

Package: pkggraph (via r-universe)

September 16, 2026

Type Package

Title Explore the R Package Dependencies on the Comprehensive R
Archive Network (CRAN) Like Repositories

Version 0.3.1

Description Explore various dependencies of a packages (on the
Comprehensive R Archive Network Like repositories). The
functions `get_neighborhood()` and `get_dependencies()` provide
dependencies of packages and `as_graph()` can be used to convert
into a 'igraph' object for further analysis and plotting.

Imports dplyr (>= 1.0.1), tidyr (>= 1.3.0), igraph (>= 1.0.1), ggplot2
(>= 4.0.0), tidygraph (>= 1.3.1), ggraph (>= 2.2.2), tibble (>=
1.3.0), rlang (>= 1.1.0), cli (>= 3.6.0), checkmate (>= 2.3.0),
stringr (>= 1.5.0), BiocManager (>= 1.30.27), tools, utils,

Depends R (>= 4.1.0),

Suggests rmarkdown

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL <https://github.com/talegari/pkggraph>

BugReports <https://github.com/talegari/pkggraph/issues>

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libglpk-dev libicu-dev libxml2-dev

Repository <https://talegari.r-universe.dev>

Date/Publication 2026-04-19 09:08:14 UTC

RemoteUrl <https://github.com/talegari/pkggraph>

RemoteRef HEAD

RemoteSha d93a0f0cf4ea5bb8d8b226d8a8d0e195b216c98b

Contents

as_graph	2
get_dependencies	3
get_neighborhood	4
init	5
plot.neighborhood_graph	6
Index	7

as_graph	<i>Get igraph object from a dataframe representing package dependencies</i>
----------	---

Description

Packages dependencies (typically obtained from `get_dependencies()` or `get_neighborhood()`) is converted into a **igraph** object.

Usage

```
as_graph(dependency_df)
```

Arguments

dependency_df (dataframe) Representing package dependencies

Value

igraph object

Examples

```
## Not run:
init()
as_graph(get_neighborhood("mlr3"))

## End(Not run)
```

get_dependencies	<i>Get (reverse) dependencies of a set of packages</i>
------------------	--

Description

Get (reverse) dependencies of a set of packages till a certain depth(level) for a set of dependency types (relation).

Usage

```
get_dependencies(  
  packages,  
  level = 1L,  
  relation = c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"),  
  strict = FALSE,  
  ignore = c("datasets", "utils", "grDevices", "graphics", "stats", "methods"),  
  reverse = FALSE  
)
```

Arguments

packages	(chr) Package names
level	(count(1)) Depth of recursive dependency
relation	(chr) Types of relations. Must be a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")
strict	(flag[T]) Whether to consider all packages (alternately only 'relation' specific packages) when computing dependencies for the next level.
ignore	(chr) package names to ignore.
reverse	(flag[F]) Whether to get reverse dependencies.

Value

(dataframe) with three columns: pkg_1, relation and pkg_2

See Also

[get_neighborhood](#)

Examples

```
## Not run:  
init()  
get_dependencies("mlr3") |>  
get_dependencies("mlr3", level = 2)  
get_dependencies("mlr3", level = 2, reverse = TRUE)  
get_dependencies("mlr3", level = 2, relation = "Imports")  
# setting strict to TRUE to only consider 'Imports' of the previous level
```

```

get_dependencies("mlr3",
                 level = 2,
                 relation = "Imports",
                 strict = TRUE
                 )

## End(Not run)

```

get_neighborhood	<i>Obtain dependencies and reverse dependencies of a set of packages</i>
------------------	--

Description

Obtain dependencies and reverse dependencies of packages till a given depth (level) for a set of dependency types (relation).

Usage

```

get_neighborhood(
  packages,
  level = 1L,
  relation = c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"),
  strict = FALSE,
  ignore = c("datasets", "utils", "grDevices", "graphics", "stats", "methods")
)

```

Arguments

packages	(chr) Package names
level	(count[1]) Depth of recursive dependency
relation	(chr) Types of relations. Must be a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")
strict	(flag[T]) Whether to consider all packages (alternately only 'relation' specific packages) when computing dependencies for the next level
ignore	(chr) Package names to ignore.

Value

(dataframe) three columns: pkg_1, relation and pkg_2

See Also

[get_dependencies](#)

Examples

```
## Not run:
init()
# explore first level dependencies
get_neighborhood("mlr3")

# explore second level dependencies
get_neighborhood("caret", level = 2)

# explore first level dependencies of multiple packages
# and consider second level dependencies
get_neighborhood(c("caret", "mlr"))

# get 'imports' specific neighborhood of 'mlr'
get_neighborhood("mlr", relation = "Imports")

## End(Not run)
```

init

Create package metadata and dependency dataframes

Description

Initiate the package by loading the data into parent frame. This should be done as soon as the package is loaded or attached. This creates(rewrites) new variables `deftable` and `packmeta` to the environment where it is run from.

Usage

```
init(repository = "CRAN", ...)
```

Arguments

<code>repository</code>	(chr[1], Default: "CRAN") One among <code>c("CRAN", "BioCsoft", "BioCann", "BioCexp", "BioCextra", "omegahat")</code> . To use a repository not in this list, set 'repository' to NULL and pass named argument called 'repos' with a valid repository address. This will be passed as is to <code>utils::available.packages()</code> .
<code>...</code>	Parameters to be passed to <code>utils::available.packages()</code>

Details

Format of `packmeta`: A dataframe with one row per package with some metadata.

Format of `deftable`: Dataframe with three columns `pkg_1` (chr), `relation` (factor, levels = Depends, Imports, Suggests, LinkingTo, Enhances), `pkg_2` (chr)

Value

TRUE (invisibly)

`plot.neighborhood_graph`*Static plot of package dependencies*

Description

Plot of packages dependencies or neighborhood with edges colored by relation and node sized by 'centrality' from a neighborhood_graph obtained from [as_graph](#).

Usage

```
## S3 method for class 'neighborhood_graph'
plot(x, layout = "sugiyama", ...)
```

Arguments

x	neighborhood_graph
layout	layout is passed to ggraph::ggraph
...	Not used

Value

ggraph plot

Examples

```
## Not run:
init()
get_neighborhood("mboost") |>
  as_graph() |>
  plot()

## End(Not run)
```

Index

`as_graph`, [2](#), [6](#)

`get_dependencies`, [3](#), [4](#)

`get_dependencies()`, [2](#)

`get_neighborhood`, [3](#), [4](#)

`get_neighborhood()`, [2](#)

`ggraph::ggraph`, [6](#)

`init`, [5](#)

`plot.neighborhood_graph`, [6](#)

`utils::available.packages()`, [5](#)