Package: repo.data (via r-universe)

June 20, 2026

Title R Repositories Data

Version 0.2.2

Description Retrieve metadata about packages from repositories to explore package dependencies, links between help pages, aliases, package availability on a given date, and other repository-dependent outcome. In addition, it provides access to information about the processes at CRAN. This metadata can be used to help package maintainers and users navigate changes to dependencies and with reproducibility.

License GPL (>= 3)

URL https://github.com/llrs/repo.data, https://repo.data.llrs.dev/

BugReports https://github.com/llrs/repo.data/issues

Depends R (>= 4.5)

Imports methods, tools, utils

Suggests igraph (>= 2.1), knitr (>= 1.49), markdown (>= 1.13), rversions (>= 2.1), spelling (>= 2.3)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 8.0.0

NeedsCompilation no

Author Lluís Revilla Sancho [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9747-2570>)

Maintainer Lluís Revilla Sancho <lluis.revilla@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-20 18:51:15 UTC

RemoteUrl https://github.com/cran/repo.data

RemoteRef HEAD

RemoteSha aec175cbe895c5da1221abbcad4ea354b5fa5cb4

Contents

alias *Links*

Description

Retrieve links of repository packages to other packages' documentation.

Usage

```
alias(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with three columns: Package, Source and Target. NA if not able to collect the data from the repository.

Note

For completeness it also provides the alias of R packages themselves.

See Also

Other alias: base_alias(), cran_alias()

Examples

```
oldrepos <- getOption("repos")
setRepositories(ind = c(1, 2), addURLs = "https://cran.r-project.org")
# show repositories
getOption("repos")

head(alias(c("ggplot2", "BiocCheck")))


# Clean  up
options(repos = oldrepos)
```

base_alias *Base R's alias*

Description

Retrieve alias available on R.

Usage

```
base_alias(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with three columns: Package, Source and Target. NA if not able to collect the data from CRAN.

See Also

The raw source of the data is: base_aliases_db().

Other alias: alias(), cran_alias()

Examples

```
ba <- base_alias()
head(ba)
```

base_help_cliques *Help pages with cliques*

Description

Some help pages have links to and from but they are closed networks.

Usage

```
base_help_cliques()
```

Details

Requires igraph

Value

Return a data.frame of help pages not connected to the network of help pages. NA if not able to collect the data from CRAN.

See Also

Other functions related to BASE help pages: base_help_pages_not_linked(), base_help_pages_wo_links()

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
    base_help_cliques()
}
```

---

base_help_pages_not_linked

Help pages without links

Description

Help pages without links to other help pages. This makes harder to navigate to related help pages.

Usage

```
base_help_pages_not_linked()
```

Value

A data.frame with two columns: Package and Source. NA if not able to collect the data from CRAN.

See Also

Other functions related to BASE help pages: base_help_cliques(), base_help_pages_wo_links()

Examples

```
bhnl <- base_help_pages_not_linked()
head(bhnl)
```

base_help_pages_wo_links *Help pages not linked from base R*

Description

Help pages without links from other help pages. This makes harder to find them.

Usage

```
base_help_pages_wo_links()
```

Value

A data.frame with two columns: Package and Source

See Also

Other functions related to BASE help pages: `base_help_cliques()`, `base_help_pages_not_linked()`

Examples

```
bhwl <- base_help_pages_wo_links()
head(bhwl)
```

base_links *Base R's links*

Description

Retrieve links on R documentation files.

Usage

```
base_links(packages = NULL)
```

Arguments

packages	A vector with packages or `NULL` for all packages.

Value

A data.frame with the links on R's files. It has 4 columns: Package, Anchor, Target and Source. `NA` if not able to collect the data from CRAN.

See Also

The raw source of the data is: base_rdxrefs_db().

Other links from R: base_pages_links(), base_pkges_links(), base_targets_links()

Examples

```
bl <- base_links()
head(bl)
```

---

base_pages_links *Links between help pages by page*

Description

Explore the relationship between base R packages and other help pages. If the target help page is ambiguous it is omitted.

Usage

```
base_pages_links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, from_Rd, to_pkg, to_Rd, n (Number of links). NA if not able to collect the data from CRAN.

