

# Package ‘pacman’

February 21, 2015

**Type** Package

**Title** Package Management Tool

**Version** 0.3.0

**Date** 2012-06-03

**Author** Tyler Rinker [aut, cre, ctb], Dason Kurkiewicz [aut, ctb]

**Maintainer** Tyler Rinker <tyler.rinker@gmail.com>

**Depends** R (>= 2.13.0)

**Imports** devtools

**Suggests** 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>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-02-21 01:22:28

## R topics documented:

|                            |   |
|----------------------------|---|
| print.search_any . . . . . | 2 |
| print.wide_table . . . . . | 3 |
| p_author . . . . .         | 3 |
| p_base . . . . .           | 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 . . . . . | 14 |
| p_interactive . . . . .     | 15 |
| p_isinstalled . . . . .     | 15 |
| p_library . . . . .         | 16 |
| p_load . . . . .            | 16 |
| p_loaded . . . . .          | 17 |
| p_load_gh . . . . .         | 18 |
| p_news . . . . .            | 19 |
| p_opendir . . . . .         | 20 |
| p_path . . . . .            | 20 |
| p_search_any . . . . .      | 21 |
| p_search_library . . . . .  | 22 |
| p_set_cranrepo . . . . .    | 22 |
| p_temp . . . . .            | 23 |
| p_unload . . . . .          | 23 |
| p_unlock . . . . .          | 24 |
| p_update . . . . .          | 25 |
| p_version . . . . .         | 26 |
| p_vignette . . . . .        | 26 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>28</b> |
|--------------|-----------|

---

|                  |                                   |
|------------------|-----------------------------------|
| print.search_any | <i>Prints a search_any Object</i> |
|------------------|-----------------------------------|

---

## 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 | <i>Prints a wide_table Object</i> |
|------------------|-----------------------------------|

---

**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  |

---

|          |                       |
|----------|-----------------------|
| p_author | <i>Package Author</i> |
|----------|-----------------------|

---

**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\_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 A 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 <a href="#">citation</a>  |

**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)
```

---

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 <b>CRAN</b> for the package.  |
| character.only | logical. If TRUE the input is a variable containing the package name.                              |
| ...            | other arguments passed to <a href="#">package_dependencies</a> and <a href="#">dependsOnPkgs</a> . |

### 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 | <i>Detects Operating System</i> |
|------------|---------------------------------|

---

**Description**

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

**Usage**

```
p_detectOS()
```

---

|          |  |
|----------|--|
| p_exists | <i>Checks if Package is On CRAN/In Local Library</i> |
|----------|--|

---

**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 <b>CRAN</b> 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 then 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".                                       |
| fields  | A character vector giving the tags of fields to return (for use inside of functions rather than ...). |
| ...     | Names of fields (see fields argument) to extract.   |

## 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,
  path = getOption("download_path"), ...)

p_get(package, character.only = FALSE, path = getOption("download_path"),
  ...)

```

**Arguments**

|                |   |
|----------------|---|
| package        | Name of package(s).   |
| character.only | logical. If TRUE ... is treated a character string.   |
| path           | The path to the directory that contains the package. It is convenient to set download_path in .Rprofile options to the downloads directory. |
| ...            | Additional parameters to pass to install.packages.  |

**See Also**

[install.packages](#)

**Examples**

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

---

`p_install_gh`                    *Installs & Loads GitHub Packages*

---

**Description**

Installs a GitHub package. A wrapper for [install\\_github](#).

**Usage**

```
p_install_gh(package, ...)
```

**Arguments**

|                      |   |
|----------------------|---|
| <code>package</code> | Repository address(es) in the format <code>username/repo[/subdir][@ref #pull]</code> . Note that this must be a character string. |
| <code>...</code>     | Additional parameters to pass to <a href="#">install_github</a> .   |

**See Also**

[install\\_github](#)

**Examples**

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

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

## End(Not run)
```

---

`p_install_version`                *Install Minimal Package Version*

---

**Description**

Install minimal package version(s).

**Usage**

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

**Arguments**

|                      |   |
|----------------------|---|
| <code>package</code> | character vector of the name of the package(s) you want to install a particular minimal version of. |
| <code>version</code> | Corresponding character vector of the minimal package version(s).                                   |

**Examples**

```
p_install_version(  
  c("pacman", "testthat"),  
  c("0.2.0", "0.9.1")  
)
```

---

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)
```

---

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\_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)
```

---

p\_load\_gh

*Load One or More GitHub Packages*

---

**Description**

This function is a wrapper for [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"))
```

**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. |
| ...     | Repository address(es) in the format username/repo[/subdir][@ref #pull].<br>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\_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**

```
p_news()
p_news(lattice)
## Grab specific version subsets
subset(p_news(lattice), Version == 0.7)
```

---

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()
```

### See Also

[.libPaths](#)

**Examples**

```
p_path()
```

---

|              |   |
|--------------|---|
| p_search_any | <i>Search CRAN Packages by Maintainer, Author, Version or Package</i> |
|--------------|---|

---

**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**

[http://cran.r-project.org/web/checks/check\\_summary\\_by\\_maintainer.html#summary\\_by\\_maintainer](http://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

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.  
...                name of package(s) or "all" (all removes all add on packages).

**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 OOLOCK(s) if it exists. Returns logical TRUE if a OOLOCK existed and FALSE if not.

**See Also**

[install.packages](#)

**Examples**

```
## Not run:
p_unlock()

## End(Not run)
```

---

|          |                                    |
|----------|------------------------------------|
| p_update | <i>Update Out-of-Date Packages</i> |
|----------|------------------------------------|

---

**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 <a href="#">update.packages</a> .  |

**See Also**

[update.packages](#), [old.packages](#)

**Examples**

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

## End(Not run)
```

---

|           |                        |
|-----------|------------------------|
| p_version | <i>Package Version</i> |
|-----------|------------------------|

---

**Description**

Determine what version a package is in your library.

