Package: pacs (via r-universe)

November 22, 2026

Title Supplementary Tools for R Packages Developers

Version 0.7.1

Maintainer Maciej Nasinski <nasinski.maciej@gmail.com>

Description Supplementary utils for CRAN maintainers and R packages developers. Validating the library, packages and lock files. Exploring a complexity of a specific package like evaluating its size in bytes with all dependencies. The shiny app complexity could be explored too. Assessing the life duration of a specific package version. Checking a CRAN package check page status for any errors and warnings. Retrieving a DESCRIPTION or NAMESPACE file for any package version. Reporting declared system requirements for packages and their dependencies. Comparing DESCRIPTION or NAMESPACE files between different package versions. Getting a list of all releases for a specific package. The Bioconductor is partly supported.

License GPL (>= 3)

URL https://github.com/Polkas/pacs, https://polkas.github.io/pacs/

BugReports https://github.com/Polkas/pacs/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports cachem, curl, memoise (>= 2.0.0), jsonlite, xml2, stringi, yaml

Suggests remotes, renv, withr, pkgsearch, mockery, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Maciej Nasinski [aut, cre]

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

Date/Publication 2026-11-22 22:35:18 UTC

RemoteUrl https://github.com/cran/pacs

RemoteRef HEAD

RemoteSha 58b3d297a1567bd4c7e02453a8b2bc73554682b2

Contents

app_deps . . . 3
app_size . . . 4
bio_releases . . . 5
biocran_repos . . . 6
checked_packages . . . 6
compareVersionsMax . . . 7
compareVersionsMin . . . 8
cran_flavors . . . 8
dir_size . . . 9
lib_validate . . . 9
lock_validate . . . 11
match_flavors . . . 13
pac_checkpage . . . 14
pac_checkred . . . 14
pac_compare_namespace . . . 15
pac_compare_news . . . 16
pac_compare_versions . . . 17
pac_deps . . . 18
pac_deps_dev . . . 20
pac_deps_heavy . . . 21
pac_deps_timemachine . . . 22
pac_deps_user . . . 23
pac_description . . . 24
pac_health . . . 25
pac_isin . . . 26
pac_islast . . . 27
pac_last . . . 28
pac_lifeduration . . . 29
pac_namespace . . . 30
pac_news . . . 31
pac_size . . . 32
pac_system_requirements . . . 33
pac_timemachine . . . 34
pac_true_size . . . 36
pac_validate . . . 37
pacs_base . . . 39
pacs_lifeduration . . . 39

Index **41**

app_deps *The shiny app dependencies*

Description

the shiny app dependencies packages are checked recursively. The `c("Depends", "Imports", "LinkingTo")` DESCRIPTION files fields are checked recursively. The required dependencies have to be installed in the local repository. The default arguments setup is recommended.

Usage

```
app_deps(
  path = ".",
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  local = TRUE,
  base = FALSE,
  description_v = FALSE,
  recursive = TRUE,
  repos = biocran_repos()
)
```

Arguments

`path`	`character` path to the shiny app. Default: `"."`
`fields`	`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. Default: `c("Depends", "Imports", "LinkingTo")`
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`
`local`	`logical` if to use local repository (or newest remote packages). Default: TRUE
`base`	`logical` if to add base packages too. If `TRUE` then `pacs::pacs_base()` are taken into account. Default: `FALSE`
`description_v`	`logical` if the dependencies version should be taken from description files, minimal required. By default installed versions are taken. Default: `FALSE`
`recursive`	`logical` if to assess the dependencies recursively. Default: TRUE
`repos`	`character` vector base URLs of the repositories to use. By default checking CRAN and newest Bioconductor per R version. Default `pacs::biocran_repos()`

Value

`character` vector with dependency packages or data.frame when checking recursively.

Note

renv package has to be installed.

Examples

```
## Not run:
library(renv)
# Please update the path to the custom shiny app
app_path <- system.file("examples/04_mpg", package = "shiny")
pacs::app_deps(app_path)
pacs::app_deps(app_path, recursive = FALSE)

## End(Not run)
```

---

app_size *Size of the shiny app*

Description

The size of shiny app is a sum of dependencies packages and the app directory. The app dependencies packages are checked recursively, and only in local repository. The default arguments setup is recommended.

Usage

```
app_size(
  path = ".",
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  recursive = TRUE
)
```

Arguments

path	character path to the shiny app. Default: "."
fields	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. Default: c("Depends", "Imports", "LinkingTo")
lib.loc	character vector of search paths with local packages. Default: .libPaths()
recursive	logical if to assess the dependencies recursively. Default: TRUE

Value

numeric size in bytes, or NA if dependencies cannot be determined. To get MB, divide by 10**6.

Note

renv package has to be installed. base packages (pacs::pacs_base()) are not taken into account.

Examples

```
## Not run:
library(renv)
# Please update the path to the shiny app
cat(pacs::app_size(system.file("examples/04_mpg", package = "shiny")) / 10**6, "MB")

## End(Not run)
```

---

bio_releases *Retrieving all Bioconductor releases*

Description

Retrieving all Bioconductor releases. The data is downloaded from Bioconductor's machine-readable https://bioconductor.org/config.yaml.

Usage

```
bio_releases()
```

Value

data.frame with Release, Date, Software packages, and R columns, or NA if the configuration cannot be read. The configuration does not provide historical software-package counts, so Software packages is NA. The R column is NA for releases without an R version mapping.

Note

Only versions up to the current Bioconductor release with a release date are included, newest first. Dates and R versions follow the Bioconductor configuration and may differ from other historical sources. Successful results are cached for 30 minutes with the memoise package; failures are not cached.

Examples

```
## Not run:
pacs::bio_releases()

## End(Not run)
```

biocran_repos *CRAN and Bioconductor repositories*

Description

CRAN and Bioconductor repositories. The newest Bioconductor release for the specific R version is assumed.

Usage

```
biocran_repos(version = NULL)
```

Arguments

version	character the Bioconductor release. By default the newest Bioconductor release for the specific R version is assumed, if not available only CRAN repository is returned. Available Bioconductor versions for your R version could be checked with pacs::bio_releases(). Default NULL

Value

named character vector of repositories.