See Also

Other links from R: base_links(), base_pkges_links(), base_targets_links()

Examples

```
bpl <- base_pages_links()
head(bpl)
```

base_pkges_links *Links between help pages by package*

Description

Explore the relationship between base R packages and other packages. If the target package is ambiguous it is omitted.

Usage

```
base_pkges_links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, to_pkg, n (Number of links). NA if not able to collect the data from CRAN.

See Also

Other links from R: base_links(), base_pages_links(), base_targets_links()

Examples

```
bpkl <- base_pkges_links()
head(bpkl)
```

base_targets_links *Links between help pages by target*

Description

Explore the relationship between base R packages and other help pages by the target they use.

Usage

```
base_targets_links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, from_Rd, to_pkg, to_target, to_Rd, n (Number of links). NA if not able to collect the data from CRAN.

See Also

Other links from R: base_links(), base_pages_links(), base_pkges_links()

Examples

```
btl <- base_targets_links()
head(btl)
```

---

bioc_cran_archived *Bioconductor packages using CRAN archived packages*

Description

Checks on the 4 Bioconductor repositories which packages depend on a archived package.

Usage

```
bioc_cran_archived(which = "strong")
```

Arguments

which	a character vector listing the types of dependencies, a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"). Character string "all" is shorthand for that vector, character string "most" for the same vector without "Enhances", character string "strong" (default) for the first three elements of that vector.

Value

A data.frame with the name of the Bioconductor packages depending on archived packages (on Archived column) and the number of missing packages (n). NA if not able to collect the data.

See Also

For CRAN's data source: tools::CRAN_package_db()

Examples

```
bca <- bioc_cran_archived()
head(bca)
```

clean_cache *Clean cache*

Description

Clean cache to download repository data again.

Usage

```
clean_cache()
```

Details

Cleans the package's environment used for caching the data.

Value

Returns NULL.

Examples

```
clean_cache()
```

cran_actions *Look at the CRAN actions db*

Description

CRAN tracks movements of packages and the actions used (for example to report the number of manual actions taken by the volunteers).

Usage

```
cran_actions(packages = NULL, silent = FALSE)
```

Arguments

packages	A vector with packages or NULL for all packages.
silent	A logical value to issue warnings about the data or not.

Details

There are three possible actions with packages source code: publish, archive and remove.

- Publish: Add it to CRAN's PACKAGES file, users can install that version.
- Archive: Removed from CRAN's repository PACKAGES file so users can't access the package with `available.packages()`. Remains on CRAN archive: `https://cran.r-project.org/src/contrib/Archive/`.
- Remove: Removed from CRAN's archive.

Value

A data.frame with Date, Time, User, Action, Package and Version columns. `NA` if not able to collect the data from CRAN.

See Also

Other meta info from CRAN: `cran_alias()`, `cran_archive()`, `cran_comments()`, `cran_doom()`, `cran_links()`, `links()`

Examples

```
ca <- cran_actions(silent = TRUE)
head(ca)
```

---

cran_alias *CRAN's alias*

Description

Retrieve alias available on CRAN.

Usage

```
cran_alias(packages = NULL)
```

Arguments

packages	A vector with packages or `NULL` for all packages.

Value

A data.frame with three columns: Package, Source and Target. `NA` if not able to collect the data from CRAN.

See Also

The raw source of the data is: CRAN_aliases_db().

Other alias: alias(), base_alias()

Other meta info from CRAN: cran_actions(), cran_archive(), cran_comments(), cran_doom(), cran_links(), links()

Examples

```
ca <- cran_alias("BWStest")
head(ca)
```

---

cran_archive *Retrieve CRAN archive*

Description

Retrieve the archive and the current database.

Usage

```
cran_archive(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Details

Some packages would get an NA in Version, if package_version() were to be used with strict = FALSE. Packages might have been on CRAN but could have been removed and won't show up. Depending on the data requested and packages currently on CRAN, you might get a warning regarding a package being both archived and current.

Value

A data.frame with 6 columns: Package, Date (of publication), Version, User, size and status (archived or current). It is sorted by package name and date. NA if not able to collect the data from CRAN.

See Also

The raw source of the data is: CRAN_archive_db(), CRAN_current_db(). For some dates and comments about archiving packages: cran_comments().

Other meta info from CRAN: cran_actions(), cran_alias(), cran_comments(), cran_doom(), cran_links(), links()

Examples

