

Package ‘cranly’

May 8, 2026

Title Package Directives and Collaboration Networks in CRAN

Version 0.6.0

Description Core visualizations and summaries for the CRAN package database. The package provides comprehensive methods for cleaning up and organizing 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 visualizations from the resulting networks and summaries. The resulting networks can be coerced 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 (>= 4.1.0)

Imports visNetwork, colorspace, igraph, stringr, ggplot2, countrycode, wordcloud, tm

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

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

Maintainer Ioannis Kosmidis <ioannis.kosmidis@warwick.ac.uk>

Repository CRAN

Date/Publication 2022-08-26 22:14:41 UTC

Contents

as.igraph.cranly_network	2
build_dependence_tree	3
build_dependence_tree.cranly_network	4
build_network.cranly_db	5
clean_CRAN_db	6
clean_up_author	7
clean_up_directives	8
compute_dependence_tree	8
compute_term_frequency	9
cranly	10
extractor-functions	11
plot.cranly_dependence_tree	14
plot.cranly_network	16
plot.summary_cranly_network	17
standardize_whitespace	18
subset.cranly_network	19
summary.cranly_dependence_tree	20
summary.cranly_network	21
word_cloud.cranly_network	23
Index	26

as.igraph.cranly_network

Coerce a `cranly_network` to an `igraph::graph` object

Description

Coerce a `cranly_network` to an `igraph::graph` object

Usage

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

Arguments

x	a <code>cranly_network</code> object.
reverse	logical. Should the direction of the edges be reversed? See details. Default is <code>TRUE</code> .
...	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 `igraph::igraph` object represents an undirected network of authors.

Examples

```
## Package directives network
cran_db <- clean_CRAN_db()

package_network <- build_network(cran_db)
igraph::as.igraph(package_network)

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

`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

<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>base</code>	logical. Should we include base packages in the subset? Default is <code>TRUE</code> .
<code>recommended</code>	logical. Should we include recommended packages in the subset? Default is <code>TRUE</code> .
<code>global</code>	logical. If <code>TRUE</code> (default) the network summary statistics are computed on object, otherwise, on the subset of object according to <code>package</code> , <code>author</code> , <code>directive</code> , <code>base</code> , <code>recommended</code> .
<code>...</code>	currently not used.

See Also

`compute_dependence_tree()` `plot.cranly_dependence_tree()` `summary.cranly_dependence_tree()`

Examples

```
cran_db <- clean_CRAN_db()  
package_network <- build_network(cran_db)  
dep_tree <- build_dependence_tree(package_network, package = "PlackettLuca")  
plot(dep_tree)
```

`build_network.cranly_db`*Compute edges and nodes of package directives and collaboration networks*

Description

Compute edges and nodes of package directives and collaboration networks

Usage

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

Arguments

<code>object</code>	a <code>cranly_db</code> object. If missing (default) a call to <code>clean_CRAN_db()</code> is issued.
<code>trace</code>	logical. Print progress information? Default is <code>FALSE</code> .
<code>perspective</code>	character. Should a "package" (default) or an "author" network be built?
<code>...</code>	Other arguments passed in <code>clean_CRAN_db()</code> when object is NULL.

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 analyzed and visualized as undirected by all methods in `cranly`.

Value

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

See Also

[clean_CRAN_db\(\)](#) [subset.cranly_network\(\)](#) [plot.cranly_network\(\)](#) [extractor-functions](#)

Examples

```
cran_db <- clean_CRAN_db()  
  
## Build 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)  
  
## Build 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)
```

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

Description

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

Usage

```
clean_CRAN_db(
  packages_db,
  clean_directives = clean_up_directives,
  clean_author = clean_up_author,
  clean_maintainer = standardize_whitespace
)
```

Arguments

`packages_db` a `data.frame` with the same structure to the output of `tools::CRAN_package_db()` (default) or `utils::available.packages()`.

`clean_directives` a function that transforms the contents of the various directives in the package descriptions to vectors of package names. Default is `clean_up_directives()`.

`clean_author` a function that transforms the contents of Author to vectors of package authors. Default is `clean_up_author()`.

`clean_maintainer` a function that transforms the contents of Maintainer to vectors of of maintainer names. Default is `standardize_whitespace()`.

Details

`clean_CRAN_db()` uses `clean_up_directives()` and `clean_up_author()` 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 `tools::CRAN_package_db()` return an organized `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

A `data.frame` with the same variables as `package_db` (but with lower case names), that also inherits from `class_db`, and has a `timestamp` attribute.

Examples