Note

The Internet connection is needed to get Bioconductor repositories. If the release configuration is unavailable or no release matches, a message is issued and only the CRAN repository is returned.

Examples

```
## Not run:
pacs::biocran_repos()

## End(Not run)
```

checked_packages *Retrieving all R CRAN packages check pages statuses.*

Description

Retrieving all R CRAN packages check pages statuses. The data is downloaded from https://cran.r-project.org/web/c

Usage

```
checked_packages()
```

Value

data.frame with the same structure as the html table on https://cran.r-project.org/web/checks/check_summary_by_

Note

Results are cached for 30 minutes with memoise package. Some packages could be duplicated as not all tests are performed for a new version so two versions still coexists. Checks with asterisks (*) indicate that checking was not fully performed, this is a case for less than 1% of all packages.

Examples

```
## Not run:
pacs::checked_packages()

## End(Not run)
```

---

compareVersionsMax *Maximum version across the vector*

Description

Reduce function over the utils::compareVersion

Usage

```
compareVersionsMax(vec, na.rm = TRUE)
```

Arguments

vec	character vector.
na.rm	logical if to remove NA values.

Value

character maximum version

Examples

```
compareVersionsMax(c("1.1.1", "0.2.0"))
```

compareVersionsMin *Minimum version across the vector*

Description

Reduce function over the `utils::compareVersion`

Usage

```
compareVersionsMin(vec, na.rm = TRUE)
```

Arguments

vec	`character` vector.
na.rm	`logical` if to remove `NA` values.

Value

`character` minimal version

Examples

```
compareVersionsMin(c("1.1.1", "0.2.0"))
```

cran_flavors *Retrieving all R CRAN servers flavors*

Description

Retrieving all R CRAN servers flavors. The data is downloaded from `https://cran.r-project.org/web/checks/check_f`

Usage

```
cran_flavors()
```

Value

`data.frame` with the same structure as the html table on `https://cran.r-project.org/web/checks/check_flavors.htm`

Note

Results are cached for 30 minutes with `memoise` package.

Examples

```
## Not run:
pacs::cran_flavors()

## End(Not run)
```

dir_size *Size of the package*

Description

size of package.

Usage

```
dir_size(path = ".", recursive = TRUE)
```

Arguments

path	character path to the shiny app. Default: "."
recursive	logical if to assess the dependencies recursively. Default: TRUE

Value

numeric size in bytes, to get MB ten divide by 10**6.

Examples

```
## Not run:
cat(pacs::dir_size(system.file(package = "stats")) / 10**6, "MB")

## End(Not run)
```

lib_validate *Validate the local library*

Description

Checking if installed packages have correct versions taking into account all DESCRIPTION files requirements. Moreover identifying which packages are newest releases. Optionally we could add life duration and CRAN check page status for each package.

Usage

```
lib_validate(
  lib.loc = .libPaths(),
  fields = c("Depends", "Imports", "LinkingTo"),
  lifeduration = FALSE,
  checkred = list(scope = character(0), flavors = NULL),
  built = FALSE,
  repos = biocran_repos()
)
```

Arguments

lib.loc	character vector of search paths with local packages. Default: .libPaths()
fields	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. Default: c("Depends", "Imports", "LinkingTo")
lifeduration	logical if to assess life duration for each package in the library. For installed newest releases of packages, a local evaluation is used. MEATCRAN CRANDB is used for libraries with less than 500 packages. Otherwise the direct web page download from CRAN is used. Default: 'FALSE"
checkred	list with two named fields, scope and flavor. scope of R CRAN check pages statuses to consider, any of c("ERROR", "FAIL", "WARN", "NOTE"). flavor is a vector of CRAN machines to consider, which might be retrieved with pacs::cran_flavors()$Flavor. By default an empty scope field deactivated assessment for checkred column, and NULL flavor will results in checking all machines. Default: list(scope = character(0), flavor = NULL)
built	logical if to add an R version under which each package was installed. Useful mainly for a local usage. Packages installed with a previous version of R could not work correctly with the new version of R. Default: FALSE
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

data.frame with 4/6/8/9/10 columns.

Package character a package name.

Version.expected.min character expected by DESCRIPTION files minimal version. "" means not specified so the newest version.

Version.have character installed package version.

version_status numeric -1/0/1 which comes from utils::compareVersion function. 0 means that we have the same version as required by DESCRIPTION files. -1 means we have too low version installed, this is an error. 1 means we have higher version.

built character package was built under this R version

built_status integer if the package was built under the current R version, then 1 (good) and for older R versions 0 (possibly bad). A package built under older R version or mix of packages built under different versions could bring possible failures.

newest logical (Internet needed) if the installed version is the newest one. For Bioconductor if is the newest one per R version.

cran logical (Internet needed) if the package is on CRAN, version is not taken into account here.

checkred (Optional) (Internet needed) logical if the NEWEST package contains any specified statuses on CRAN check page. pacs::checked_packages is used to quickly retrieve all statuses at once.

lifeduration (Optional) (Internet needed) integer number of days a package was released.

Note

Version.expected.min column not count packages which are not a dependency for any package, so could not be find in DESCRIPTION files. When turn on the `lifeduration` options, calculations might be time consuming for libraries bigger than 500 packages. Results are cached for 30 minutes with `memoise` package. `BioConductor` packages are tested only in available scope, `checkred` is not assessed for them. The `crandb` R packages database is a part of `METACRAN` project, source: Csárdi G, Salmon M (2022). `pkgsearch`: Search and Query CRAN R Packages. `https://github.com/r-hub/pkgsearch`, `https://r-hub.github.io/pkgsearch/`.

Examples

```
## Not run:
pacs::lib_validate()
pacs::lib_validate(checkred = list(scope = c("ERROR", "FAIL", "WARN")))
pacs::lib_validate(checkred = list(
  scope = c("ERROR", "FAIL"),
  flavors = pacs::match_flavors()
))
# activate lifeduration argument, could be time consuming for bigger libraries.
pacs::lib_validate(
  lifeduration = TRUE,
  checkred = list(scope = c("ERROR", "FAIL"))
)
# only R CRAN repository
pacs::lib_validate(repos = "https://cran.rstudio.com/")

## End(Not run)
```

`lock_validate` *Validate a specific renv lock file*

Description

This function will be especially useful when renv lock file is built manually. Checking if packages in the lock file have correct versions taking into account their DESCRIPTION files requirements (`c("Depends", "Imports", "LinkingTo")`). Moreover identifying which packages are newest releases. Optionally we could add life duration and CRAN check page status for each dependency.