```
ap <- available.packages()
if (NROW(ap)) {
    a_package <- rownames(ap)[startsWith(rownames(ap), "A")][2]
    ca <- cran_archive(a_package)
    head(ca)
}
```

`cran_comments` *CRAN comments*

Description

CRAN volunteers document since ~2009 why they archive packages. This function retrieves the data and prepares it for analysis, classifying the actions taken by the team per package and date.

Usage

```
cran_comments(packages = NULL)
```

Arguments

`packages`	A vector with packages or `NULL` for all packages.

Details

The comments are slightly edited: multiple comments for the same action are joined together so that they can be displayed on a single line. Actions are inferred from 7 keywords: archived, orphaned, removed, renamed, replaced, unarchived, unorphaned.

Value

A data.frame with four columns: package, comment, date and action. `NA` if not able to collect the data from CRAN.

Note

There can be room for improvement: some comments describe two actions, please let me know if you think this can be improved. Other actions can be described on multiple comments/lines or out of order. Compare with the original file in case of doubts.

References

Original file: https://cran.r-project.org/src/contrib/PACKAGES.in

See Also

Other meta info from CRAN: `cran_actions()`, `cran_alias()`, `cran_archive()`, `cran_doom()`, `cran_links()`, `links()`

Examples

```
cc <- cran_comments()
head(cc)
```

`cran_date` *Estimate CRAN's date of packages*

Description

Check which CRAN dates are possible for a given packages and versions.

Usage

```
cran_date(versions)

cran_session(session = sessionInfo())
```

Arguments

`versions`	A data.frame with the packages names and versions
`session`	Session information.

Value

Last installation date from CRAN.

See Also

Other utilities: `cran_doom()`, `cran_snapshot()`, `duplicated_alias()`, `package_date()`, `package_repos()`, `repos_dependencies()`, `update_dependencies()`

Examples

```
# ip <- installed.packages()
ip <- data.frame(Package = c("A3", "AER"), Version = c("1.0.0", "1.2-15"))
cran_date(ip)
cran_session()
```

`cran_doom` *Calculate time till packages are archived*

Description

Given the deadlines by the CRAN volunteers packages can be archived which can trigger some other packages to be archived. This code calculates how much time the chain reaction will go on if maintainer don't fix/update the packages.

Usage

```
cran_doom(which = "strong", bioc = FALSE)
```

Arguments

`which`	a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.
`bioc`	Logical value if Bioconductor packages should be provided, (Requires internet connection).

Details

Packages on Suggested: field should

Value

A list with multiple elements:

- time_till_last: Time till last package is affected.
- last_archived: the date of the last package that would be affected.
- npackages: Numeric vector with the number of packages used.
- details: A data.frame with information for each individual package: Name, date affected, affected directly, repository, times it is affected (by archival causing issues.) `NA` if not able to collect the data from CRAN.

References

Original code from: https://github.com/schochastics/cran-doomsday/blob/main/index.qmd

See Also

The raw source of the data is: `tools::CRAN_package_db()`

Other utilities: `cran_date()`, `cran_snapshot()`, `duplicated_alias()`, `package_date()`, `package_repos()`, `repos_dependencies()`, `update_dependencies()`

Other meta info from CRAN: `cran_actions()`, `cran_alias()`, `cran_archive()`, `cran_comments()`, `cran_links()`, `links()`

Examples

```
cd <- cran_doom()
if (length(cd) > 1L) head(cd$details)
```

---

cran_help_cliques *Help pages with cliques*

Description

Some help pages have links to other pages and they might be linked from others but they are closed network: there is no link that leads to different help pages. Each group of linked help pages is a clique.

Usage

```
cran_help_cliques(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Details

The first clique is the biggest one. You might want to check if others cliques can be connected to this one.

Requires igraph.

Value

Return a data.frame of help pages not connected to the network of help pages. Or NULL if nothing are found. NA if not able to collect the data from CRAN.

See Also

Other functions related to CRAN help pages: `cran_help_pages_not_linked()`, `cran_help_pages_wo_links()`

Examples

```
chc <- cran_help_cliques("BaseSet")
if (!is.null(dim(chc))) {
   table(chc$clique)
   chc[chc$clique != 1L, ]
}
```

`cran_help_pages_links_wo_deps` *Links without dependencies*

Description

On WRE section "2.5 Cross-references" explains that packages shouldn't link to help pages outside the dependency.

