edges`	A string specifying paths containing unmeasured covariates.
`footnotes`	Add a footnote to the bottom of the graph.
`direction`	Specify the direction of diagram flow. Defaults to `"LR"`, for left-right.
`embed`	For use within R chunks in RMarkdown only. You will probably want to have the `echo` chunk option set to `FALSE`, unless you want to display the R code itself. The `embed` defaults to `TRUE`.
`...`	Pass arguments to interior functions for PNG or SVG files. For instance, specify `height` and `width` arguments.

Note

This is not a true DAG package in the sense that it will not prevent the inclusion of feedback loops or bidirectional arrows. It's meant mostly to create reasonable-looking DAGs quickly and easily with a minimum of layout or formatting code. DiagrammeR in general does a pretty good job at layout. Those interested in DAGs might check out other R packages like dagR or dagitty, both of which I've yet to explore in detail.

Suggestions and issue reports welcome at `https://github.com/jrgant/quickDAG/issues`!

References

A fair amount of the heavy lifting here is done thanks to code snippets from users HJAllen and puterleat on the following thread: `https://github.com/rich-iannone/DiagrammeR/issues/133`

Packages used: DiagrammeR, DiagrammeRsvg, rsvg

Examples

```
# Using your own graph code

# Examples have been removed, as makeDAG() is now deprecated.
```

`qd_dag` *Generate a graph object*

Description

Provide simple syntax specifying paths between nodes to generate a graph object.

Usage

```
qd_dag(
  edgelist,
  node.labs = NULL,
  node.aes.opts = list(),
  edge.aes.opts = list(),
  verbose = TRUE,
  check.dag = TRUE,
  theme = "base",
  ...
)
```

Arguments

`edgelist`	A vector of edge relationships. Must be strictly organized (see example for format).
`node.labs`	A named character vector containing label names. Defaults to `NULL`.
`node.aes.opts`	A list feeding aesthetic options for nodes to `DiagrammeR::node_aes()`. Defaults to empty list. See `?node_aes` to view available parameters.
`edge.aes.opts`	A list feeding aesthetic options for edges to `DiagrammeR::edge_aes()`. Defaults to empty list. See `?edge_aes` to view available parameters.
`verbose`	Indicate whether to print node and edge dataframes to the console. See NOTE below. Defaults to `TRUE`.
`check.dag`	Logical. Check whether the graph conforms to the rules of DAGs. Defaults to `TRUE`.
`theme`	Choose theme for plot output. Defaults to "base". Setting theme to NULL will use DiagrammeR's NULL attribute theme.
`...`	Pass optional `conditioned` argument to qd_themes().

Details

Suggestions and bug reports welcome at https://github.com/jrgant/quickDAG/issues.

Packages used: DiagrammeR, stringr, purrr

Note

Leaving the checks option selected may be advisable to ensure labels and IDs have not been mismatched. By default, `qd_dag()` alphabetizes nodes included in `edgelist` and does the same for `node.labs` under a first assumption that labels will begin with the same letter as their corresponding `alpha.id`, which may not always be the case.

Examples

```
# Provide a list of edges, with nodes specified as letters.
# Do not list a node as a parent more than once.
# Each line should contain a single edge character '->'.
edges <- c("A -> { B C }",
           "B -> C")

# make a DAG object and render the graph using the default theme
g.obj <- qd_dag(edges)
render_graph(g.obj)

# Pass labels and aesthetic options for nodes or edges
```

```
g.obj2 <- qd_dag(edges,
                 node.labs = c("A" = "Alcohol",
                               "B" = "BP",
                               "C" = "CVD"),
                 node.aes.opts = list(shape = "plaintext",
                                      fillcolor = "none",
                                      color = "black"),
                 edge.aes.opts = list(arrowsize = 0.5,
                                      color = "gray"),
                 theme = NULL)
render_graph(g.obj2)
```

qd_embed — *Embed diagrams in RMarkdown*

Description

A wrapper around `qd_save()` meant for use within R code chunks in RMarkdown documents.

Usage

```
qd_embed(...)
```

Arguments

`...`	Pass arguments to `qd_save()`.

qd_save — *Save graph objects*

Description

Export high-quality, scalable graphics for both print and online.

Usage

```
qd_save(graph, filename = NULL, filetype = "pdf", embed = F, ...)
```

Arguments

`graph`	Either a DiagrammeR graph object or a diagram generated by `render_graph()`.
`filename`	String for filename. Defaults to `NULL`.
`filetype`	One of `pdf`, eps, `pdf`, `svg`. Defaults to `pdf`.
`embed`	Defaults to FALSE. Automatically set to TRUE by `qd_embed()`.
`...`	Pass arguments to `render_graph()`, e.g., width and height (pixels) for .png files.