Usage

```
lock_validate(
  path,
  lifeduration = FALSE,
  checkred = list(scope = character(0), flavors = NULL),
  lib.loc = .libPaths(),
  repos = biocran_repos()
)
```

Arguments

path	character path to the shiny app. Default: "."
lifeduration	logical if to assess life duration for each package in the library. For installed newest releases of packages, a local evaluation is used. MEATCRAN CRANDB is used for libraries with less than 500 packages. Otherwise the direct web page download from CRAN is used. Default: 'FALSE"
checkred	list with two named fields, scope and flavor. scope of R CRAN check pages statuses to consider, any of c("ERROR", "FAIL", "WARN", "NOTE"). flavor is a vector of CRAN machines to consider, which might be retrieved with pacs::cran_flavors()$Flavor. By default an empty scope field deactivated assessment for checkred column, and NULL flavor will results in checking all machines. Default: list(scope = character(0), flavor = NULL)
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

data.frame with 2/6/7/8 columns.

Package character a package name.

Version.expected.min (conditional) (Internet needed) character expected by DESCRIPTION files minimal version. "" means not specified so the newest version.

Version.expected character package version in the renv lock file.

version_status (conditional) numeric -1/0/1 which comes from utils::compareVersion function. 0 means that we have the same version as required by DESCRIPTION files. -1 means we have too low version installed, this is an error. 1 means we have higher version.

newest logical (Internet needed) if the installed version is the newest one.

cran logical (Internet needed) if the package is on CRAN, version is not taken into account here.

checkred (Optional) (Internet needed) logical if the NEWEST package contains any specified statuses on CRAN check page.

lifeduration (Optional) (Internet needed) integer number of days a package was released.

Note

Version.expected.min column not count packages which are not a dependency for any package, so could not be find in DESCRIPTION files. Version.expected.min and version_status are assessed only if there are less than 500 packages in the lock file. When turn on the lifeduration option, calculations might be time consuming when there is more than 500 packages. The crandb R packages database is a part of METACRAN project, source: Csárdi G, Salmon M (2022). pkgsearch: Search and Query CRAN R Packages. https://github.com/r-hub/pkgsearch, https://r-hub.github.io/pkgsearch/

Examples

```
## Not run:
# path or url
url <- "https://raw.githubusercontent.com/Polkas/pacs/master/tests/testthat/files/renv_test.lock"
pacs::lock_validate(url)

pacs::lock_validate(
  url,
  checkred = list(scope = c("ERROR", "FAIL"), flavors = pacs::match_flavors())
)

pacs::lock_validate(
  url,
  lifeduration = TRUE,
  checkred = list(scope = c("ERROR", "FAIL"), flavors = NULL)
)

## End(Not run)
```

`match_flavors` *Get all matched CRAN servers to the local* `OS`

Description

CRAN servers matched to the local `OS`.

Usage

```
match_flavors()
```

Value

`character` vector matched server names.

Note

The Internet connection is needed to use the function.

Examples

```
## Not run:
pacs::match_flavors()

## End(Not run)
```

pac_checkpage *Retrieving the R CRAN package check page*

Description

Retrieving the R CRAN package check page.

Usage

```
pac_checkpage(pac)
```

Arguments

pac	character a package name.

Value

data.frame.

Note

Results are cached for 30 minutes with memoise package. If you need to check many packages at once then is recommended usage of pacs::checked_packages. Please as a courtesy to the R CRAN, don't overload their server by constantly using this function.

Examples

```
## Not run:
pacs::pac_checkpage("dplyr")

## End(Not run)
```

pac_checkred *Checking the R CRAN package check page status*

Description

using package R CRAN check page to validate if there are ANY errors and/or fails and/or warnings and/or notes.

Usage

```
pac_checkred(pac, scope = c("ERROR", "FAIL"), flavors = NULL)
```

Arguments

pac	character a package name.
scope	character vector scope of the check, accepted values c("ERROR", "FAIL", "WARN", "NOTE"). Default: c("ERROR", "FAIL")
flavors	character vector of CRAN server names to consider, possible names could be get with pacs::cran_flavors()$Flavor. The pacs::match_flavors() function could be used to get CRAN server names matched for your local OS. By default all CRAN machines are considered NULL value. Default: NULL

Value

logical if the package fails under the specified criteria, or NA if the required check statuses cannot be retrieved.

Note

Results are cached for 30 minutes with memoise package. If you need to check many packages at once then is recommended usage of pacs::checked_packages. The used repository https://cran.rstudio.com/. Please as a courtesy to the R CRAN, don't overload their server by constantly using this function.

Examples

```
## Not run:
pacs::pac_checkred("dplyr")
pacs::pac_checkred("dplyr", scope = c("ERROR"))
pacs::pac_checkred("dplyr",
  scope = c("ERROR", "FAIL", "WARN"),
  flavors = pacs::match_flavors()
)

## End(Not run)
```

---

pac_compare_namespace *Compare NAMESPACE exports between specific CRAN packages versions*

Description

using the remote github CRAN mirror to compare NAMESPACE exports between specific packages versions.

Usage

```
pac_compare_namespace(
  pac,
  old = NULL,
  new = NULL,
```

```
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

pac	character a package name.
old	character an old version of package, default local version. Default: NULL
new	character a new version of package, default newest version. Default: NULL
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector repositories URLs to use. Used only for the validation. Default https://cran.rstudio.com/

Value

list with c("imports", "exports", "exportPatterns", "importClasses", "importMethods", "exportClasses", "exportMethods", "exportClassPatterns", "dynlibs", "S3methods") slots, and added and removed ones for each of them.

Examples

```
## Not run:
pacs::pac_compare_namespace("shiny", "1.0.0", "1.6.0")
pacs::pac_compare_namespace("shiny", "1.0.0", "1.6.0")$exports
# local version to newest one
pacs::pac_compare_namespace("shiny")

## End(Not run)
```

---

pac_compare_news *Compare NEWS files between specific CRAN packages versions*

Description

using the remote github CRAN mirror to compare NEWS files between specific packages versions.

Usage

```
pac_compare_news(
  pac,
  old = NULL,
  new = NULL,
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

pac	character a package name.
old	character an old version of package, default local version. Default: NULL
new	character a new version of package, default newest version. Default: NULL
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector repositories URLs to use. Used only for the validation. Default https://cran.rstudio.com/

Value

character with NEWS content between specific versions, or NA if the requested version headings cannot be found in release order.