Usage

```
cran_help_pages_links_wo_deps(packages = NULL)
```

Arguments

`packages`	A vector with packages or `NULL` for all packages.

Value

A data.frame of help pages and links. `NA` if not able to collect the data from CRAN.

References

https://cran.r-project.org/doc/manuals/r-devel/R-exts.html#Cross_002dreferences

Examples

```
evmix <- cran_help_pages_links_wo_deps("evmix")
```

cran_help_pages_not_linked *Help pages without links*

Description

Help pages without links to other help pages. This makes harder to navigate to related help pages.

Usage

```
cran_help_pages_not_linked(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with two columns: Package and Source NA if not able to collect the data from CRAN.

See Also

Other functions related to CRAN help pages: cran_help_cliques(), cran_help_pages_wo_links()

Examples

```
ap <- available.packages()
if (NROW(ap)) {
    a_package <- rownames(ap)[startsWith(rownames(ap), "A")][1]
    chnl <- cran_help_pages_not_linked(a_package)
    head(chnl)
}
```

cran_help_pages_wo_links *Help pages not linked*

Description

Help pages without links from other help pages. This makes harder to find them.

Usage

```
cran_help_pages_wo_links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with two columns: Package and Source NA if not able to collect the data from CRAN.

See Also

Other functions related to CRAN help pages: cran_help_cliques(), cran_help_pages_not_linked()

Examples

```
ap <- available.packages()
if (NROW(ap)) {
    a_package <- rownames(ap)[startsWith(rownames(ap), "a")][1]
    chwl <- cran_help_pages_wo_links(a_package)
    head(chwl)
}
```

---

cran_issues *CRAN issues*

Description

Reports the notifications sent to package maintainers with the hour it was sent and the deadline used.

Usage

```
cran_issues()
```

Value

A data.frame with 8 columns:

ID Year.number: Id of the issue.

Package Package name.

Date POSIXct object when the issue was sent.

From CRAN member that sent that notification.

Before Date by which the issue should be fixed.

Title Reason of the issue.

Label Some classification of the issue.

Info Unstructured text with some information.

Examples

```
ci <- cran_issues()
if (!is.null(dim(ci))) {
    head(ci)
}
```

`cran_links` *CRAN's links*

Description

Retrieve links on CRAN packages' R documentation files.

Usage

```
cran_links(packages = NULL)
```

Arguments

`packages`	A vector with packages or `NULL` for all packages.

Value

A data.frame with the links on CRAN's packages. It has 4 columns: Package, Anchor, Target and Source. `NA` if not able to collect the data from CRAN.

See Also

The raw source of the data is: `CRAN_rdxrefs_db()`.

Other links from CRAN: `cran_pages_links()`, `cran_pkges_links()`, `cran_targets_links()`, `links()`

Other meta info from CRAN: `cran_actions()`, `cran_alias()`, `cran_archive()`, `cran_comments()`, `cran_doom()`, `links()`

Examples

```
cl <- cran_links("CytoSimplex")
head(cl)
```

cran_maintainers *Tidy information about maintainers*

Description

Make more accessible information about maintainers. Extracts and makes comparable some dates. It also provides the user name used and cleans the names of the maintainer of extra quotes.

Usage

```
cran_maintainers()
```

Details

User is what the machine building the package reported. This might indicate some collaboration, repackaging or simply nothing as it can be hidden/modified. The name, email and user might help identify similar named people (or confuse between maintainers)

Value

A data.frame with one row per package and 11 columns. The package name, Maintainer field, user maintainer manual date, packaged date, published date, name of maintainer used, email used, direction and domain. NA if not able to collect the data from CRAN.

See Also

The raw source of the data is: CRAN_authors_db()

Examples

```
maintainers <- cran_maintainers()
head(maintainers)
```

cran_pages_links *Links between help pages by page*

Description

Explore the relationship between CRAN packages and other help pages. If the target help page is ambiguous it is omitted.

Usage

```
cran_pages_links(packages = NULL)
```

Arguments

packages A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, from_Rd, to_pkg, to_Rd, n (Number of links).

