Package: pacman (via r-universe)

October 26, 2026

Type Package

Title Package Management Tool

Version 0.5.3

Depends R (>= 3.5.0)

Imports remotes, methods, stats, utils

Suggests BiocManager, knitr, lattice, testthat (>= 0.9.0), XML

BugReports https://github.com/trinker/pacman/issues?state=open

Description Tools to more conveniently perform tasks associated with add-on packages. pacman conveniently wraps library and package related functions and names them in an intuitive and consistent fashion. It seeks to combine functionality from lower level functions which can speed up workflow.

License GPL-2

URL https://github.com/trinker/pacman

Roxygen list(wrap = FALSE)

RoxygenNote 6.1.1

Config/pak/sysreqs git

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

Date/Publication 2020-05-15 02:33:39 UTC

RemoteUrl https://github.com/trinker/pacman

RemoteRef HEAD

RemoteSha ace0936406ca0186a67ee0f5cd52698f663780db

Contents

p_author 2
p_base 3
p_boot 4
p_citation 4

p_cran . . . 5
p_data . . . 6
p_delete . . . 7
p_depends . . . 8
p_detectOS . . . 9
p_exists . . . 9
p_extract . . . 10
p_functions . . . 10
p_help . . . 11
p_information . . . 12
p_install . . . 13
p_install_gh . . . 14
p_install_version . . . 15
p_install_version_gh . . . 15
p_interactive . . . 16
p_isinstalled . . . 16
p_library . . . 17
p_load . . . 17
p_load_current_gh . . . 18
p_load_gh . . . 19
p_loaded . . . 20
p_news . . . 21
p_old . . . 22
p_opendir . . . 22
p_path . . . 23
p_search_any . . . 24
p_search_library . . . 25
p_set_cranrepo . . . 25
p_temp . . . 26
p_unload . . . 26
p_unlock . . . 27
p_update . . . 28
p_version . . . 29
p_vignette . . . 30
print.p_version_diff . . . 31
print.search_any . . . 31
print.wide_table . . . 32

Index **33**

`p_author` *Package Author*

Description

Returns the author of a package.

Usage

```
p_author(package = "base")
```

Arguments

package	Name of the package you want the author of.

See Also

packageDescription

Examples

```
p_author(pacman)
p_author()
```

---

p_base *Base Install Packages*

Description

List just base packages or list all the packages in the local library and mark those in a base install.

Usage

```
p_base(base.only = TRUE, open = FALSE, basemarker = "***")
```

Arguments

base.only	logical. If TRUE a character vector of only base install packages is returned.
open	logical. If TRUE opens the directory of the base install packages.
basemarker	Character string. The string to append to mark which packages are part of the default packages.

Note

Packages that are installed when R starts are marked with an asterisk(*).

See Also

getOption

Examples

```
## Not run:
p_base()
p_base(TRUE)

## End(Not run)
```

p_boot *Script Header: Ensure* **pacman** *is Installed*

Description

Generate a string for the standard **pacman** script header that, when added to scripts, will ensure **pacman** is installed before attempting to use it. **pacman** will attempt to copy this string (standard script header) to the clipboard for easy cut and paste.

Usage

```
p_boot(load = TRUE, copy2clip = interactive())
```

Arguments

load	logical. If TRUE ; library(pacman) is added to the end of the script header.
copy2clip	logical. If TRUE attempts to copy the output to the clipboard.

Details

The script header takes the form of:

```
if (!require("pacman")) install.packages("pacman"); library(pacman)
```

This can be copied to the top of scripts to make it easy to run scripts if the user shares them with others or to aid in long term script management. This may also be useful for blog posts and **R** help sites like TalkStats or StackOverflow. In this way functions like p_load can be used without fear that others don't have **pacman** installed.

Value

Returns a script header string (optionally copies to the clipboard).

Examples

```
p_boot()
```

p_citation *Package Citation*

Description

Generate citation for a package.