Note

Versions are matched against the first complete version number in each Markdown heading.

Examples

```
## Not run:
pacs::pac_compare_news("shiny", "1.0.0", "1.6.0")
# local version to newest one
pacs::pac_compare_news("shiny")

## End(Not run)
```

---

pac_compare_versions *Compare DESCRIPTION files dependencies between specific CRAN packages versions*

Description

using the remote github CRAN mirror to compare DESCRIPTION files dependencies between specific packages versions.

Usage

```
pac_compare_versions(
  pac,
  old = NULL,
  new = NULL,
  fields = c("Imports", "Depends", "LinkingTo"),
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

pac	character a package name.
old	character an old version of package, default local version. Default: NULL
new	character a new version of package, default newest version. Default: NULL
fields	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. Default: c("Depends", "Imports", "LinkingTo")
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector repositories URLs to use. Used only for the validation. Default https://cran.rstudio.com/

Value

data.frame with 4 columns.

Package character package names.

Version.OLD character versions of dependencies required by an old package version.

Version.NEW character versions of dependencies required by a new package version.

version_status numeric -1/0/1 which comes from utils::compareVersion function. 0 means that both versions have the same requirement. -1 means that the new version remove this requirement. 1 means that the new version added a new requirement.

Examples

```
## Not run:
pacs::pac_compare_versions("memoise", "0.2.1", "2.0.0")
pacs::pac_compare_versions("memoise", "0.2.1")
# local version to newest one
pacs::pac_compare_versions("memoise")

## End(Not run)
```

---

pac_deps *Package dependencies*

Description

Package dependencies from DESCRIPTION files with installed or expected versions or newest released.

Usage

```
pac_deps(
  pac,
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  base = FALSE,
  local = TRUE,
  description_v = FALSE,
  attr = TRUE,
  recursive = TRUE,
  repos = biocran_repos()
)
```

Arguments

pac	character a package name.
fields	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. Default: c("Depends", "Imports", "LinkingTo")
lib.loc	character vector of search paths with local packages. Default: .libPaths()
base	logical if to add base packages too. If TRUE then pacs::pacs_base() are taken into account. Default: FALSE
local	logical if to use local repository (or newest remote packages). Default: TRUE
description_v	logical if the dependencies version should be taken from description files, minimal required. By default installed versions are taken. Default: FALSE
attr	logical if a package and its version should be added as an attribute of data.frame or for FALSE as an additional record. Default: TRUE
recursive	logical if to assess the dependencies recursively. Default: TRUE
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

data.frame with packages and their versions. Versions are taken from installed.packages or newest released.

Note

When function is invoked in the loop afterwards results could be aggregated like, stats::aggregate(results[, c("Version"), drop = FALSE], list(Package = results$Package), pacs::compareVersionsMax).

Examples

```
## Not run:
pacs::pac_deps("stats", base = TRUE)$Package
pacs::pac_deps("memoise")$Package
pacs::pac_deps("memoise", description_v = FALSE)
# raw dependencies from DESCRIPTION file
pacs::pac_deps("memoise", description_v = TRUE, recursive = FALSE)
# raw dependencies from DESCRIPTION file - last release
pacs::pac_deps("memoise", description_v = TRUE, local = FALSE, recursive = FALSE)

## End(Not run)
```

---

pac_deps_dev *Package dependencies - developer perspective*

Description

A higher-level function, build from `pacs::pacs_deps`. Package dependencies installed when e.g. `R CMD check` a package. `"Depends", "Imports", "LinkingTo", "Suggests"` fields from the DESCRIPTION file and their recursive dependencies taken from `"Depends", "Imports", "LinkingTo"` fields. Dependencies are taken remotely for the newest version.

Usage

```
pac_deps_dev(
  pac,
  base = FALSE,
  local = FALSE,
  attr = TRUE,
  repos = pacs::biocran_repos()
)
```

Arguments

`pac`	`character` a package name.
`base`	`logical` if to add base packages too. If `TRUE` then `pacs::pacs_base()` are taken into account. Default: `FALSE`
`local`	`logical` if to use local repository (or newest remote packages). Default: `FALSE`
`attr`	`logical` if a package and its version should be added as an attribute of data.frame or for FALSE as an additional record. Default: TRUE
`repos`	`character` vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default `pacs::biocran_repos()`

Value

`data.frame` with packages and their versions. Versions are taken from `installed.packages` or newest released.

Examples

```
## Not run:
pacs::pac_deps_dev("dplyr")
pacs::pac_deps_dev("pacs")
# with the main package in the list
pacs::pac_deps_dev("pacs", attr = FALSE)

## End(Not run)
```

---

pac_deps_heavy *Package direct dependencies and number of dependencies for each of them*

Description

A higher-level function, build from `pacs::pacs_deps` and `tools::package_dependencies`. A tool to identify a main sources of dependencies, which direct dependencies are the heaviest one.

Usage

```
pac_deps_heavy(
  pac,
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  base = FALSE,
  local = FALSE,
  repos = pacs::biocran_repos()
)
```

Arguments

pac	`character` a package name.
fields	`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. Default: `c("Depends", "Imports", "LinkingTo")`
lib.loc	`character` vector of search paths with local packages. Default: `.libPaths()`
base	`logical` if to add base packages too. If `TRUE` then `pacs::pacs_base()` are taken into account. Default: `FALSE`
local	`logical` if to use local repository (or newest remote packages). Default: `FALSE`
repos	`character` vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default `pacs::biocran_repos()`

Value

`data.frame` with three columns `c("Package", "NrDeps", "NrUniqueDeps")`: package name, number of dependencies and number of unique dependencies (not shared by other direct dependencies).

Note

Please take into account that the sum of the dependencies is not equal to the number of dependencies of the main package, because some dependencies are overlapping.

Examples