See Also

Other links from CRAN: cran_links(), cran_pkges_links(), cran_targets_links(), links()

Examples

```
cpl <- cran_pages_links("Matrix")
head(cpl)
```

---

cran_pkges_links *Links between help pages by package*

Description

Explore the relationship between CRAN packages and other packages. If the target package is ambiguous it is omitted.

Usage

```
cran_pkges_links(packages = NULL)
```

Arguments

packages A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, to_pkg, n (Number of links). NA if not able to collect the data from CRAN.

See Also

Other links from CRAN: cran_links(), cran_pages_links(), cran_targets_links(), links()

Examples

```
cpkl <- cran_pkges_links()
head(cpkl)
```

cran_snapshot *Check CRAN package state on any given date*

Description

Given the available information which packages were on CRAN on a given date?

Usage

```
cran_snapshot(date)
```

Arguments

date	The date you want to check.

Value

The data.frame with the packages and versions at a given date. NA if not able to collect the data from CRAN.

Note

Due to missing of CRAN comments some packages are not annotated when were they archived and more packages than present might be returned for any given date.

See Also

Other utilities: cran_date(), cran_doom(), duplicated_alias(), package_date(), package_repos(), repos_dependencies(), update_dependencies()

Examples

```
cs <- cran_snapshot(Sys.Date() -2 )
head(cs)
```

cran_targets_links *Links between help pages by target*

Description

Explore the relationship between CRAN packages and other help pages by the target they use.

Usage

```
cran_targets_links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with 6 columns: from_pkg, from_Rd, to_pkg, to_target, to_Rd, n (Number of links).

See Also

Other links from CRAN: cran_links(), cran_pages_links(), cran_pkges_links(), links()

Examples

```
ctl <- cran_targets_links("BaseSet")
head(ctl)
```

---

cran_version *Install a specific version of a package*

Description

Install a package from CRAN of a specific version.

Usage

```
cran_version(package, version, ...)
```

Arguments

package	Name of the package present on CRAN archive.
version	The version number.
...	Other arguments passed to install.packages.

Details

Uses CRAN specific API

<cran.r-project.org/package=%s&version=%s>

to install a package. As this is an archived copy we only use source version.

Value

Same as install.packages().

References

CRAN pages.

Examples

```
## Not run:
cran_version("repo.data", "0.1.5", lib = tempdir())

## End(Not run)
```

`duplicated_alias` *Report duplicated alias*

Description

Report duplicated alias

Usage

```
duplicated_alias(alias)
```

Arguments

`alias`	The output of `cran_alias()` or `base_alias()`

Value

A sorted data.frame with the Target, Package and Source of the duplicate alias.

See Also

Other utilities: `cran_date()`, `cran_doom()`, `cran_snapshot()`, `package_date()`, `package_repos()`, `repos_dependencies()`, `update_dependencies()`

Examples

```
# Checking the overlap between to seemingly unrelated packages:
alias <- alias(c("fect", "gsynth"))
if (length(alias)) {
   dup_alias <- duplicated_alias(alias)
   head(dup_alias)
}
```

links *Links*

Description

Retrieve links of repository packages to other packages' documentation.

Usage

```
links(packages = NULL)
```

Arguments

packages	A vector with packages or NULL for all packages.

Value

A data.frame with the links on packages. It has 4 columns: Package, Anchor, Target and Source. NA if not able to collect the data from the repository.

See Also

Other links from CRAN: cran_links(), cran_pages_links(), cran_pkges_links(), cran_targets_links()

Other meta info from CRAN: cran_actions(), cran_alias(), cran_archive(), cran_comments(), cran_doom(), cran_links()

Examples

```
oldrepos <- getOption("repos")
setRepositories(ind = c(1, 2), addURLs = "https://cran.r-project.org")

head(links(c("ggplot2", "BiocCheck")))

# Clean  up
options(repos = oldrepos)
```

os_alias *Base R OS specific alias*

Description

Data from the R source code of OS specific man help pages. This is to complement tools::base_aliases_db() which only provides links for Unix.

Usage

```
os_alias
```

Format

An object of class `matrix` (inherits from `array`) with 33 rows and 5 columns.

Value

A matrix with 33 rows and 5 columns:

Package Package name

os OS system where this applies

file Name of the Rd file.

Source Path to the file.

Target Name of the Target

Examples

```
os_alias
```

---

package_date *Find earliest date of compatibility*

Description

Search the DESCRIPTION file for the release dates of dependencies and return the earliest date according to CRAN's archive. This is the date at which the package could be installed.

Usage