```
## Download today's CRAN package database
cran_db <- tools::CRAN_package_db()

## Before clean up
cran_db[cran_db$Package == "weights", "Author"]

## After clean up
package_db <- clean_CRAN_db(cran_db)
package_db[package_db$package == "weights", "author"]
```

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` *Clean up package directives*

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

`x` a [cranly_network](#) object.
`package` a vector of character strings with the package names to be matched. If `Inf` all available packages in `x` are returned. If `NULL` (default) nothing is matched.
`generation` 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_network\(\)](#)

compute_term_frequency

Compute term frequencies from a vector of text

Description

Compute term frequencies from a vector of text

Usage

```
compute_term_frequency(
  txt,
  ignore_words = c("www.jstor.org", "www.arxiv.org", "arxiv.org", "provides", "https"),
  stem = FALSE,
  remove_punctuation = TRUE,
  remove_stopwords = TRUE,
  remove_numbers = TRUE,
  to_lower = TRUE,
  frequency = "term"
)
```

Arguments

txt	a vector of character strings.
ignore_words	a vector of words to be ignored when forming the corpus.
stem	should words be stemmed using Porter's stemming algorithm? Default is FALSE. See tm::stemDocument() .
remove_punctuation	should punctuation be removed when forming the corpus? Default is TRUE. See tm::removePunctuation() .
remove_stopwords	should english stopwords be removed when forming the corpus? Default is TRUE. See tm::removeWords and tm::stopwords .
remove_numbers	should numbers be removed when forming the corpus? Default is TRUE. See tm::removeNumbers .
to_lower	should all terms be coerced to lower-case when forming the corpus? Default is TRUE.
frequency	the type of term frequencies to return. Options are "term" (default; a named vector of term frequencies), "document-term" (a document-term frequency matrix; see tm::TermDocumentMatrix()), "term-document" (a term-document frequency matrix; see tm::DocumentTermMatrix()).

The operations are taking place as follows: remove special characters, convert to lower-case (depending on the values of `to_lower`), remove numbers (depending on the value of `remove_numbers`), remove stop words (depending on the value of `remove_stopwords`), remove custom words (depending on the value of `ignore_words`), remove punctuation (depending on the value of `remove_punctuation`), clean up any leading or trailing whitespace, and, finally stem words (depending on the value of `stem`).

Details

If `txt` is a named vector then the names are used as document id's when forming the corpus.

Value

Either a named numeric vector (`frequency = "term"`), or an object of class `tm::DocumentTermMatrix` (`frequency = "document-term"`), or an object of class `tm::TermDocumentMatrix` (`frequency = "term-document"`).

See Also

[word_cloud\(\)](#)

cranly

cranly: CRAN package database analytics and visualizations

Description

cranly provides core visualizations and summaries for the CRAN package database. The package provides comprehensive methods for cleaning up and organizing 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 visualizations from the resulting networks. Network visualization 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.

Details

Acknowledgements:

- David Selby (<https://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).

extractor-functions *Find packages, authors, maintainers, license, versions etc by authors, packages or names matching a specific string*

Description

Find packages, authors, maintainers, license, versions etc by authors, packages or names matching a specific string

Usage

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

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

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

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

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

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

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

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

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

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

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

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

```
## S3 method for class 'cranly_network'  
enhanced_by(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
enhancing(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
maintainer_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
maintained_by(x, author = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
email_of(x, author = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
email_with(x, name = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
description_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
title_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
license_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
version_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
## S3 method for class 'cranly_network'  
release_date_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
packages_by(x, author, exact, flat)  
  
packages_with(x, name = NULL, exact = FALSE, flat = TRUE)  
  
authors_with(x, name = NULL, exact = FALSE, flat = TRUE)  
  
authors_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
emails_of(x, author = NULL, exact = FALSE, flat = TRUE)  
  
emails_with(x, name = NULL, exact = FALSE, flat = TRUE)  
  
descriptions_of(x, package = NULL, exact = FALSE, flat = TRUE)  
  