```
## Not run:
pacs::pac_deps_heavy("caret")
pacs::pac_deps_heavy("dplyr")

## End(Not run)
```

---

pac_deps_timemachine *R CRAN package dependencies for a certain version or time point*

Description

Package dependencies from DESCRIPTION files retrieved recursively for certain version or time point.

Usage

```
pac_deps_timemachine(
  pac,
  version = NULL,
  at = NULL,
  fields = c("Depends", "Imports", "LinkingTo"),
  recursive = TRUE
)
```

Arguments

`pac`	`character` a package name.
`version`	`character` version of a package. Default: `NULL`
`at`	`Date` from which to take the version. Default: `NULL`
`fields`	`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. Default: `c("Depends", "Imports", "LinkingTo")`
`recursive`	`logical` if to assess the dependencies recursively. Default: TRUE

Value

named vector package dependencies and their versions at the release date of main package plus one day.

Note

Longer lived version is taken if 2 is available at the same date (switch time).

Examples

```
## Not run:
pacs::pac_deps_timemachine("memoise", "0.2.1")
pacs::pac_deps_timemachine("memoise", at = as.Date("2019-01-01"))
pacs::pac_deps_timemachine("dplyr", at = as.Date("2015-01-01"))

## End(Not run)
```

---

pac_deps_user *Package dependencies - user perspective*

Description

A higher-level function, build from pacs::pacs_deps. Package dependencies installed when run installed.packages. "Depends", "Imports", "LinkingTo" fields from the DESCRIPTION file and their recursive dependencies taken from the same fields. Dependencies are taken remotely for the newest version.

Usage

```
pac_deps_user(
  pac,
  base = FALSE,
  local = FALSE,
  attr = TRUE,
  repos = pacs::biocran_repos()
)
```

Arguments

pac	character a package name.
base	logical if to add base packages too. If TRUE then pacs::pacs_base() are taken into account. Default: FALSE
local	logical if to use local repository (or newest remote packages). Default: FALSE
attr	logical if a package and its version should be added as an attribute of data.frame or for FALSE as an additional record. Default: TRUE
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

`data.frame` with packages and their versions. Versions are taken from `installed.packages` or newest released.

Examples

```
## Not run:
pacs::pac_deps_user("dplyr")
pacs::pac_deps_user("pacs")
# with the main package in the list
pacs::pac_deps_user("pacs", attr = FALSE)

## End(Not run)
```

---

pac_description *package DESCRIPTION file*

Description

CRAN package DESCRIPTION file taken locally or remotely from GITHUB CRAN mirror or CRAN website. By default works for the newest package version.

Usage

```
pac_description(
  pac,
  version = NULL,
  at = NULL,
  local = FALSE,
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

`pac`	`character` a package name.
`version`	`character` version of a package. Default: `NULL`
`at`	`Date` from which to take the version. Default: `NULL`
`local`	`logical` if to use local repository (or newest remote packages). Default: `FALSE`
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`
`repos`	`character` vector repositories URLs to use. Used only for the validation. Default `https://cran.rstudio.com/`

Value

`list` with names proper for DESCRIPTION file fields.

Note

Results are cached for 30 minutes with `memoise` package.

Examples

```
## Not run:
pacs::pac_description("dplyr", version = "0.8.0")
pacs::pac_description("dplyr", at = as.Date("2019-02-01"))

## End(Not run)
```

---

pac_health *CRAN package health state at a specific Date or for a specific version*

Description

a package health for a certain version or at a specific Date. By default works for the newest package version. A healthy package was published for more than x days, where default is 14 days. CRAN team gives around one/two week to resolved a package which gave errors under the check page. The newest release is checked for any warnings/errors on the R CRAN package check page.

Usage

```
pac_health(
  pac,
  version = NULL,
  at = NULL,
  limit = 14,
  scope = c("ERROR", "FAIL"),
  flavors = NULL,
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/",
  source = c("cran", "crandb")
)
```

Arguments

`pac`	`character` a package name.
`version`	`character` version of a package. Default: `NULL`
`at`	`Date` from which to take the version. Default: `NULL`
`limit`	`numeric` at least days to treat as healthy, ">=limit". Default: 14
`scope`	`character` vector scope of the check, accepted values c("ERROR", "FAIL", "WARN", "NOTE"). Default: `c("ERROR", "FAIL")`
`flavors`	`character` vector of CRAN server names to consider, possible names could be get with `pacs::cran_flavors()$Flavor`. The `pacs::match_flavors()` function could be used to get CRAN server names matched for your local `OS`. By default all CRAN machines are considered `NULL` value. Default: `NULL`

lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector repositories URLs to use. Default https://cran.rstudio.com/
source	character one of c("cran", "crandb"). Using the METACRAN CRANDB or the direct web page download from CRAN. When "crandb" is set then options(pacs.crandb_ntry = 3) and options(pacs.crandb_nsleep = 0.001) can be used to control the fetch. "pacs.crandb_ntry" can be used to set the maximum number of try outs, by default 3. "pacs.crandb_nsleep" can be used to set the sleep duration between fetch try outs, by default 0.001. Default: "cran"

Value

logical if a package is healthy, or NA when the required release or CRAN check information is unavailable.

Note

Results are cached for 30 minutes with memoise package. The crandb R packages database is a part of METACRAN project, source: Csárdi G, Salmon M (2022). pkgsearch: Search and Query CRAN R Packages. https://github.com/r-hub/pkgsearch, https://r-hub.github.io/pkgsearch/. For source = "cran"the function will scrap two CRAN URLS. Works only with CRAN packages. Please as a courtesy to the R CRAN, don't overload their servers by constantly using this function.

Examples

```
## Not run:
pacs::pac_health("memoise")
pacs::pac_health("dplyr", version = "0.8.0", limit = 14)
pacs::pac_health("dplyr", at = as.Date("2019-02-14"))
pacs::pac_health("dplyr", limit = 14, scope = c("ERROR", "FAIL"))

## End(Not run)
```

---

pac_isin *Checking if a package is in repositories*

Description

using utils::available.packages to check if package is in repositories.