```
package_date(packages = ".", which = "strong")
```

Arguments

`packages`	Path to the package folder and/or name of packages published.
`which`	a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.

Details

Currently this function assumes that packages only use ">=" and not other operators. This might change on the future if other operators are more used.

Value

A vector with the datetimes of the published package (or current date if not published) and the datetime when the requirements were met. NA if not able to collect the data from CRAN.

See Also

Other utilities: cran_date(), cran_doom(), cran_snapshot(), duplicated_alias(), package_repos(), repos_dependencies(), update_dependencies()

Examples

```
package_date("ABACUS")
package_date("paramtest")
package_date("Seurat") # Dependencies on packages not on CRAN
package_date("afmToolkit") # Dependency was removed from CRAN
```

---

package_date_actions *Package dates*

Description

Same as package_date() but using CRAN's actions instead of public archive.

Usage

```
package_date_actions(packages = ".", which = "strong")
```

Arguments

packages	Name of the package on CRAN. It accepts also local path to packages source directories but then the function works as if the package is not released yet.
which	a character vector listing the types of dependencies, a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"). Character string "all" is shorthand for that vector, character string "most" for the same vector without "Enhances", character string "strong" (default) for the first three elements of that vector.

Details

This provides information about when a package was removed or archived for a more accurate estimation.

Value

A vector with the time when the package was publish and when were the dependencies available.

Examples

```
package_date_actions("afmToolkit")
```

package_dependencies *Find current installations*

Description

Despite the description minimal requirements find which versions are required due to dependencies. Reports the minimal version for each package that would be installed now.

Usage

```
package_dependencies(packages = ".", which = "strong")
```

Arguments

packages	Path to a folder with a DESCRIPTION file or package's names from a repository. If NULL will pick all packages and their dependencies available.
which	a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.

Value

A data.frame with the Package, Type, Name, Op and Version required. If only one package requires it it also show the name of the package. NA if not able to collect the data from repositories.

Note

It keeps the base packages too even if just knowing the R version required would be enough.

Examples

```
pd <- package_dependencies("ggeasy")
head(pd)
```

package_repos *Package dependencies to repositories*

Description

Explore the relationships between packages and repositories.

Usage

```
package_repos(packages = NULL, repos = getOption("repos"), which = "all")
```

Arguments

packages	a character vector of package names.
repos	Repositories and their names are taken from `getOptions("repos")`.
which	a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.

Value

A data.frame with one line per package and at least one column per repository. It also has a column for Other repositories (Additional_repositories, or missing repositories), and the total number of dependencies and total number of repositories used. NA if not able to collect the data from repositories.

See Also

Other utilities: `cran_date()`, `cran_doom()`, `cran_snapshot()`, `duplicated_alias()`, `package_date()`, `repos_dependencies()`, `update_dependencies()`

Examples

```
pr <- package_repos("experDesign")
head(pr)
```

---

repos_dependencies *Tidy dependencies*

Description

Extract the packages dependencies, name of the dependency, operator and version for each type and package of current repositories (`getOptions("repos")`).

Usage

```
repos_dependencies(packages = NULL, which = "all")
```

Arguments

packages	a character vector of package names.
which	a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.

Value

A data.frame with 5 columns: the name of the dependency, the operator (op), the version it depends the type of dependency and the package. NA if not able to collect the data from CRAN.

See Also

Other utilities: cran_date(), cran_doom(), cran_snapshot(), duplicated_alias(), package_date(), package_repos(), update_dependencies()

Examples

```
rd <- repos_dependencies("BaseSet")
head(rd)
```

---

update_dependencies *Upgradable versions*

Description

Helper function to detect which package have a required version on the dependencies that could be upgraded.

Usage

```
update_dependencies(packages)
```

Arguments

packages	A character vector of packages names.

Details

Increasing this version requirements won't affect users as they already should have these versions installed as required by other dependencies.

Value

The data.frame filtered with some relevant rows. NA if not able to collect the data from repositories.

See Also

package_dependencies()

Other utilities: cran_date(), cran_doom(), cran_snapshot(), duplicated_alias(), package_date(), package_repos(), repos_dependencies()

Examples

```
update_dependencies("arrow")
```

Index