Usage

```
p_citation(package = "r", copy2clip = interactive(),
  tex = getOption("pac_tex"), ...)

p_cite(package = "r", copy2clip = interactive(),
  tex = getOption("pac_tex"), ...)
```

Arguments

package	Name of the package you want a citation for.
copy2clip	logical. If TRUE attempts to copy the output to the clipboard.
tex	logical. If TRUE only the BibTex version of the citation is copied to the clipboard. If FALSE the standard citation is copied to the clipboard. If NA both are copied to the clipboard. Default allows the user to set a "pac_tex" in his/her .Rprofile.
...	Additional inputs to citation

See Also

citation

Examples

```
## Not run:
p_citation()
p_cite(pacman)
p_citation(pacman, tex = FALSE)
p_citation(tex = FALSE)
p_cite(knitr)

## End(Not run)
```

---

p_cran *CRAN Packages*

Description

p_cran - Generate a vector of all available packages.

p_iscran - Logical check if a package is available on CRAN.

Usage

```
p_cran(menu = FALSE)

p_iscran(package)
```

Arguments

menu	logical. If TRUE allows user to select the package and return that package name.
package	Name of package.

See Also

available.packages

Examples

```
## Not run:
p_cran()
p_cran(TRUE)
p_iscran(pacman)

## End(Not run)
```

---

p_data *Package Data Sets*

Description

Generate a script of all data sets contained in package.

Usage

```
p_data(package = "datasets", static = FALSE)
```

Arguments

package	name of package (default is the base install datasets package).
static	logical. If TRUE a static text document is returned (e.g. data("datasets")).

Value

Returns the data sets of a package as a data.frame. (static = FALSE) or as a static text file (static = TRUE).

See Also

data

Examples

```
p_data()
p_data(lattice)
## Not run:
p_data(static=TRUE)

## End(Not run)
```

p_delete *Permanently Remove Package Removal(s) From Library*

Description

Remove package(s) from the library permanently.

Usage

```
p_delete(..., char, character.only = FALSE, quiet = FALSE)

p_del(..., char, character.only = FALSE, quiet = FALSE)
```

Arguments

char	Character vector containing packages to load. If you are calling p_delete from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
character.only	logical. If TRUE then p_load will only accept a single input which is a character vector containing the names of packages to load.
quiet	logical. Passed to print.p_delete as an attribute. If TRUE no messages confirming package deletions are printed.
...	name(s) of package(s).

Warning

Using this function will remove the package from your library and cannot be loaded again without reinstalling the package.

See Also

remove.packages

Examples

```
## Not run:
p_delete(pacman) # You never want to run this

## End(Not run)
```

p_depends *Package Dependencies*

Description

p_depends - Get CRAN or local package dependencies.

p_depends_reverse - Get CRAN or local reverse dependencies.

Usage

```
p_depends(package, local = FALSE, character.only = FALSE, ...)

p_depends_reverse(package, local = FALSE, character.only = FALSE, ...)
```

Arguments

package	Name of the package you want the list of dependencies/reverse dependencies for.
local	logical. If TRUE checks user's local library for existence; if FALSE CRAN for the package.
character.only	logical. If TRUE the input is a variable containing the package name.
...	other arguments passed to package_dependencies and dependsOnPkgs.

Value

Returns a list of dependencies/reverse dependencies.

See Also

p_info, package_dependencies, dependsOnPkgs

Examples

```
p_depends(lattice)
p_depends_reverse(lattice)

## Not run:
## dependencies from CRAN
p_depends(pacman)
p_depends_reverse("pacman")

## local dependencies
p_depends(pacman, local = TRUE)
p_depends_reverse("qdap", local = TRUE)

## End(Not run)
```

p_detectOS *Detects Operating System*

Description

Attempts to detect the operating system. Returns: "Windows", "Darwin" on Mac, "Linux", or "SunOS" on Solaris

Usage

```
p_detectOS()
```

p_exists *Checks if Package is On CRAN/In Local Library*

Description

Checks CRAN to determine if a package exists.

Usage

```
p_exists(package, local = FALSE)
```

Arguments

package	Name of package.
local	logical. If TRUE checks user's local library for existence; if FALSE CRAN for the package.