Usage

```
pac_isin(pac, repos = biocran_repos())
```

Arguments

pac	character a package name.
repos	character vector base URLs of the repositories to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

`logical` if a package is inside repositories.

Note

Results are cached for 30 minutes with `memoise` package.

Examples

```
## Not run:
pac_isin("dplyr")
pac_isin("dplyr", repos = "https://cran.rstudio.com/")
pac_isin("dplyr", repos = biocran_repos()[grep("Bio", names(biocran_repos()))])

## End(Not run)
```

---

pac_islast *Checking if a package version is the most recent one*

Description

checking if a package version is the most recent one, by default the installed version is compared.

Usage

```
pac_islast(pac, version = NULL, lib.loc = .libPaths(), repos = biocran_repos())
```

Arguments

`pac`	`character` a package name.
`version`	`character` version of a package. Default: `NULL`
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`
`repos`	`character` vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default `pacs::biocran_repos()`

Value

`logical` if a package is inside repositories.

Note

Results are cached for 30 minutes with `memoise` package.

Examples

```
## Not run:
pac_islast("memoise")
pac_islast("dplyr", version = "1.0.0")
pac_islast("S4Vectors")
pac_islast("S4Vectors", version = pac_last("S4Vectors"))

## End(Not run)
```

`pac_last` *Getting the most recent package version*

Description

using `utils::available.packages` to get the newest package version.

Usage

```
pac_last(pac, repos = biocran_repos())
```

Arguments

`pac`	`character` a package name.
`repos`	`character` vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default `pacs::biocran_repos()`

Value

`character` most recent package version.

Note

Results are cached for 30 minutes with `memoise` package. For Bioconductor the newest one per R version.

Examples

```
## Not run:
pac_last("dplyr")
pac_last("S4Vectors")

## End(Not run)
```

pac_lifeduration *Package version life duration at specific Date or for a specific version*

Description

a package life duration for a certain version or at a specific Date. By default works for the newest package version.

Usage

```
pac_lifeduration(
  pac,
  version = NULL,
  at = NULL,
  lib.loc = .libPaths(),
  repos = biocran_repos(),
  source = c("cran", "crandb")
)
```

Arguments

pac	character a package name.
version	character version of a package. Default: NULL
at	Date from which to take the version. Default: NULL
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()
source	character one of c("cran", "crandb"). Using the MEATCRAN CRANDB or the direct web page download from CRAN. When "crandb" is set then options(pacs.crandb_ntry = 3) and options(pacs.crandb_nsleep = 0.001) can be used to control the fetch. "pacs.crandb_ntry" can be used to set the maximum number of try outs, by default 3. "pacs.crandb_nsleep" can be used to set the sleep duration between fetch try outs, by default 0.001. Default: "cran"

Value

difftime, number of days package version was the newest one.

Note

Results are cached for 30 minutes with memoise package. The crandb R packages database is a part of METACRAN project, source: Csárdi G, Salmon M (2022). pkgsearch: Search and Query CRAN R Packages. https://github.com/r-hub/pkgsearch, https://r-hub.github.io/pkgsearch/. For source = "cran"the function will scrap two CRAN URLS. Works only with CRAN packages. Please as a courtesy to the R CRAN, don't overload their servers by constantly using this function.

Examples

```
## Not run:
pacs::pac_lifeduration("memoise")
pacs::pac_lifeduration("memoise", source = "cran")
pacs::pac_lifeduration("dplyr", version = "0.8.0")
pacs::pac_lifeduration("dplyr", at = as.Date("2019-02-14"))
# For Bioconductor packages it will work only for the newest per R version and installed ones.
pacs::pac_lifeduration("S4Vectors")

## End(Not run)
```

---

pac_namespace *package NAMESPACE file*

Description

CRAN package NAMESPACE file taken locally or remotely from GITHUB CRAN mirror or CRAN website. By default works for the newest package version.

Usage

```
pac_namespace(
  pac,
  version = NULL,
  at = NULL,
  local = FALSE,
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

pac	character a package name.
version	character version of a package. Default: NULL
at	Date from which to take the version. Default: NULL
local	logical if to use local repository (or newest remote packages). Default: FALSE
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector repositories URLs to use. Used only for the validation. Default https://cran.rstudio.com/

Value

list with names proper for NAMESPACE file, the same as format as returned by base::parseNamespaceFile.

Note

Results are cached for 30 minutes with `memoise` package. This function is mainly built under source code from `base::parseNamespaceFile`.

Examples

```
## Not run:
pacs::pac_namespace("dplyr", version = "0.8.0")
pacs::pac_namespace("dplyr", at = as.Date("2019-02-01"))
pacs::pac_namespace("memoise", local = TRUE)

## End(Not run)
```

---

pac_news *Get NEWS for a package*

Description

Get NEWS for a package from CRAN or local

Usage

```
pac_news(
  pac,
  version = NULL,
  at = NULL,
  local = FALSE,
  lib.loc = .libPaths(),
  repos = "https://cran.rstudio.com/"
)
```

Arguments

`pac`	`character` a package name.
`version`	`character` version of a package. Default: `NULL`
`at`	`Date` from which to take the version. Default: `NULL`
`local`	`logical` if to use local repository (or newest remote packages). Default: `FALSE`
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`
`repos`	`character` vector repositories URLs to use. Used only for the validation. Default `https://cran.rstudio.com/`

Value

`character` with NEWS content.

Note

Results are cached for 30 minutes with `memoise` package.

Examples

```
## Not run:
pacs::pac_news("dplyr", version = "0.8.0")
pacs::pac_news("dplyr", at = as.Date("2019-02-01"))

## End(Not run)
```

`pac_size` *Size of the package*

Description

Size of package.

Usage

```
pac_size(pac, lib.loc = .libPaths())
```

Arguments

`pac`	`character` a package name.
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`

Value

`numeric` size in bytes, to get MB ten divide by `10**6`.

Examples

```
## Not run:
cat(pacs::pac_size("stats") / 10**6, "MB")

## End(Not run)
```

pac_system_requirements *Package system requirements*

Description

Report the SystemRequirements fields declared by a package and its dependencies. These fields are free text and can describe optional or platform-specific requirements; this function does not check whether they are installed or resolve dependency version constraints. It reports the installed versions or the versions selected from the repository indexes.

Usage

