

# Package ‘pkggraph’

May 9, 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>

**NeedsCompilation** no

**Author** Komala Sheshachala Srikanth [aut, cre],  
Singh Nikhil [aut]

**Maintainer** Komala Sheshachala Srikanth <sri.teach@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-04-22 12:00:09 UTC

## Contents

<code>as_graph</code> . . . . .	2
<code>get_dependencies</code> . . . . .	2

get_neighborhood . . . . .	4
init . . . . .	5
plot.neighborhood_graph . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

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      *Obtain dependencies and reverse dependencies of a set of packages*

---

### 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 `deptable` and `packmeta` to the environment where it is run from.

## Usage

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

## Arguments

`repository` (chr[1], Default: "CRAN") One among `c("CRAN", "BioCsoft", "BioCann", "BioCexp", "BioCextra", "omegahat")`. 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 `utils::available.packages()`.

`...` Parameters to be passed to `utils::available.packages()`

## Details

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

Format of `deptable`: 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

<code>x</code>	neighborhood_graph
<code>layout</code>	layout is passed to <a href="#">ggraph::ggraph</a>
<code>...</code>	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`, [2](#), [4](#)

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

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

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

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

`init`, [5](#)

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

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