Examples

```
## Not run:
p_exists(pacman)
p_exists(pacman, FALSE)
p_exists(I_dont_exist)

## End(Not run)
```

p_extract *Convert String With Commas Into Elements*

Description

p_extract is designed to be used in conjunction with p_information to convert a single comma separated string into a vector of package names.

Usage

```
p_extract(x, use.names = TRUE)
```

Arguments

x	A character string of packages separated by commas; for example the strings returned from p_information.
use.names	logical. If TRUE package names, including version number, are used.

Value

Returns a character vector of packages.

See Also

p_information

Examples

```
## Not run:
p_extract(p_info(ggplot2, "Depends"))
p_extract(p_info(ggplot2, "Imports"))
lapply(p_info(ggplot2, "Imports", "Depends", "Suggests"), p_extract)

## End(Not run)
```

p_functions *Package Functions*

Description

List the functions from a package.

Usage

```
p_functions(package = "base", all = FALSE, character.only = FALSE)

p_funs(package = "base", all = FALSE, character.only = FALSE)
```

Arguments

package	Name of the package you want the list of functions for.
all	logical. If TRUE all of the functions from the package will be displayed regardless of whether they're exported or not.
character.only	logical. If TRUE the input is a variable containing the package name.

Examples

```
p_functions()
p_funs()
p_funs(pacman)
```

---

p_help *Package Help Manual*

Description

Generate an html, web or pdf of a package's help manual.

Usage

```
p_help(package = NULL, web = TRUE, build.pdf = FALSE)
```

Arguments

package	Name of package.
web	logical. If TRUE grabs current pdf help manual from the web (pdf argument is ignored).
build.pdf	logical. If TRUE attempts to locate the file first and ten uses a LaTeX compiler to generate a pdf.

Warning

Setting build.pdf = TRUE requires the user to have a pdf compiler (e.g., MikTex or Tex Live) installed.

References

http://r.789695.n4.nabble.com/Opening-package-manual-from-within-R-td3763938.html

See Also

help

Examples

```
## Not run:
p_help()
p_help(pacman)
p_help(pacman, web=TRUE)
p_help(pacman, build.pdf=TRUE)

## End(Not run)
```

`p_information` *Package Information*

Description

Provides the information from for a package from the *NAMESPACE*. Information may include: title, version, author, maintainer, description, depends, imports, suggests

Usage

```
p_information(package = "base", ..., fields = NULL)

p_info(package = "base", ..., fields = NULL)
```

Arguments

`package`	Name of the package to grab information for. Default is `"base"`.
`...`	Names of fields (see `fields` argument) to extract.
`fields`	A character vector giving the tags of fields to return (for use inside of functions rather than ...).

Value

Returns a list of fields.

Note

Note that the output from `p_information` (when no fields are passed) prints pretty but is actually an accessible list (use `names(p_info())` test).

See Also

`packageDescription`, `p_information`

Examples

```
p_information()
p_info()
names(p_info())
p_info()[names(p_info())]
p_info(pacman)
p_info(pacman, Author)
p_info(pacman, BugReports, URL)
p_info(pacman, fields = "Version")
## Not run:
p_extract(p_info(ggplot2, "Depends"))
p_extract(p_info(ggplot2, "Imports"))
lapply(p_info(ggplot2, "Imports", "Depends", "Suggests"), p_extract)

## End(Not run)
```

---

p_install *Installs & Loads Packages*

Description

Installs a package provided the package is a CRAN package.

Usage

```
p_install(package, character.only = FALSE, force = TRUE,
  path = getOption("download_path"), try.bioconductor = TRUE,
  update.bioconductor = FALSE, ...)

p_get(package, character.only = FALSE, force = TRUE,
  path = getOption("download_path"), try.bioconductor = TRUE,
  update.bioconductor = FALSE, ...)
```

Arguments

package	Name of package(s).
character.only	logical. If TRUE ... is treated a character string.
force	logical. Should package be installed if it already exists on local system?
path	The path to the directory that contains the package. It is convenient to set download_path in .Rprofile options to the downloads directory.
try.bioconductor	If TRUE, tries to install the package from Bioconductor if it is not found on CRAN using **BiocManager**.
update.bioconductor	If TRUE, tries to update dependencies used by try.bioconductor.
...	Additional parameters to pass to install.packages.