```
pac_system_requirements(
  pac,
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  local = TRUE,
  recursive = TRUE,
  recursive_fields = c("Depends", "Imports", "LinkingTo"),
  repos = biocran_repos()
)
```

Arguments

pac	character a package name.
fields	Dependency fields to follow for the requested package. In addition to the values described in pac_deps(), character(0) considers only the requested package.
lib.loc	character vector of search paths with local packages. Default: .libPaths()
local	Use installed packages when TRUE (the default), or current source packages in repos when FALSE.
recursive	Follow dependencies recursively. Default: TRUE.
recursive_fields	Dependency fields to follow for indirect dependencies. Defaults to c("Depends", "Imports", "LinkingTo"). Accepts the same values as fields, including character(0).
repos	Repositories to use for source packages when local = FALSE. Default: biocran_repos(). Bulk repository metadata is used when available; otherwise source archives are downloaded to read their DESCRIPTION files.

Details

With recursive = FALSE, only the package and its direct dependencies are considered. With recursive = TRUE, dependencies are followed recursively. By default, Suggests and Enhances are only followed for the requested package when included in fields; indirect dependencies use recursive_fields. R and its base packages are excluded from the dependencies, but the requested package is always included.

Value

A named `list`, similar to `tools::package_dependencies()`. Names are package names, starting with `pac`, and each element is a `character` vector of declared system requirements. The `SystemRequirements` field is split on top-level commas and semicolons (not inside parentheses), with whitespace normalized; fields written as sentences are kept whole. Splitting is heuristic because the field is free text. Packages without a declared field have `character(0)`. Dependencies missing from the library or repositories have `NA_character_` and produce a warning, because their own dependencies cannot be assessed. The `"versions"` attribute is a named `character` vector with the installed or repository versions. Use `unique(unlist(result))` to get all requirements.

Note

By default, source repository indexes are filtered for the current R version and platform. Bulk metadata and downloaded DESCRIPTION fields are cached for 30 minutes. A missing `SystemRequirements` field does not guarantee that a package has no system requirements. Bioconductor bulk metadata can normalize whitespace in the declared field.

Examples

```
pac_system_requirements("stats", fields = character())
## Not run:
pac_system_requirements("curl", recursive = FALSE)
pac_system_requirements("sf", local = FALSE, repos = "https://cran.r-project.org")
sf <- pac_system_requirements("sf", fields = "most", local = FALSE)
sf[["sf"]]
# all requirements of sf and its dependencies
unique(unlist(sf, use.names = FALSE))

## End(Not run)
```

---

pac_timemachine *Package metadata for all releases*

Description

Using CRAN website to get a package metadata used at a specific Date or a Date interval or for specific version.

Usage

```
pac_timemachine(
  pac,
  at = NULL,
  from = NULL,
  to = NULL,
  version = NULL,
  source = c("cran", "crandb")
)
```

Arguments

pac	character a package name.
at	Date from which to take the version. Default: NULL
from	Date the lower limit. Default: NULL
to	Date the upper limit. Default: NULL
version	character version of a package. Default: NULL
source	character one of c("cran", "crandb"). Using the MEATCRAN CRANDB or the direct web page download from CRAN. When "crandb" is set then options(pacs.crandb_ntry = 3) and options(pacs.crandb_nsleep = 0.001) can be used to control the fetch. "pacs.crandb_ntry" can be used to set the maximum number of try outs, by default 3. "pacs.crandb_nsleep" can be used to set the sleep duration between fetch try outs, by default 0.001. Default: "cran"

Value

data.frame with 7 columns

Package character package name.

Version character package version.

Released character release Date

Archived character archived Date.

LifeDuration difftime number of days the version was the newest one.

URL character the suffix of the base URL to tar.gz file. The base part of URL in the result is https://cran.r-project.org/src/contrib/.

Size character size of the tar.gz file.

Note

Results are cached for 30 minutes with memoise package. The crandb R packages database is a part of METACRAN project, source: Csárdi G, Salmon M (2022). pkgsearch: Search and Query CRAN R Packages. https://github.com/r-hub/pkgsearch, https://r-hub.github.io/pkgsearch/. For source = "cran"the function will scrap two CRAN URLS. Works only with CRAN packages. Please as a courtesy to the R CRAN, don't overload their servers by constantly using this function.

Examples

```
## Not run:
pacs::pac_timemachine("dplyr")
pacs::pac_timemachine("dplyr", at = as.Date("2017-02-02"))
pacs::pac_timemachine("dplyr", from = as.Date("2017-02-02"), to = as.Date("2018-04-02"))
pacs::pac_timemachine("dplyr", at = Sys.Date())
pacs::pac_timemachine("tidyr", from = as.Date("2020-06-01"), to = Sys.Date())

## End(Not run)
```

`pac_true_size` *True size of the package*

Description

True size of the package as it takes into account its all dependencies, recursively.

Usage

```
pac_true_size(
  pac,
  fields = c("Depends", "Imports", "LinkingTo"),
  lib.loc = .libPaths(),
  exclude_joint = 0L
)
```

Arguments

`pac`	`character` a package name.
`fields`	`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. Default: `c("Depends", "Imports", "LinkingTo")`
`lib.loc`	`character` vector of search paths with local packages. Default: `.libPaths()`
`exclude_joint`	`integer` exclude packages which are dependencies of at least N other packages, not count main package dependencies. Default: `0`

Value

`numeric` size in bytes, to get MB then divide by `10**6`.

Note

R base packages are not counted. The default value of `fields` should be suited for almost all scenarios.

Examples

```
## Not run:
# size in MB, with all its dependencies
pacs::pac_true_size("memoise") / 10**6

## End(Not run)
```

pac_validate *Validate a specific local package*

Description

Checking if installed package dependencies have correct versions taking into account their DESCRIPTION files requirements. Moreover identifying which packages are newest releases. Optionally we could add life duration and CRAN check page status for each dependency.

Usage

