

# Package ‘cranly’

February 14, 2019

**Title** Package Directives and Collaboration Networks in CRAN

**Version** 0.3

**Description**

Provides core visualisations and summaries for the CRAN package database. The package provides comprehensive methods for cleaning up and organising the information in the CRAN package database, for building package directives networks (depends, imports, suggests, enhances, linking to) and collaboration networks, producing package dependence trees, and for computing useful summaries and producing interactive visualisations from the resulting networks. The package also provides functions to coerce the networks to 'igraph' <<https://CRAN.R-project.org/package=igraph>> objects for further analyses and modelling.

**URL** <https://github.com/ikosmidis/cranly>

**BugReports** <https://github.com/ikosmidis/cranly/issues>

**Depends** R (>= 3.4.0)

**Imports** visNetwork, colorspace, igraph, magrittr, stringr, ggplot2, countrycode

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Ioannis Kosmidis [aut, cre] (<<https://orcid.org/0000-0003-1556-0302>>)

**Maintainer** Ioannis Kosmidis <[ioannis.kosmidis@warwick.ac.uk](mailto:ioannis.kosmidis@warwick.ac.uk)>

**Repository** CRAN

**Date/Publication** 2019-02-14 21:42:14 UTC

**R topics documented:**

as.igraph.cranly_network . . . . .	2
build_dependence_tree . . . . .	3
build_dependence_tree.cranly_network . . . . .	4
build_network . . . . .	5
build_network.cranly_db . . . . .	5
clean_CRAN_db . . . . .	6
clean_up_author . . . . .	8
clean_up_directives . . . . .	8
compute_dependence_tree . . . . .	9
cranly . . . . .	9
package_by . . . . .	10
plot.cranly_dependence_tree . . . . .	12
plot.cranly_network . . . . .	13
plot.summary_cranly_network . . . . .	14
subset.cranly_network . . . . .	15
summary.cranly_dependence_tree . . . . .	16
summary.cranly_network . . . . .	17
<b>Index</b>	<b>20</b>

---

as.igraph.cranly\_network

*Coerce a [cranly\\_network](#) to an [graph](#) object*

---

**Description**

Coerce a [cranly\\_network](#) to an [graph](#) object

**Usage**

```
## S3 method for class 'cranly_network'
as.igraph(x, reverse = FALSE, ...)
```

**Arguments**

x	a <a href="#">cranly_network</a> object
reverse	logical. Should the direction of the edges be reversed? See details. Default is TRUE
...	currently not used

## Details

The convention for a `cranly_network` object with `perspective = "package"` is that the direction of an edge is from the package that is imported by, suggested by, enhances or is a dependency of another package, to the latter package. `reverse` reverses that direction to correctly compute relevant network summaries (see `summary.cranly_network`). `reverse` is only relevant when the `attr(x, "perspective")` is "package" and is ignored when `attr(x, "perspective")` is "author", in which case the resulting `graph` object represents an undirected network of authors.

## Examples

```
## Not run:

#' cran_db <- clean_CRAN_db()
## Package directives network
package_network <- build_network(object = cran_db, perspective = "package")
igraph::as.igraph(package_network)

## Author collaboration network
author_network <- build_network(object = cran_db, perspective = "author")
igraph::as.igraph(author_network)

## End(Not run)
```

---

`build_dependence_tree` `build_dependence_tree` *method for an object*

---

## Description

`build_dependence_tree` method for an object

## Usage

```
build_dependence_tree(x, ...)
```

## Arguments

<code>x</code>	an object to use for building a dependence tree
<code>...</code>	other arguments to be passed to the method

## See Also

`build_network.cranly_network` `compute_dependence_tree`

---

build\_dependence\_tree.cranly\_network

*Construct a [cranly\\_dependence\\_tree](#) object*

---

## Description

Construct a [cranly\\_dependence\\_tree](#) object

## Usage

```
## S3 method for class 'cranly_network'  
build_dependence_tree(x, package = Inf,  
  base = FALSE, recommended = TRUE, global = TRUE, ...)
```

## Arguments

x	a <a href="#">cranly_network</a> object
package	a vector of character strings with the package names to be matched. Default is Inf which returns all available packages in x for further subsetting
base	logical. Should we include base packages in the subset? Default is TRUE
recommended	logical. Should we include recommended packages in the subset? Default is TRUE
global	logical. If TRUE (default) the network summary statistics are computed on object, otherwise, on the subset of object according to package, author, directive, base, recommended
...	currently not used