Functions

- `qd_save()`:

`qd_swig` *Generate a single-world intervention graph (SWIG)*

Description

Take a DAG graph object and, in the simplest case, create a single-world intervention template corresponding to a world in which the fixed nodes are set to a given value. Alternatively, tell qd_swig which values fixed nodes will be set to.

Usage

```
qd_swig(
  graph.obj,
  fixed.nodes,
  custom.values = NULL,
  fixed.sep = "vlin",
  sep.point.size = 15
)
```

Arguments

`graph.obj`	A DAG object created by `qd_dag()`.
`fixed.nodes`	A vector containing the nodes to be intervened upon.
`custom.values`	A named vector containing alternative labels identifying explicit values for fixed nodes (e.g., a = 1).
`fixed.sep`	A character string indicating which character to use as a separator in fixed nodes. Defaults to "vlin". Run `sep_opts(T)` for available options.
`sep.point.size`	A numerical value specifying the point size for fixed node separators.

Examples

```
# Provide a DAG object and a list of nodes to be fixed
edges <- c("A -> Y",
           "L -> { A Y }")

dag  <- qd_dag(edges)

swig <- dag %>%
        qd_swig(fixed.nodes = "A",
                custom.values = c("A" = "1"))

swig %>% render_graph()
```

`qd_themes` *Diagram themes*

Description

Apply various pre-fabricated themes to diagrams.

Usage

```
qd_themes(graph.obj, theme, ...)

theme_base(graph.obj, font = "serif", ...)

theme_circles(graph.obj, font = "serif", ...)

theme_dots(graph.obj, font = "serif", ...)

get_conditioned_nodes(graph.obj, conditioned = NULL)
```

Arguments

`graph.obj`	A DAG object created by `qd_dag()`.
`theme`	A character string indicating the theme to use. Defaults to "base". Set to `NULL` to use GraphViz defaults.
`...`	Pass arguments to theme call (e.g., `theme_base()`), such as `conditioned` or `font`
`font`	A character vector indicating the font family to use for node labels. Defaults to "serif".
`conditioned`	A character vector indicating which nodes are conditioned upon. The shape for these nodes will be set to "square".

`qd_todagitty` *Identify variables for adjustment*

Description

Format an edgelist and send it to dagitty to identify variable adjustment sets.

Usage

```
qd_todagitty(
  edgelist,
  diagram_type = "dag",
  showplot = FALSE,
  send.global = FALSE,
  dagitty.obj.name = NULL,
  exposure,
  outcome,
  ...
)
```

Arguments

`edgelist`	A vector of edge relationships. Must be strictly organized (see example for format).
`diagram_type`	Character identifying the diagram type. Defaults to `"dag"`, but user can specify another graph type (see dagitty documentation).
`showplot`	Logical indicating whether to produce a dagitty plot. Defaults to `FALSE`.
`send.global`	Logical indicating whether to make the dagitty object available in the global environment. Defaults to `FALSE`.
`dagitty.obj.name`	Character specifying the name of the dagitty object. Only used and required if `send.global` = `TRUE`.
`exposure`	Character. Specify exposure of interest. (Required)
`outcome`	Character. Specifiy outcome of interest. (Required)
`...`	Pass arguments to `adjustmentSets()`. See dagitty documentation for options.

Note

The `exposure` and `outcome` options map to dagitty functions of the same name.

Examples

```
edges <- c("A -> { B C D }",
           "B -> C",
           "E -> { B C }")

# must pass exposure and outcome arguments to dagitty::adjustmentSets()
qd_todagitty(edges, exposure = "A", outcome = "C")
qd_todagitty(edges, exposure = "A", outcome = "C", type = "minimal")
```

`sep_opts` *View options for fixed node separator characters*

Description

Preview character options for use as the fixed node separator in SWIGs.

Usage

```
sep_opts(table = FALSE)
```

Arguments

`table`	Logical to show or hide HTML table display of available characters. Defaults to `FALSE`.

Index

get_conditioned_nodes *(qd_themes)*, 7

makeDAG, 2

qd_dag, 3
qd_embed, 5
qd_save, 5
qd_swig, 6
qd_themes, 7
qd_todagitty, 7

sep_opts, 9

theme_base *(qd_themes)*, 7
theme_circles *(qd_themes)*, 7
theme_dots *(qd_themes)*, 7