```
pac_validate(
  pac,
  lib.loc = .libPaths(),
  fields = c("Depends", "Imports", "LinkingTo"),
  lifeduration = FALSE,
  checkred = list(scope = character(0), flavors = NULL),
  repos = biocran_repos()
)
```

Arguments

pac	character a package name.
lib.loc	character vector of search paths with local packages. Default: .libPaths()
fields	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. Default: c("Depends", "Imports", "LinkingTo")
lifeduration	logical if to assess life duration for each package in the library. For installed newest releases of packages, a local evaluation is used. MEATCRAN CRANDB is used for libraries with less than 500 packages. Otherwise the direct web page download from CRAN is used. Default: 'FALSE"
checkred	list with two named fields, scope and flavor. scope of R CRAN check pages statuses to consider, any of c("ERROR", "FAIL", "WARN", "NOTE"). flavor is a vector of CRAN machines to consider, which might be retrieved with pacs::cran_flavors()$Flavor. By default an empty scope field deactivated assessment for checkred column, and NULL flavor will results in checking all machines. Default: list(scope = character(0), flavor = NULL)
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

data.frame with 5/7/8/9 columns.

Package character a package name.

Version.expected.min character expected by DESCRIPTION files minimal version. "" means not specified so the newest version.

Version.have character installed package version.

version_status numeric -1/0/1 which comes from `utils::compareVersion` function. 0 means that we have the same version as required by DESCRIPTION files. -1 means we have too low version installed, this is an error. 1 means we have higher version.

direct logical if the package is in the first dependency layer, direct dependencies from DESCRIPTION file.

newest logical (Internet needed) if the installed version is the newest one.

cran logical (Internet needed) if the package is on CRAN, version is not taken into account here.

checkred (Optional) (Internet needed) logical if the NEWEST package contains any specified statuses on CRAN check page.

lifeduration (Optional) (Internet needed) integer number of days a package was released.

Note

Version.expected.min column not count packages which are not a dependency for any package, so could not be find in DESCRIPTION files. When turn on the `lifeduration` option, calculations might be time consuming when there is more than 500 packages. Please as a courtesy to the R CRAN, don't overload their server by constantly using this function with `lifeduration` or `checkred` turned on. Results are cached with `memoise` package, memory cache. The `crandb` R packages database is a part of `METACRAN` project, source: Csárdi G, Salmon M (2022). `pkgsearch`: Search and Query CRAN R Packages. `https://github.com/r-hub/pkgsearch`, `https://r-hub.github.io/pkgsearch/`

Examples

```
## Not run:
pacs::pac_validate("memoise")
pacs::pac_validate(
  "memoise",
  lifeduration = TRUE,
  checkred = list(scope = c("ERROR", "FAIL"), flavors = NULL)
)
pacs::pac_validate(
  "memoise",
  lifeduration = TRUE,
  checkred = list(scope = c("ERROR", "FAIL"), flavors = pacs::match_flavors())
)

## End(Not run)
```

pacs_base *Get base R packages*

Description

get base packages, all or only `startup`.

Usage

```
pacs_base(startup = FALSE)
```

Arguments

`startup`	`logical` include only `startup` packages. Default: `FALSE`

Value

`character` vector

Examples

```
## Not run:
pacs_base()
pacs_base(startup = TRUE)

## End(Not run)
```

pacs_lifeduration *Packages life duration for a specific version*

Description

packages life duration for certain versions.

Usage

```
pacs_lifeduration(
  pacs,
  versions,
  source = c("crandb", "loop_crandb", "loop_cran"),
  lib.loc = .libPaths(),
  repos = biocran_repos()
)
```

Arguments

pacs	character vector of packages names.
versions	character vector of packages versions.
source	character one of c("crandb", "loop_crandb", "loop_cran"). The "crandb" works if less than getOption("pacs.crandb_limit") (currently 500) packages are looked for. When crandb linked source is used then options(pacs.crandb_ntry = 3) and options(pacs.crandb_nsleep = 0.001) can be used to control the fetch. "pacs.crandb_ntry" can be used to set the maximum number of try outs, by default 3. "pacs.crandb_nsleep" can be used to set the sleep duration between fetch try outs, by default 0.001. Default: "crandb"
lib.loc	character vector of search paths with local packages. Default: .libPaths()
repos	character vector of repositories URLs to use. By default checking CRAN and newest Bioconductor per R version. Default pacs::biocran_repos()

Value

data.frame with two columns package name and life duration.

Note

Results are cached for 30 minutes with memoise package. The crandb R packages database is a part of METACRAN project, source: Csárdi G, Salmon M (2022). pkgsearch: Search and Query CRAN R Packages. https://github.com/r-hub/pkgsearch, https://r-hub.github.io/pkgsearch/. For source = "loop_cran"the function will scrap two CRAN URLS. Works only with CRAN packages. Please as a courtesy to the R CRAN, don't overload their servers by constantly using this function.

Examples

```
## Not run:
pacs::pacs_lifeduration(c("dplyr", "tidyr"), c("1.0.0", "1.2.0"))
pacs::pacs_lifeduration(c("dplyr", "tidyr"), c("1.0.0", "1.2.0"), source = "loop_cran")
# last versions
pacs::pacs_lifeduration(c("dplyr", "tidyr"), sapply(c("dplyr", "tidyr"), pacs::pac_last))

## End(Not run)
```

Index

app_deps, 3
app_size, 4

bio_releases, 5
biocran_repos, 6
biocran_repos(), *33*

checked_packages, 6
compareVersionsMax, 7
compareVersionsMin, 8
cran_flavors, 8

dir_size, 9

lib_validate, 9
lock_validate, 11

match_flavors, 13

pac_checkpage, 14
pac_checkred, 14
pac_compare_namespace, 15
pac_compare_news, 16
pac_compare_versions, 17
pac_deps, 18
pac_deps(), *33*
pac_deps_dev, 20
pac_deps_heavy, 21
pac_deps_timemachine, 22
pac_deps_user, 23
pac_description, 24
pac_health, 25
pac_isin, 26
pac_islast, 27
pac_last, 28
pac_lifeduration, 29
pac_namespace, 30
pac_news, 31
pac_size, 32
pac_system_requirements, 33
pac_timemachine, 34
pac_true_size, 36
pac_validate, 37
pacs_base, 39
pacs_lifeduration, 39

tools::package_dependencies(), *34*