**Usage**

```
p_version(package = "R")
```

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

---

|            |                                 |
|------------|---------------------------------|
| p_vignette | <i>View Package Vignette(s)</i> |
|------------|---------------------------------|

---

**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**

|                             |   |
|-----------------------------|---|
| <code>char</code>           | Character vector containing packages to find vignettes for. If you are calling <code>p_vignette</code> 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. |
| <code>interactive</code>    | logical. If TRUE will generate an HTML list of selections.  |
| <code>character.only</code> | logical. If TRUE then <code>p_vignette</code> will only accept a single input which is a character vector containing the names of packages to find vignettes for.   |
| <code>...</code>            | 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)
```

# Index

- \*Topic **CRAN**
  - p\_cran, 5
- \*Topic **author**
  - p\_author, 3
  - p\_search\_any, 21
- \*Topic **base**
  - p\_base, 4
- \*Topic **citation**
  - p\_citation, 4
- \*Topic **cite**
  - p\_citation, 4
- \*Topic **data**
  - p\_data, 6
- \*Topic **delete**
  - p\_delete, 7
- \*Topic **dependencies**
  - p\_depends, 8
- \*Topic **dependency**
  - p\_depends, 8
- \*Topic **detach**
  - p\_unload, 23
- \*Topic **exists**
  - p\_exists, 9
- \*Topic **function**
  - p\_functions, 10
- \*Topic **github**
  - p\_install\_gh, 14
- \*Topic **help**
  - p\_help, 11
- \*Topic **information**
  - p\_information, 12
- \*Topic **info**
  - p\_information, 12
- \*Topic **installed**
  - p\_isinstalled, 15
- \*Topic **install**
  - p\_install, 13
  - p\_install\_gh, 14
- \*Topic **library**
  - p\_library, 16
  - p\_path, 20
  - p\_search\_library, 22
- \*Topic **loaded**
  - p\_loaded, 17
- \*Topic **location**
  - p\_path, 20
- \*Topic **lock**
  - p\_unlock, 24
- \*Topic **manual**
  - p\_help, 11
- \*Topic **news**
  - p\_news, 19
- \*Topic **packageDescription**
  - p\_information, 12
- \*Topic **packages**
  - p\_loaded, 17
  - p\_news, 19
  - p\_update, 25
- \*Topic **package**
  - p\_author, 3
  - p\_base, 4
  - 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, 15
  - p\_isinstalled, 15
  - p\_library, 16
  - p\_path, 20
  - p\_search\_any, 21
  - p\_search\_library, 22
  - p\_unload, 23
  - p\_version, 26
  - p\_vignette, 26

- \*Topic **path**
  - p\_path, 20
- \*Topic **search**
  - p\_search\_any, 21
  - p\_search\_library, 22
- \*Topic **update**
  - p\_update, 25
- \*Topic **version**
  - p\_install\_version, 14
  - p\_version, 26
- \*Topic **vignette**
  - p\_vignette, 26
- .libPaths, 20
- .packages, 18
- agrep, 21
- available.packages, 5
- browseVignettes, 27
- citation, 5
- data, 6
- data.frame, 6
- dependsOnPkgs, 8
- detach, 24
- getOption, 4
- help, 11
- install.packages, 13, 17, 24, 25
- install\_github, 14, 18, 19
- library, 16, 17, 19
- news, 19
- old.packages, 25
- p\_author, 3
- p\_base, 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, 14
- p\_inter (p\_interactive), 15
- p\_interactive, 15
- p\_iscran (p\_cran), 5
- p\_isinstalled, 15
- p\_isloaded (p\_loaded), 17
- p\_lib (p\_library), 16
- p\_library, 16
- p\_load, 16
- p\_load\_gh, 18
- p\_loaded, 17
- p\_news, 19
- p\_opendir, 20
- p\_path, 20
- p\_sa (p\_search\_any), 21
- p\_search\_any, 21
- p\_search\_library, 22
- p\_set\_cranrepo, 22
- p\_sl (p\_search\_library), 22
- p\_temp, 23
- p\_unload, 23
- p\_unlock, 24
- p\_up (p\_update), 25
- p\_update, 25
- p\_ver (p\_version), 26
- p\_version, 26
- p\_vign (p\_vignette), 26
- p\_vignette, 26
- package\_dependencies, 8
- packageDescription, 3, 12, 26
- print.search\_any, 2
- print.wide\_table, 3
- remove.packages, 7
- require, 16–19
- sessionInfo, 18

update.packages, [25](#)

vignette, [27](#)