See Also

install.packages

Examples

```
## Not run:
p_install(pacman)

## End(Not run)
```

---

p_install_gh *Installs & Loads GitHub Packages*

Description

Installs a GitHub package. A wrapper for install_github which is the same as install_github.

Usage

```
p_install_gh(package, dependencies = TRUE, ...)
```

Arguments

package	Repository address(es) in the format username/repo[/subdir][@ref|#pull]. Note that this must be a character string.
dependencies	logical. If TRUE necessary dependencies will be installed as well.
...	Additional parameters to pass to install_github.

See Also

install_github

Examples

```
## Not run:
p_install_gh("trinker/pacman")

## Package doesn't exist
p_install_gh("trinker/pacmanAwesomer")

## End(Not run)
```

p_install_version *Install Minimal Package Version*

Description

Install minimal package version(s).

Usage

```
p_install_version(package, version)
```

Arguments

package	character vector of the name of the package(s) you want to install a particular minimal version of.
version	Corresponding character vector of the minimal package version(s).

Examples

```
## Not run:
p_install_version(
    c("pacman", "testthat"),
    c("0.2.0", "0.9.1")
)

## End(Not run)
```

p_install_version_gh *Install Minimal GitHub Package Version*

Description

Install minimal GitHub package version(s).

Usage

```
p_install_version_gh(package, version, dependencies = TRUE)
```

Arguments

package	character vector of the repository address(es) of the package(s) you want to install a particular minimal version of. Repository address(es) in the format username/repo[/subdir][@ref|#pull].
version	Corresponding character vector of the minimal package version(s).
dependencies	logical. If TRUE necessary dependencies will be installed as well.

Examples

```
## Not run:
p_install_version_gh(
    c("trinker/pacman", "hadley/testthat"),
    c("0.2.0", "0.9.1")
)

## End(Not run)
```

`p_interactive` *Interactive Package Exploration*

Description

Interactively search through packages, looking at functions and optionally attaching the package and looking at the help page.

Usage

```
p_interactive()

p_inter()
```

Examples

```
## Not run:
p_interactive()
p_inter()

## End(Not run)
```

`p_isinstalled` *Checks if Package is Installed*

Description

Check if package is installed locally.

Usage

```
p_isinstalled(package)
```

Arguments

`package`	Name of package you want to check. This can be quoted or unquoted.

Examples

```
## Not run:
p_installed(pacman)
p_installed(fakePackage)

## End(Not run)
```

---

p_library *Display Library Packages*

Description

Generates a vector of all packages available to the user and optionally opens the user's library (this isn't necessarily where all of the available packages are stored).

Usage

```
p_library(open = FALSE)

p_lib(open = FALSE)
```

Arguments

open	logical. If TRUE opens the directory of the add on packages library.

Examples

```
p_lib()
p_library()
## Not run:
p_lib(TRUE)

## End(Not run)
```

---

p_load *Load One or More Packages*

Description

This function is a wrapper for library and require. It checks to see if a package is installed, if not it attempts to install the package from CRAN and/or any other repository in the **pacman** repository list.

Usage

```
p_load(..., char, install = TRUE, update = getOption("pac_update"),
  character.only = FALSE)
```

Arguments

char	Character vector containing packages to load. If you are calling p_load from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
install	logical. If TRUE will attempt to install a package not found in the library.
update	logical. If TRUE will attempt to update all out of date packages. Default allows the user to set a "pac_update" in his/her .Rprofile.
character.only	logical. If TRUE then p_load will only accept a single input which is a character vector containing the names of packages to load.
...	name(s) of package(s).

See Also

library, require, install.packages

Examples

```
## Not run:
p_load(lattice)
p_unload(lattice)
p_load(lattice, foreign, boot, rpart)
p_loaded()
p_unload(lattice, foreign, boot, rpart)
p_loaded()

## End(Not run)
```

---

p_load_current_gh *Force Install and Load One or More GitHub Packages*

Description

This function is a wrapper for install_github which is the same as install_github and require. It checks to see if a package is installed, if not it attempts to install the package from GitHub. Use this over p_load_gh if you want to force install the most recent GitHub version of a package.