## See Also

[compute\\_dependence\\_tree](#)

## Examples

```
## Not run:  
cran_db <- clean_CRAN_db()  
package_network <- build_network(object = cran_db)  
dep_tree <- build_dependence_tree(package_network, package = "PlackettLuce")  
plot(dep_tree)  
  
## End(Not run)
```

---

build_network	build_network <i>method for an object</i>
---------------	---

---

**Description**

build\_network method for an object

**Usage**

```
build_network(object, ...)
```

**Arguments**

object	an object to use for building a network
...	other arguments to be passed to the method

**See Also**

build\_network.cranly\_network

---

build_network.cranly_db	<i>Compute edges and nodes of package directives and collaboration networks</i>
-------------------------	---

---

**Description**

Compute edges and nodes of package directives and collaboration networks

**Usage**

```
## S3 method for class 'cranly_db'
build_network(object = clean_CRAN_db(),
  trace = FALSE, perspective = "package", ...)
```

**Arguments**

object	a <a href="#">cranly_db</a> object
trace	logical. Print progress information? Default is FALSE
perspective	character. Build a "package" (default) or an "author" network?
...	Currently not used

## Details

The convention for a `cranly_network` object with `perspective = "package"` is that the direction of an edge is from the package that is imported by, suggested by, enhances or is a dependency of another package, to the latter package. The author collaboration network is analysed and visualised as undirected by all methods in `cranly`.

## Value

A list of 2 `data.frames` with the edges and nodes of the network

## Examples

```
## Not run:
cran_db <- clean_CRAN_db()
## Package directives network
package_network <- build_network(object = cran_db, perspective = "package")
head(package_network$edges)
head(package_network$nodes)
attr(package_network, "timestamp")
class(package_network)

## Author collaboration network
author_network <- build_network(object = cran_db, perspective = "author")
head(author_network$edges)
head(author_network$nodes)
attr(author_network, "timestamp")
class(author_network)

## End(Not run)
```

---

<code>clean_CRAN_db</code>	<i>Clean and organise package and author names in the output of <code>tools::CRAN_package_db()</code></i>
----------------------------	---

---

## Description

Clean and organise package and author names in the output of `tools::CRAN_package_db()`

## Usage

```
clean_CRAN_db/packages_db = tools::CRAN_package_db(),
  clean_directives = clean_up_directives,
  clean_author = clean_up_author)
```

## Arguments

packages_db	a <code>data.frame</code> with the same structure to the output of <code>CRAN_package_db</code> (default) or <code>available.packages</code>
clean_directives	a function that transforms the contents of the various directives in the package descriptions to vectors of package names. Default is <code>clean_up_directives</code> .
clean_author	a function that transforms the contents of Author to vectors of package authors. Default is <code>clean_up_author</code> .

## Details

`clean_CRAN_db` uses `clean_up_directives` and `clean_up_authors` to clean up the author names and package names in the various directives (like Imports, Depends, Suggests, Enhances, LinkingTo) as in the `data.frame` that results from `CRAN_package_db`) and return an organised `data.frame` of class `cranly_db` that can be used for further analysis.

The function tries hard to identify and eliminate mistakes in the Author field of the description file, and extract a clean list of only author names. The relevant operations are coded in the `clean_up_author` function. Specifically, some references to copyright holders had to go because they were contaminating the list of authors (most are not necessary anyway, but that is a different story...). The current version of `clean_up_author` is far from best practice in using regex but it currently does a fair job in cleaning up messy Author fields. It will be improving in future versions.

Custom clean-up functions can also be supplied via the `clean_directives` and `clean_author` arguments.

## Value

An `data.frame` with the same variables as `packaged_db` (but with lower case names), a `timestamp` attribute and also inheriting from `class_db`.

## Examples

```
## Not run:
## Before cleaning
cran_db <- tools::CRAN_package_db()
cran_db[cran_db$Package == "weights", "Author"]

## After clean up
package_db <- clean_CRAN_db(cran_db)
package_db[package_db$Package == "weights", "Author"]

## End(Not run)
```

---

clean_up_author	<i>Clean up author names</i>
-----------------	------------------------------

---

**Description**

Clean up author names

**Usage**

```
clean_up_author(variable)
```

**Arguments**

variable      a character string

**Value**

A list of one vector of character strings

**Examples**

```
clean_up_author(paste("The R Core team, Brian & with some assistance from Achim, Hadley;",  
                      "Kurt\n Portugal; Ireland; Italy; Greece; Spain"))
```