titles_of(x, package = NULL, exact = FALSE, flat = TRUE)
```

```
licenses_of(x, package = NULL, exact = FALSE, flat = TRUE)
```

```
release_dates_of(x, package = NULL, exact = FALSE, flat = TRUE)
```

```
versions_of(x, package = NULL, exact = FALSE, flat = TRUE)
```

Arguments

x	a cranly_network object.
author	a vector of character strings with the author names to be matched. If Inf all available authors in x are returned. If NULL (default) nothing is matched.
exact	logical. Should we use exact matching? Default is TRUE.
flat	if TRUE (default) then the result is an unnamed character vector. See Value for more details of what is returned.
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. If Inf all available packages in x are returned. If NULL (default) nothing is matched.

Details

The extractor functions all try to figure out what y is in the statement y is (the) extractor-function a package/author.

For example, for

- "y is the package by "Kurt Hornik"" we do `package_by(x, "Kurt Hornik")`
- "y is the author of a package with a name matching "MASS"" we do `author_of(x, "MASS")`
- "y is the package enhanced by the "prediction" package we do `enhanced_by(x, "prediction", exact = TRUE)`
- "y is the package linking to "Rcpp" we do `linking_to(x, "Rcpp", exact = TRUE)`

Value

If flat = TRUE then the result of the extraction function is a `data.frame`, which is the subset of `x$nodes` matching author, name or package (according to the value of exact). If flat = FALSE then the results is a vector.

When flat = TRUE any [NAs](#) are removed before the result is returned.

See Also

[build_network.cranly_db\(\)](#) [subset.cranly_network\(\)](#) [plot.cranly_network\(\)](#)

Examples

```

# Using a package directives network
cran_db <- clean_CRAN_db()
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 = FALSE)
## Find all packages suggested by Rcpp
suggested_by(package_network, package = "Rcpp", exact = TRUE)
## Find all packages imported by Rcpp
imported_by(package_network, package = "Rcpp", exact = TRUE)
## Find all packages enhancing brglm
enhancing(package_network, package = "brglm", exact = TRUE)
## Find all packages linking to RcppArmadillo
linking_to(package_network, package = "RcppArmadillo", exact = TRUE)
## Find the release date of RcppArmadillo
release_date_of(package_network, package = "RcppArmadillo", exact = TRUE)
## Find the release data of all packages with "brglm" in their name
release_date_of(package_network, package = "brglm", exact = FALSE)
## More information about packages with "brglm" in their name
release_date_of(package_network, package = "brglm", exact = FALSE,
                flat = FALSE)[c("package", "version")]

## 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 = FALSE)

```

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 cranly_dependence_tree object.
physics_threshold	integer. How many nodes before switching off physics simulations for edges? Default is 200. See, also visNetwork::visEdges() .
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 visualization be returned? Default is TRUE .
...	currently not used.

See Also

[compute_dependence_tree\(\)](#) [build_dependence_tree.cranly_network\(\)](#)

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", "linking_to"),
  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 cranly_network 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 <code>visNetwork::visEdges()</code> .
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 <code>TRUE</code> .
dragView	logical. Should the user be able to drag the view around? Default is <code>TRUE</code> .
zoomView	logical. Should the user be able to zoom in? Default is <code>TRUE</code> .
legend	logical. Should a legend be added on the resulting visualization? Default is <code>TRUE</code> .
title	logical. Should a title be added on the resulting visualization? Default is <code>TRUE</code> .
plot	logical. Should the visualization be returned? Default is <code>TRUE</code> .
...	currently not used.

Examples

```
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 crandb
plot(package_network, author = "Ioannis", exact = FALSE)
## 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", exact = FALSE, title = TRUE)
```

```
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 summary_cranly_network 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 summary.cranly_network for available statistics.
scale	logical. Should the statistics be scaled to lie between 0 and 1 before plotting? Default is <code>FALSE</code> .
...	currently not used

Examples

```
cran_db <- clean_CRAN_db()
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)
```

standardize_whitespace

Standardize whitespace in strings

Description

Standardize whitespace in strings

Usage

```
standardize_whitespace(variable)
```

Arguments

variable a character string.

Value

A list of one vector of character strings.