Usage

```
p_load_current_gh(..., char, update = getOption("pac_update"),
  dependencies = TRUE)
```

Arguments

char	Character vector containing repository address to load. If you are calling p_load_gh from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
update	logical. If TRUE will attempt to update all out of date packages. Default allows the user to set a "pac_update" in his/her .Rprofile.
dependencies	logical. If TRUE necessary dependencies will be installed as well.
...	Repository address(es) in the format username/repo[/subdir][@ref|#pull]. Note that this must be a character string.

See Also

install_github library, require

Examples

```
## Not run:
p_load_current_gh(c("Dasonk/Dmisc", "trinker/clustext", "trinker/termco"))

## End(Not run)
```

---

p_load_gh *Load One or More GitHub Packages*

Description

This function is a wrapper for install_github which is the same as install_github and require. It checks to see if a package is installed, if not it attempts to install the package from GitHub.

Usage

```
p_load_gh(..., char, install = TRUE, update = getOption("pac_update"),
  dependencies = TRUE)
```

Arguments

char	Character vector containing repository address to load. If you are calling p_load_gh from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
install	logical. If TRUE will attempt to install a package not found in the library.
update	logical. If TRUE will attempt to update all out of date packages. Default allows the user to set a "pac_update" in his/her .Rprofile.
dependencies	logical. If TRUE necessary dependencies will be installed as well.
...	Repository address(es) in the format username/repo[/subdir][@ref|#pull]. Note that this must be a character string.

See Also

install_github library, require

Examples

```
## Not run:
p_load_gh("Dasonk/Dmisc", "trinker/regexr")

p_load_gh(c("trinker/regexTools",
    "hadley/lubridate",
    "ramnathv/rCharts"))

## End(Not run)
```

---

p_loaded *Check for Loaded Packages*

Description

p_loaded - Output is a character string of loaded packages.

p_isloaded - Check if package(s) is loaded.

Usage

```
p_loaded(..., all = FALSE, char, character.only = FALSE)

p_isloaded(...)
```

Arguments

all	logical. If TRUE will show all packages including base install; FALSE will show all packages excluding base install packages that install when R loads.
char	Character vector containing packages to load. If you are calling p_loaded from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
character.only	logical. If TRUE then p_loaded will only accept a single input which is a character vector containing the names of packages to load.
...	Optional package names. Adding package names will check their individual load status.

See Also

.packages, sessionInfo

Examples

```
## Not run:
p_load(lattice, ggplot2)

## End(Not run)
p_loaded()
p_loaded(all=TRUE)
p_loaded(ggplot2, tm, qdap)

p_isloaded(ggplot2)
p_isloaded(ggplot2, dfs, pacman)
## Not run: p
_unload(lattice)

## End(Not run)
```

---

p_news *Package/R News*

Description

Find out news on a package or R.

Usage

```
p_news(package = NULL)
```

Arguments

package	Name of package (default is to see news for R).

See Also

news

Examples

```
## Not run:
p_news()
p_news(lattice)
## Grab specific version subsets
subset(p_news(lattice), Version == 0.7)

## End(Not run)
```

p_old *Compare Installed Packages with CRAN-like Repositories*

Description

Indicates packages which have a (suitable) later version on the repositories

Usage

```
p_old()
```

Value

Retuns a `data.frame` with info regarding out of data packages.

See Also

`old.packages`

Examples

```
## Not run:
p_old()

## End(Not run)
```

p_opendir *Attempts to open a directory in a file browser*

Description

Attempts to open a directory in a file browser. Opening a directory isn't a platform independent but it is used in more than one function so moving this functionality to its own non-exported function makes sense.

Usage

```
p_opendir(dir = getwd())
```

Arguments

dir	A character string representing the path (either relative or absolute) to the directory to be opened. Defaults to the working directory.

Note

Most likely this function will move to a different package at some point as it's not specifically package related.