---

clean_up_directives	<i>Clean up package directives</i>
---------------------	------------------------------------

---

**Description**

Clean up package directives

**Usage**

```
clean_up_directives(variable)
```

**Arguments**

variable      a character string

**Value**

A list of one vector of character strings

**Examples**

```
clean_up_directives("R (234)\n stats (>0.01),      base\n graphics")
```



---

`compute_dependence_tree`*Computes the dependence tree of a package*

---

**Description**

Computes the dependence tree of a package

**Usage**

```
compute_dependence_tree(x, package = NULL, generation = 0)
```

**Arguments**

<code>x</code>	a <code>cranly_network</code> object
<code>package</code>	a vector of character strings with the package names to be matched. Default is <code>Inf</code> which returns all available packages in <code>x</code> for further subsetting
<code>generation</code>	integer. The original generation for the package.

**Details**

Implements a recursion that computes the full dependence tree of a package from `x`. Specifically, the packages that are requirements for package (Depends, Imports or LinkingTo) are found, then the requirements for those packages are found, and so on.

**See Also**

`build_dependence_tree`

---

`cranly`*cranly: CRAN package database analytics and visualizations*

---

**Description**

`cranly`: CRAN package database analytics and visualizations

**Details**

**cranly** provides core visualisations and summaries for the CRAN package database. The package provides comprehensive methods for cleaning up and organising the information in the CRAN package database, for building package directives networks (depends, imports, suggests, enhances, linking to) and collaboration networks, and for computing summaries and producing interactive visualisations from the resulting networks. Network visualisation is through the **visNetwork** (<https://CRAN.R-project.org/package=visNetwork>) package. The package also provides functions to coerce the networks to **igraph** <https://CRAN.R-project.org/package=igraph> objects for further analyses and modelling.

## Acknowledgements

- David Selby (<http://selbydavid.com>) experimented with and provided helpful comments and feedback on a pre-release version of **cranly**. His help is gratefully acknowledged.
- This work has been partially supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1 (Turing award number TU/B/000082)

---

package_by	<i>Find packages and authors by authors or packages with names matching a specific string</i>
------------	---

---

## Description

Find packages and authors by authors or packages with names matching a specific string

## Usage

```
package_by(x, author = NULL, exact = FALSE)
package_with(x, name = NULL, exact = FALSE)
author_with(x, name = NULL, exact = FALSE)
author_of(x, package = NULL, exact = FALSE)
suggests(x, package = NULL, exact = FALSE)
imports(x, package = NULL, exact = FALSE)
depends(x, package = NULL, exact = FALSE)
linking_to(x, package = NULL, exact = FALSE)
enhances(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
package_by(x, author = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
package_with(x, name = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
author_of(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
author_with(x, name = NULL, exact = FALSE)
```

```
## S3 method for class 'cranly_network'
suggests(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
imports(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
depends(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
linking_to(x, package = NULL, exact = FALSE)

## S3 method for class 'cranly_network'
enhances(x, package = NULL, exact = FALSE)
```

### Arguments

x	a <a href="#">cranly_network</a> object
author	a vector of character strings with the author names to be matched. Default is Inf which returns all available author in x for further subsetting
exact	logical. Should we use exact matching? Default is TRUE
name	a vector of character strings with the names to be matched. If Inf all available names in x are returned. If NULL (default) nothing is matched
package	a vector of character strings with the package names to be matched. Default is Inf which returns all available packages in x for further subsetting

### Examples

```
## Not run:
cran_db <- clean_CRAN_db()
## Using a package directives network
package_network <- build_network(cran_db)
## Find all packages containing glm in their name
package_with(package_network, name = "glm")
## Find all authors of packages containing brglm in their name
author_of(package_network, package = "rglm", exact = FALSE)
## Find all packages with brglm in their name
package_with(package_network, name = "rglm", exact = FALSE)
## Find all authors of the package brglm2
author_of(package_network, package = "brglm2", exact = TRUE)
## Find all authors with Ioannis in their name
author_with(package_network, name = "Ioannis", exact = TRUE)
## Find all packages that package Rcpp suggests
suggests(package_network, package = "Rcpp", exact = TRUE)
## Find all packages that package Rcpp imports
imports(package_network, package = "Rcpp", exact = TRUE)
## Find all packages that package RcppArmadillo is linking to
linking_to(package_network, package = "RcppArmadillo", exact = TRUE)

## Using an author collaboration network
```

```

author_network <- build_network(cran_db, perspective = "author")
## Find all packages containing glm in their name
package_with(author_network, name = "glm")
## Find all authors of packages containing brglm in their name
author_of(author_network, package = "rglm", exact = FALSE)
## Find all packages with brglm in their name
package_with(author_network, name = "rglm", exact = FALSE)
## Find all authors of the package brglm2
author_of(author_network, package = "brglm2", exact = TRUE)
## Find all authors with Ioannis in their name
author_with(author_network, name = "Ioannis", exact = TRUE)

## End(Not run)

