Package 'havel'

May 8, 2026

Type Package

Title Visualize and Tabulate 'R' Package Dependencies

Version 0.1.2

Description Plot an 'R' package's recursive dependency graph and tabulate the number of unique downstream dependencies added by top-level dependencies. This helps 'R' package developers identify which of their declared dependencies add the most downstream dependencies in order to prioritize them for removal if needed. Uses graph stress minimization adapted from Schoch (2023) <doi:10.21105/joss.05238> and originally reported in Gansner et al. (2004) <doi:10.1007/978-3-540-31843-9_25>.

License GPL (>= 3)

Encoding UTF-8

LinkingTo Rcpp

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports cli, collapse, cppRouting, data.table, graphics, grDevices, grid, methods, pak, Rcpp, rlang, stats, tools, utils

Suggests ggplot2, ggraph, igraph, pals, tinytest

LazyData true

URL https://github.com/andrewGhazi/havel

BugReports https://github.com/andrewGhazi/havel/issues

NeedsCompilation yes

Author Andrew Ghazi [aut, cre, cph], David Schoch [cph] (Original author of 'graphlayouts' code adapted in src/stress.cpp)

Maintainer Andrew Ghazi <andrew_ghazi@hms.harvard.edu>

Repository CRAN

Date/Publication 2025-12-18 14:10:02 UTC

Contents

`havel-package` *A short title line describing what the package does*

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

`pkg_deps_ex` *Example results from* `pkg_deps`

Description

This is a named list containing pre-computed examples of package dependency information for packages `data.table` (v1.17.8), `ggplot2` (v4.0.1), and `utils` pulled on Monday November 15th 2025. The first two use `pkg_deps` from the pak package and the last one uses an internal function in `havel` that falls back to `package_dependencies` from `tools` (`pkg_deps` fails on base packages). These precomputed results are used to provide data to run the examples without requiring internet access (notably on CRAN's servers).

Usage

```
pkg_deps_ex
```

Format

A named list containing three data frames of package dependency information:

data.table a tibble describing `data.table`'s dependencies

ggplot2 a tibble describing `ggplot2`'s dependencies

utils a data.table describing `utils`'s dependencies

`plot_deps_graph` *Plot dependency graph*

Description

Plot a package's dependency graph, coloring each node by the number of packages it depends on.

Usage

```
plot_deps_graph(
  pkg,
  dep_type = c("depends", "imports", "linkingto"),
  pak_res = NULL,
  info_method = "pak",
  n_iter = 100,
  init = "mds",
  gg = FALSE,
  lwd = 1,
  cex = 1,
  pad_h = 0.09,
  pad_w = 0.07,
  arw = 1,
  legend_loc = "topright",
  ...
)
```

Arguments

`pkg`	package string passed to `pak::pkg_deps`
`dep_type`	type(s) of dependencies to look up. Valid values are `c("depends", "imports", "linkingto")`
`pak_res`	a pre-computed result from `pak::pkg_deps`
`info_method`	either "pak" or "tools". The latter will use `tools::package_dependencies` to look up package info.
`n_iter`	number of iterations for stress graph layout computation
`init`	how to initialize layout, MDS (default) or randomly
`gg`	If true, use ggplot2 + ggraph to layout/draw the plot instead of base graphics. Other graphical arguments below will be ignored.

lwd	line width
cex	text size multiplication factor (see graphics::par)
pad_h	height padding
pad_w	width padding
arw	factor by which to lengthen/shorten arrowheads
legend_loc	one of "topright"/"topleft"/"bottomright"/"bottomleft" indicating where to draw the legend
...	other arguments passed to par()

Details

dep_type = "suggests" is currently not supported because Suggests can be cyclic.

Pre-computing the dependency lookup with pak::pkg_deps and passing it to the pak_res argument can be handy when fiddling with graphical parameters. Also handy to avoid hitting the bundled GitHub PAT limits used by pak::pkg_deps().

The arrows and padding will be off if you resize the graphics window with the base graphics version. Either set gg=TRUE or set your desired graphics device size, then re-run your command.

The layout in the base graphics version is initialized with a bit of random jitter. If you want to tweak it change the random seed and try again.

Random layout initialization only applies to base graphics when used, and typically looks worse than MDS initialization. Generally only useful if the MDS layout happens to overlap the legend or something.

Value

By default, no return value, called for side effect of producing a plot. If gg = TRUE, then a ggplot object.

See Also

uniq_pkg_deps

Examples

```
plot_deps_graph("ggplot2", pak_res = pkg_deps_ex$ggplot2)
# ^ ggplot2 has a moderate number of dependencies

plot_deps_graph("data.table", pak_res = pkg_deps_ex$data.table)
# ^ data.table has only one

# The `pak_res` arguments here are pre-computed results to avoid internet
# access while running the examples on CRAN's servers. They aren't required.
```

`uniq_pkg_deps` *Count unique dependencies*

Description

A function for package developers to count which direct dependencies add the most unique dependencies (direct plus downstream) in total.

Usage

```
uniq_pkg_deps(
  pkg,
  dep_type = c("depends", "imports", "linkingto"),
  pak_res = NULL,
  info_method = "pak",
  order = 1
)
```

Arguments

`pkg`	a package to check
`dep_type`	type(s) of dependencies to look up. Valid values are `c("depends", "imports", "linkingto")`
`pak_res`	a pre-computed result from `pak::pkg_deps`
`info_method`	either "pak" or "tools". The latter will use `tools::package_dependencies` to look up package info.
`order`	Consider combinations of this many packages. Be careful going beyond 3.

Details

Use this function to tabulate which packages have the most unique downstream dependencies. For a package author, these are good targets to prioritize for removal if possible.

Because of the graph structure of dependencies, sometimes there isn't any one package that adds a lot of unique dependencies, but there is a PAIR or TRIPLET do. Set the `order` argument to check which pairs, triplets, etc have the most unique dependencies.

Value

A data.table listing the packages, number of unique dependencies, and the unique dependencies themselves.

See Also

`plot_deps_graph`

Examples

```
uniq_pkg_deps("ggplot2", pak_res = pkg_deps_ex$ggplot2)
# ^ scales adds the most unique dependencies to ggplot2 -- 6 including itself.
# The `pak_res` argument here is a pre-computed result to avoid internet
# access while running the examples on CRAN's servers. It's not required.
```

Index