Examples

```
standardize_whitespace(" My spacebar key is broken. ")
```

```
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,
  maintainer = Inf,
  directive = c("imports", "suggests", "enhances", "depends", "linking_to"),
  base = TRUE,
  recommended = TRUE,
  exact = TRUE,
  only = FALSE,
  ...
)
```

Arguments

x	a cranly_network 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.
maintainer	a vector of character strings with the maintainer names to be matched. Default is Inf which returns all available maintainers 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.

Value

A `cranly_network` object that is the subject of `x`.

```
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

<code>object</code>	a <code>cranly_dependence_tree</code> object.
<code>...</code>	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 penalized 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.cranly_network\(\)](#) [compute_dependence_tree\(\)](#)

Examples

```
cran_db <- clean_CRAN_db()
package_network <- build_network(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)
```

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

object	a cranly_network object.
advanced	logical. If FALSE (default) only basic network statistics are computed; if TRUE advanced statistics are also included in the computation (see Details).
...	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 `igraph::betweenness()`
- `closeness` (advanced). the closeness centrality of the package in the package network; as computed by `igraph::closeness()`
- `page_rank` (advanced). the Google PageRank of the package in the package network; as computed by `igraph::page_rank()`
- `degree` (advanced). the degree of the package in the package network; as computed by `igraph::degree()`
- `eigen_centrality` (advanced). the eigenvector centrality score of the package in the package network; as computed by `igraph::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 `igraph::betweenness()`
- `closeness` (advanced). the closeness centrality of the author in the author network; as computed by `igraph::closeness()`
- `page_rank` (advanced). the Google PageRank of the author in the author network; as computed by `igraph::page_rank()`
- `degree` (advanced). the degree of the author in the author network; as computed by `igraph::degree()`; same as `n_collaborators`
- `eigen_centrality` (advanced). the eigenvector centrality score of the author in the author network; as computed by `igraph::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.

```
word_cloud.cranly_network
```

wordcloud of author names, package descriptions, and package titles

Description

wordcloud of author names, package descriptions, and package titles

Usage

```
## S3 method for class 'cranly_network'
word_cloud(
  x,
  package = Inf,
  author = Inf,
  maintainer = Inf,
  base = TRUE,
  recommended = TRUE,
  exact = TRUE,
  perspective = "description",
  random_order = FALSE,
  ignore_words = c("www.jstor.org", "www.arxiv.org", "arxiv.org", "provides", "https"),
  stem = FALSE,
  colors = rev(colorspace::heat_hcl(10)),
  ...
)

## S3 method for class 'numeric'
word_cloud(
  x,
  random_order = FALSE,
  colors = rev(colorspace::heat_hcl(10)),
  ...
)
```

Arguments

<code>x</code>	either a <code>cranly_network</code> object or a named vector of term frequencies (typically the output of <code>compute_term_frequency()</code> with <code>frequency = "term"</code>).
<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.

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.
maintainer	a vector of character strings with the maintainer names to be matched. Default is Inf which returns all available maintainers 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.
exact	logical. Should we use exact matching? Default is TRUE.
perspective	should the wordcloud be that of package descriptions (perspective = "description"; default), of package titles (perspective = "title") or of author names (perspective = "author").
random_order	should words be plotted in random order? If FALSE (default) words are plotted in decreasing frequency.
ignore_words	a vector of words to be ignored when forming the corpus.
stem	should words be stemmed using Porter's stemming algorithm? Default is FALSE. See <code>tm::stemDocument()</code> .
colors	color words from least to most frequent
...	other arguments to be passed to <code>wordcloud::wordcloud</code> (except <code>random.order</code> which is already defined through <code>random_order</code>).

Details

When applied to `cranly_network` objects, `word_cloud()` subsets either according to author (using the intersection of the result of `author_of()` and `author_with()`) or according to package (using the intersection of the results of `package_with()` and `package_by()`).

For handling more complex queries, one can manually extract the #' term frequencies from a supplied vector of character strings (see `compute_term_frequency()`), and use `word_cloud()` on them. See the examples.

Value

A word cloud.

See Also

[compute_term_frequency\(\)](#)

Examples

```
## Package directives network
cran_db <- clean_CRAN_db()
package_network <- build_network(cran_db)
## Descriptions of all packages in tidyverse
tidyverse <- imported_by(package_network, "tidyverse", exact = TRUE)
set.seed(123)
word_cloud(package_network, package = tidyverse, exact = TRUE, min.freq = 2)
```