```

---

plot.cranly\_dependence\_tree

*Interactive visualization of package(s) dependence tree from a [cranly\\_network](#)*

---

## Description

Interactive visualization of package(s) dependence tree from a [cranly\\_network](#)

## Usage

```

## S3 method for class 'cranly_dependence_tree'
plot(x, physics_threshold = 200,
     height = NULL, width = NULL, dragNodes = TRUE, dragView = TRUE,
     zoomView = TRUE, legend = TRUE, title = TRUE, plot = TRUE, ...)

```

## Arguments

x	a <a href="#">cranly_dependence_tree</a> object
physics_threshold	integer. How many nodes before switching off physics simulations for edges? Default is 200. See, also <a href="#">visEdges</a>
height	: Height (optional, defaults to automatic sizing)
width	: Width (optional, defaults to automatic sizing)
dragNodes	logical. Should the user be able to drag the nodes that are not fixed? Default is TRUE
dragView	logical. Should the user be able to drag the view around? Default is TRUE
zoomView	logical. Should the user be able to zoom in? Default is TRUE
legend	logical. Should a legend be added on the resulting visualization? Default is TRUE
title	logical. Should a title be added on the resulting visualization? Default is TRUE
plot	logical. Should the visualisation be returned? Default is TRUE
...	currently not used

**See Also**

compute\_dependence\_tree dependence\_tree

---

plot.cranly\_network *Interactive visualization of a package or author* [cranly\\_network](#)

---

**Description**

Interactive visualization of a package or author [cranly\\_network](#)

**Usage**

```
## S3 method for class 'cranly_network'
plot(x, package = Inf, author = Inf,
     directive = c("imports", "suggests", "enhances", "depends",
                  "linkingto"), base = TRUE, recommended = TRUE, exact = TRUE,
     global = TRUE, physics_threshold = 200, height = NULL,
     width = NULL, dragNodes = TRUE, dragView = TRUE, zoomView = TRUE,
     legend = TRUE, title = TRUE, plot = TRUE, ...)
```

**Arguments**

x	a <a href="#">cranly_network</a> object
package	a vector of character strings with the package names to be matched. Default is Inf which returns all available packages in x for further subsetting
author	a vector of character strings with the author names to be matched. Default is Inf which returns all available author in x for further subsetting
directive	a vector of at least one of "Imports", "Suggests", "Enhances", "Depends"
base	logical. Should we include base packages in the subset? Default is TRUE
recommended	logical. Should we include recommended packages in the subset? Default is TRUE
exact	logical. Should we use exact matching? Default is TRUE
global	logical. If TRUE (default) the network summary statistics are computed on object, otherwise, on the subset of object according to package, author, directive, base, recommended
physics_threshold	integer. How many nodes before switching off physics simulations for edges? Default is 200. See, also <a href="#">visEdges</a>
height	: Height (optional, defaults to automatic sizing)
width	: Width (optional, defaults to automatic sizing)
dragNodes	logical. Should the user be able to drag the nodes that are not fixed? Default is TRUE
dragView	logical. Should the user be able to drag the view around? Default is TRUE

zoomView	logical. Should the user be able to zoom in? Default is TRUE
legend	logical. Should a legend be added on the resulting visualization? Default is TRUE
title	logical. Should a title be added on the resulting visualization? Default is TRUE
plot	logical. Should the visualisation be returned? Default is TRUE
...	currently not used

### Examples

```
## Not run:
cran_db <- clean_CRAN_db()
package_network <- build_network(cran_db)
## The package directives network of all users with Ioannis in
## their name from the CRAN database subset cran_db
plot(package_network, author = "Ioannis")
## The package directives network of "Achim Zeileis"
plot(package_network, author = "Achim Zeileis")

author_network <- build_network(cran_db, perspective = "author")
plot(author_network, author = "Ioannis", title = TRUE)

## End(Not run)
```

---

```
plot.summary_cranly_network
```

*Top-n package or author barplots according to a range of network statistics*

---

### Description

Top-n package or author barplots according to a range of network statistics

### Usage

```
## S3 method for class 'summary_cranly_network'
plot(x, top = 20, according_to = NULL,
     scale = FALSE, ...)
```