Examples

```
## Not run:
p_opendir() # opens working directory
p_opendir(path.expand("~")) # opens home directory
p_opendir(pacman:::p_basepath())

## End(Not run)
```

---

p_path *Path to Library of Add-On Packages*

Description

Path to library of add-on packages.

Usage

```
p_path(package = "R")
```

Arguments

package	Name of package (default returns path to library of add-on packages).

See Also

.libPaths

Examples

```
p_path()
p_path(pacman)
```

p_search_any *Search CRAN Packages by Maintainer, Author, Version or Package*

Description

Uses agrep to find packages by maintainer (often this is the author as well) or by name.

Usage

```
p_search_any(term, search.by = "Maintainer")

p_sa(term, search.by = "Maintainer")
```

Arguments

term	A search term (character string).
search.by	The variable to search by (takes a integer or a character string): 1-"Maintainer", 1-"Author", 2-"Package", 3-"Version"

Details

Useful for finding packages by the same author (usually the same as the maintainer). This function will take some time as the function is searching thousands of packages via CRAN's website.

Author(s)

BondedDust (stackoverflow.com) and Tyler Rinker <tyler.rinker@gmail.com>

References

https://cran.r-project.org/web/checks/check_summary_by_maintainer.html#summary_by_maintainer http://stackoverflow.com/a/10082624/1000343

Examples

```
## Not run:
p_search_any("hadley", 1)
p_sa("hadley", "author")
p_sa("color", 2)
p_sa("psych", "package")

## End(Not run)
```

p_search_library *Partial Matching Package Search*

Description

Search library packages using partial matching. Search for packages by partial matching letter(s) or by any letter(s) contained within the package's name. Useful for those times when you can't remember that package name but you know "it starts with..."

Usage

```
p_search_library(begins.with = NULL, contains = NULL)

p_sl(begins.with = NULL, contains = NULL)
```

Arguments

begins.with	A character string to search for packages starting with the letter(s).
contains	A character string to search for packages containing the letter(s).

Examples

```
## Not run:
p_search_library(begins.with = "ma")
p_search_library(begins.with = "r", contains = "ar")
p_search_library(contains = "att")

## End(Not run)
```

p_set_cranrepo *Check if Repo is Set*

Description

Check if a repo is already set and if not choose an appropriate repo.

Usage

```
p_set_cranrepo(default_repo = "http://cran.rstudio.com/")
```

Arguments

default_repo	The default package repository.

p_temp *Install a Package Temporarily*

Description

Installs and loads a package for the current session. The package won't be available in future sessions and will eventually be deleted from the machine with no additional effort needed by the user. This will also install the necessary dependencies temporarily as well.

Usage

```
p_temp(package, character.only = FALSE)
```

Arguments

package	The package we want to install temporarily
character.only	logical. Is the input a character string?

Author(s)

juba (stackoverflow.com) and Dason Kurkiewicz

References

http://stackoverflow.com/a/14896943/1003565

p_unload *Unloads package(s)*

Description

Unloads package(s) or all packages.

Usage

```
p_unload(..., negate = FALSE, char, character.only = FALSE)
```

Arguments

...	name of package(s) or "all" (all removes all add on packages).
negate	logical. If TRUE will unload all add on packages except those provided to p_unload.
char	Character vector containing packages to load. If you are calling p_unload from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to load to this parameter directly.
character.only	logical. If TRUE then p_unload will only accept a single input which is a character vector containing the names of packages to load.

Note

p_unload will not unload the base install packages that load when R boots up. See the comments in the help for detach about some issues with unloading and reloading namespaces.

See Also

detach

Examples

```
## Not run:
p_load(lattice)
p_loaded()
p_unload(lattice)
p_loaded()

p_load("lattice", "MASS")
p_loaded()
p_unload(all)
p_loaded() # will not work as you unloaded pacman

library(pacman)
p_load(lattice, MASS, foreign)
p_loaded()
p_unload(pacman, negate=TRUE)
p_loaded()

## End(Not run)
```

---

p_unlock *Delete 00LOCK Directory*

Description

Deletes the 00LOCK directory accidentally left behind by a fail in install.packages.