```
## or by manually creating the term frequencies from descriptions
descriptions <- descriptions_of(package_network, tidyverse, exact = TRUE)
term_freq <- compute_term_frequency(descriptions)
set.seed(123)
word_cloud(term_freq, min.freq = 2)
```

Index

- as.igraph.cranly_network, 2
- author_of (extractor-functions), 11
- author_of(), 24
- author_with (extractor-functions), 11
- author_with(), 24
- authors_of (extractor-functions), 11
- authors_with (extractor-functions), 11

- build_dependence_tree, 3
- build_dependence_tree.cranly_network, 4
- build_dependence_tree.cranly_network(), 9, 15, 21
- build_network
 - (build_network.cranly_db), 5
- build_network.cranly_db, 5
- build_network.cranly_db(), 13

- clean_CRAN_db, 6
- clean_CRAN_db(), 5, 6
- clean_up_author, 7
- clean_up_author(), 6, 7
- clean_up_directives, 8
- clean_up_directives(), 6
- compute_dependence_tree, 8
- compute_dependence_tree(), 4, 15, 21
- compute_term_frequency, 9
- compute_term_frequency(), 23, 24
- cranly, 10
- cranly_db, 5, 6
- cranly_db (clean_CRAN_db), 6
- cranly_dependence_tree, 4, 15, 20
- cranly_dependence_tree
 - (build_dependence_tree.cranly_network), 4, 13
- cranly_network, 2–5, 8, 13–16, 19–21, 23, 24
- cranly_network
 - (build_network.cranly_db), 5

- data.frame, 5–7, 23

- dependency_of (extractor-functions), 11
- depending_on (extractor-functions), 11
- description_of (extractor-functions), 11
- descriptions_of (extractor-functions), 11

- email_of (extractor-functions), 11
- email_with (extractor-functions), 11
- emails_of (extractor-functions), 11
- emails_with (extractor-functions), 11
- enhanced_by (extractor-functions), 11
- enhancing (extractor-functions), 11
- extractor-functions, 11

- FALSE, 5, 18, 21

- igraph::betweenness(), 22
- igraph::closeness(), 22
- igraph::degree(), 22
- igraph::eigen_centrality(), 22
- igraph::graph, 2, 3
- igraph::page_rank(), 22
- imported_by (extractor-functions), 11
- importing (extractor-functions), 11
- Inf, 4, 16, 19, 23

- license_of (extractor-functions), 11
- licenses_of (extractor-functions), 11
- linked_by (extractor-functions), 11
- linking_to (extractor-functions), 11

- maintained_by (extractor-functions), 11
- maintainer_of (extractor-functions), 11

- package_by (extractor-functions), 11
- package_by(), 24
- package_with (extractor-functions), 11
- package_with(), 24
- packages_by (extractor-functions), 11

packages_with (extractor-functions), 11
plot.cranly_dependence_tree, 14
plot.cranly_dependence_tree(), 4
plot.cranly_network, 16
plot.cranly_network(), 5, 13
plot.summary_cranly_network, 17

release_date_of (extractor-functions),
11
release_dates_of (extractor-functions),
11

standardize_whitespace, 18
standardize_whitespace(), 6
subset.cranly_network, 19
subset.cranly_network(), 5, 13
suggested_by (extractor-functions), 11
suggesting (extractor-functions), 11
summary.cranly_dependence_tree, 20
summary.cranly_dependence_tree(), 4
summary.cranly_network, 18, 21
summary_cranly_network, 18
summary_cranly_network
(summary.cranly_network), 21

title_of (extractor-functions), 11
titles_of (extractor-functions), 11
tm::DocumentTermMatrix, 10
tm::DocumentTermMatrix(), 9
tm::removeNumbers, 9
tm::removePunctuation(), 9
tm::removeWords, 9
tm::stemDocument(), 9, 24
tm::stopwords, 9
tm::TermDocumentMatrix, 10
tm::TermDocumentMatrix(), 9
tools::CRAN_package_db(), 6
TRUE, 2, 15, 17, 21

utils::available.packages(), 6

version_of (extractor-functions), 11
versions_of (extractor-functions), 11
visNetwork::visEdges(), 15, 17

word_cloud (word_cloud.cranly_network),
23
word_cloud(), 10, 24
word_cloud.cranly_network, 23
wordcloud::wordcloud, 24