### Arguments

x	a <a href="#">summary_cranly_network</a> object
top	integer. How may top packages or authors should be plotted? Default is 20
according_to	the statistic according to which the top-top list is produced. See <a href="#">summary.cranly_network</a> for available statistics
scale	logical. Should the statistics be scaled to lie between 0 and 1 before plotting? Default is FALSE
...	currently not used

**Examples**

```
## Not run:
cran_db <- clean_CRAN_db()
## package network
package_network <- build_network(cran_db)
package_summaries <- summary(package_network)
plot(package_summaries, according_to = "n_imported_by", top = 30)
plot(package_summaries, according_to = "n_depended_by", top = 30)
plot(package_summaries, according_to = "page_rank", top = 30)

## author network
author_network <- build_network(cran_db, perspective = "author")
author_summaries <- summary(author_network)
plot(author_summaries, according_to = "n_collaborators", top = 30)
plot(author_summaries, according_to = "n_packages", top = 30)
plot(author_summaries, according_to = "page_rank", top = 30)

## End(Not run)
```

---

subset.cranly\_network *Subset a [cranly\\_network](#) according to author, package and/or directive*

---

**Description**

Subset a [cranly\\_network](#) according to author, package and/or directive

**Usage**

```
## S3 method for class 'cranly_network'
subset(x, package = Inf, author = Inf,
       directive = c("imports", "suggests", "enhances", "depends",
                    "linkingto"), base = TRUE, recommended = TRUE, exact = TRUE,
       only = FALSE, ...)
```

**Arguments**

x	a <a href="#">cranly_network</a> object
package	a vector of character strings with the package names to be matched. Default is Inf which returns all available packages in x for further subsetting
author	a vector of character strings with the author names to be matched. Default is Inf which returns all available author in x for further subsetting
directive	a vector of at least one of "Imports", "Suggests", "Enhances", "Depends"
base	logical. Should we include base packages in the subset? Default is TRUE

recommended	logical. Should we include recommended packages in the subset? Default is TRUE
exact	logical. Should we use exact matching? Default is TRUE
only	logical. If TRUE the subset includes only the edges between packages named in package and/or authors named in author. If FALSE (default) edges to and from all other packages and/or authors that are linked to package and/or author are included in the subset
...	currently not used

---

summary.cranly\_dependence\_tree

*summary method for [cranly\\_dependence\\_tree](#) objects*

---

## Description

Hard-dependence summaries for R packages from a [cranly\\_dependence\\_tree](#) object

## Usage

```
## S3 method for class 'cranly_dependence_tree'
summary(object, ...)
```

## Arguments

object	a <a href="#">cranly_dependence_tree</a> object
...	currently not used

## Details

The summary method for a [cranly\\_dependence\\_tree](#) object returns the number of generations the R package(s) in the object inherit from (`n_generations`), the immediate parents of the R package(s) (`parents`), and a dependence index `dependence_index` defined as

$$\frac{\sum_{i \in C_p; i \neq p} \frac{1}{N_i} g_i}{\sum_{i \in C_p; i \neq p} \frac{1}{N_i}}$$

where  $C_p$  is the dependence tree for the package(s)  $p$ ,  $N_i$  is the total number of packages that depend, link or import package  $i$ , and  $g_i$  is the generation that package  $i$  appears in the dependence tree of package(s)  $p$ . The generation takes values on the non-positive integers, with the package(s)  $p$  being placed at generation 0, the packages that  $p$  links to, depends or imports at generation -1 and so on.

A dependence index of zero means that the  $p$  only has immediate parents. The dependence index weights the dependencies based on how popular these are, in the sense that the index is not penalised if the package depends on popular packages. The greatest the dependence index is the more baggage the package carries, and the maintainers may want to remove any dependencies that are not necessary.



**Value**

A list with components `n_generations`, `parents`, and `dependence_index`.