Usage

```
p_unlock(lib.loc = p_path())
```

Arguments

lib.loc	Path to library location.

Details

Sometimes install.packages can "fail so badly that the lock directory is not removed: this inhibits any further installs to the library directory (or for –pkglock, of the package) until the lock directory is removed manually." p_unlock deletes the directory 00LOCK that is left behind.

Value

Attempts to delete a 00LOCK(s) if it exists. Returns logical TRUE if a 00LOCK existed and FALSE if not.

See Also

install.packages

Examples

```
## Not run:
p_unlock()

## End(Not run)
```

---

p_update *Update Out-of-Date Packages*

Description

Either view out of date packages or update out of data packages.

Usage

```
p_update(update = TRUE, ask = FALSE, ...)

p_up(update = TRUE, ask = FALSE, ...)
```

Arguments

update	logical. If TRUE updates any out-of-date packages; if FALSE returns a list of out-of-date packages.
ask	logical. If TRUE asks user before packages are actually downloaded and installed, or the character string "graphics", which brings up a widget to allow the user to (de-)select from the list of packages which could be updated or added.
...	Other arguments passed to update.packages.

See Also

update.packages, old.packages

Examples

```
## Not run:
p_update()
p_update(FALSE)
p_up(FALSE)

## End(Not run)
```

p_version *Package Version*

Description

p_version - Determine what version a package is in your library.

p_version_cran - Determine what version a package is on CRAN.

p_version_difference - Determine version difference between a local package and CRAN.

Usage

```
p_version(package = "R")

p_ver(package = "R")

p_version_cran(package = "R")

p_ver_cran(package = "R")

p_version_diff(package = "R")

p_ver_diff(package = "R")
```

Arguments

package	Name of package (default returns R version).

See Also

packageDescription

Examples

```
## Not run:
p_ver()
p_version()
p_ver(pacman)
p_version(pacman)

p_ver_cran()
p_ver_cran(pacman)

## Compare local to CRAN version
p_ver(pacman) == p_ver_cran(pacman)
p_ver(pacman) > p_ver_cran(pacman)

p_ver_diff()
p_ver_diff(pacman)
```

```
## End(Not run)
```

---

p_vignette *View Package Vignette(s)*

Description

Interactively view vignettes for package(s) or return a dataframe of vignettes and accompanying information.

Usage

```
p_vignette(..., char, interactive = TRUE, character.only = FALSE)

p_vign(..., char, interactive = TRUE, character.only = FALSE)
```

Arguments

char	Character vector containing packages to find vignettes for. If you are calling p_vignette from within a function (or just having difficulties calling it using a character vector input) then pass your character vector of packages to this parameter directly.
interactive	logical. If TRUE will generate an HTML list of selections.
character.only	logical. If TRUE then p_vignette will only accept a single input which is a character vector containing the names of packages to find vignettes for.
...	name(s) of package(s).

See Also

vignette, browseVignettes

Examples

```
## Not run:
p_vignette(interactive = FALSE)
p_vignette()
p_vign()
p_vign(pacman)
p_vign(grid, utils)
p_vign(grid, utils, interactive = FALSE)
p_vign(fortunes)

## End(Not run)
```

print.p_version_diff *Prints a p_version_diff Object*

Description

Prints a p_version_diff object.

Usage

```
## S3 method for class 'p_version_diff'
print(x, ...)
```

Arguments

x	The p_version_diff object.
...	ignored

print.search_any *Prints a search_any Object*

Description

Prints a search_any object.

Usage

```
## S3 method for class 'search_any'
print(x, ...)
```

Arguments

x	The search_any object.
...	ignored

`print.wide_table` *Prints a wide_table Object*

Description

Prints a wide_table object.