**See Also**

`build_dependence_tree` `compute_dependence_tree`

**Examples**

```
## Not run:
cran_db <- clean_CRAN_db()
package_network <- build_network(object = cran_db)

## Two light packages
dep_tree <- build_dependence_tree(package_network, package = "brglm")
summary(dep_tree)

dep_tree <- build_dependence_tree(package_network, package = "gnm")
summary(dep_tree)

## A somewhat heavier package (sorry)...
dep_tree <- build_dependence_tree(package_network, package = "cranly")
summary(dep_tree)

## End(Not run)
```

---

summary.cranly\_network

*Compute a range of package directives and collaboration network statistics*

---

**Description**

Compute a range of package directives and collaboration network statistics

**Usage**

```
## S3 method for class 'cranly_network'
summary(object, advanced = TRUE, ...)
```

**Arguments**

<code>object</code>	a <code>cranly_network</code> object
<code>advanced</code>	logical. If FALSE (default) only basic network statistics are computed; if TRUE advanced statistics are also included in the computation (see Details).
<code>...</code>	currently not used

## Details

If `attr(object, "perspective")` is "package" then the resulting `data.frame` will have the following variables:

- `package`. package name
- `n_authors` (basic). number of authors for the package
- `n_imports` (basic). number of packages the package imports
- `n_imported_by` (basic). number of times the package is imported by other packages
- `n_suggests` (basic). number of packages the package suggests
- `n_suggested_by` (basic). number of times the package is suggested by other packages
- `n_depends` (basic). number of packages the package depends on
- `n_depended_by` (basic). number of packages that have the package as a dependency
- `n_enhances` (basic). number of packages the package enhances
- `n_enhanced_by` (basic). number of packages the package is enhanced by
- `n_linking_to` (basic). number of packages the package links to
- `n_linked_by` (basic). number of packages the package is linked by
- `betweenness` (advanced). the package betweenness in the package network; as computed by [betweenness](#)
- `closeness` (advanced). the closeness centrality of the package in the package network; as computed by [closeness](#)
- `page_rank` (advanced). the Google PageRank of the package in the package network; as computed by [page\\_rank](#)
- `degree` (advanced). the degree of the package in the package network; as computed by [degree](#)
- `eigen_centrality` (advanced). the eigenvector centrality score of the package in the package network; as computed by [eigen\\_centrality](#)

If `attr(object, "perspective")` is "author" then the resulting `data.frame` will have the following variables:

- `author`. author name
- `n_packages` (basic). number of packages the author appears in the package authors
- `n_collaborators` (basic). total number of co-authors the author has in CRAN
- `betweenness` (advanced). the author betweenness in the author network; as computed by [betweenness](#)
- `closeness` (advanced). the closeness centrality of the author in the author network; as computed by [closeness](#)
- `page_rank` (advanced). the Google PageRank of the author in the author network; as computed by [page\\_rank](#)
- `degree` (advanced). the degree of the author in the author network; as computed by [degree](#); same as `n_collaborators`
- `eigen_centrality` (advanced). the eigenvector centrality score of the author in the author network; as computed by [eigen\\_centrality](#)

**Value**

A [data.frame](#) of various statistics for the author collaboration network or the package directives network, depending on whether `attr(object, "perspective")` is "author" or "package", respectively. See Details for the current list of statistics returned.

# Index

as.igraph.cranly\_network, 2  
author\_of (package\_by), 10  
author\_with (package\_by), 10  
available.packages, 7

betweenness, 18  
build\_dependence\_tree, 3  
build\_dependence\_tree.cranly\_network,  
4  
build\_network, 5  
build\_network.cranly\_db, 5

clean\_CRAN\_db, 6  
clean\_up\_author, 7, 8  
clean\_up\_directives, 7, 8  
closeness, 18  
compute\_dependence\_tree, 9  
CRAN\_package\_db, 7  
cranly, 9  
cranly-package (cranly), 9  
cranly\_db, 5  
cranly\_db (clean\_CRAN\_db), 6  
cranly\_dependence\_tree, 4, 12, 16  
cranly\_dependence\_tree  
(build\_dependence\_tree.cranly\_network),  
4  
cranly\_network, 2–4, 6, 9, 11–13, 15, 17  
cranly\_network  
(build\_network.cranly\_db), 5

data.frame, 6, 7, 19  
degree, 18  
depends (package\_by), 10

eigen\_centrality, 18  
enhances (package\_by), 10

graph, 2, 3

imports (package\_by), 10

linking\_to (package\_by), 10

package\_by, 10  
package\_with (package\_by), 10  
page\_rank, 18  
plot.cranly\_dependence\_tree, 12  
plot.cranly\_network, 13  
plot.summary\_cranly\_network, 14

subset.cranly\_network, 15  
suggests (package\_by), 10  
summary.cranly\_dependence\_tree, 16  
summary.cranly\_network, 14, 17  
summary\_cranly\_network, 14  
summary\_cranly\_network  
(summary.cranly\_network), 17

visEdges, 12, 13