Usage

```
## S3 method for class 'wide_table'
print(x, right = FALSE, ...)
```

Arguments

`x`	The wide_table object.
`right`	logical. If `FALSE` stings will be left-aligned.
`...`	ignored

Index

∗ **CRAN**
p_cran, 5
∗ **author**
p_author, 2
p_search_any, 24
∗ **base**
p_base, 3
∗ **citation**
p_citation, 4
∗ **cite**
p_citation, 4
∗ **data**
p_data, 6
∗ **delete**
p_delete, 7
∗ **dependencies**
p_depends, 8
∗ **dependency**
p_depends, 8
∗ **detach**
p_unload, 26
∗ **exists**
p_exists, 9
∗ **function**
p_functions, 10
∗ **github**
p_install_gh, 14
∗ **header**
p_boot, 4
∗ **help**
p_help, 11
∗ **information**
p_information, 12
∗ **info**
p_information, 12
∗ **installed**
p_isinstalled, 16
∗ **install**
p_install, 13
p_install_gh, 14
∗ **library**
p_library, 17
p_path, 23
p_search_library, 25
∗ **loaded**
p_loaded, 20
∗ **location**
p_path, 23
∗ **lock**
p_unlock, 27
∗ **manual**
p_help, 11
∗ **news**
p_news, 21
∗ **old**
p_old, 22
∗ **outdated**
p_old, 22
∗ **packageDescription**
p_information, 12
∗ **packages**
p_loaded, 20
p_news, 21
p_update, 28
∗ **package**
p_author, 2
p_base, 3
p_cran, 5
p_data, 6
p_delete, 7
p_exists, 9
p_functions, 10
p_help, 11
p_information, 12
p_install, 13
p_interactive, 16
p_isinstalled, 16
p_library, 17

p_path, 23
p_search_any, 24
p_search_library, 25
p_unload, 26
p_version, 29
p_vignette, 30
∗ **path**
p_path, 23
∗ **search**
p_search_any, 24
p_search_library, 25
∗ **update**
p_update, 28
∗ **version**
p_install_version, 15
p_install_version_gh, 15
p_version, 29
∗ **vignette**
p_vignette, 30
.libPaths, *23*
.packages, *20*

agrep, *24*
available.packages, *6*

browseVignettes, *30*

citation, *5*

data, *6*
data.frame, *6*, *22*
dependsOnPkgs, *8*
detach, *27*

getOption, *3*

help, *11*

install.packages, *14*, *18*, *27, 28*
install_github, *14*, *18–20*

library, *17–20*

news, *21*

old.packages, *22*, *28*

p_author, 2
p_base, 3
p_boot, 4
p_citation, 4
p_cite *(p_citation)*, 4
p_cran, 5
p_data, 6
p_del *(p_delete)*, 7
p_delete, 7
p_depends, 8
p_depends_reverse *(p_depends)*, 8
p_detectOS, 9
p_exists, 9
p_extract, 10
p_functions, 10
p_funs *(p_functions)*, 10
p_get *(p_install)*, 13
p_help, 11
p_info, *8*
p_info *(p_information)*, 12
p_information, *10*, 12, *12*
p_install, 13
p_install_gh, 14
p_install_version, 15
p_install_version_gh, 15
p_inter *(p_interactive)*, 16
p_interactive, 16
p_iscran *(p_cran)*, 5
p_isinstalled, 16
p_isloaded *(p_loaded)*, 20
p_lib *(p_library)*, 17
p_library, 17
p_load, 17
p_load_current_gh, 18
p_load_gh, 19
p_loaded, 20
p_news, 21
p_old, 22
p_opendir, 22
p_path, 23
p_sa *(p_search_any)*, 24
p_search_any, 24
p_search_library, 25
p_set_cranrepo, 25
p_sl *(p_search_library)*, 25
p_temp, 26
p_unload, 26
p_unlock, 27
p_up *(p_update)*, 28
p_update, 28
p_ver *(p_version)*, 29
p_ver_cran *(p_version)*, 29

p_ver_diff *(*p_version*)*, 29
p_version, 29
p_version_cran *(*p_version*)*, 29
p_version_diff *(*p_version*)*, 29
p_vign *(*p_vignette*)*, 30
p_vignette, 30
package_dependencies, *8*
packageDescription, *3*, *12*, *29*
print.p_version_diff, 31
print.search_any, 31
print.wide_table, 32

remove.packages, *7*
require, *17–20*

sessionInfo, *20*

update.packages, *28*

vignette, *30*