Package 'pak'

November 20, 2020

Version 0.1.2.1

Title Another Approach to Package Installation

Description The goal of 'pak' is to make package installation faster and more reliable. In particular, it performs all HTTP operations in parallel, so metadata resolution and package downloads are fast. Metadata and package files are cached on the local disk as well. 'pak' has a dependency solver, so it finds version conflicts before performing the installation. This version of 'pak' supports CRAN, 'Bioconductor' and 'GitHub' packages as well.

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.1.1

Imports assertthat, base64enc, callr (>= 3.0.0.9002), cli (>= 1.0.0), cliapp (>= 0.0.0.9002), crayon (>= 1.3.4), curl (>= 3.2), desc (>= 1.2.0), filelock (>= 1.0.2), glue (>= 1.3.0), jsonlite, lpSolve, pkgbuild (>= 1.0.2), pkgcache (>= 1.0.3), prettyunits, processx (>= 3.2.1), ps (>= 1.3.0), R6, rematch2, rprojroot (>= 1.3.2), tibble, utils

Depends R (>= 3.2)

Suggests covr, mockery, pingr, testthat, withr

URL https://pak.r-lib.org/

BugReports https://github.com/r-lib/pak/issues

NeedsCompilation no

Author Jim Hester [aut],
Gábor Csárdi [aut, cre],
RStudio [cph]

Maintainer Gábor Csárdi <csardi.gabor@gmail.com>

Repository CRAN

Date/Publication 2020-11-19 23:10:02 UTC

R topics documented:

lib_status 2
local_install 3
local_install_deps 3
local_install_dev_deps 4
local_package_trees 5
pak_cleanup 6
pak_package_sources 6
pak_private_library 9
pak_setup 9
pak_sitrep 10
pkg_install 11
pkg_remove 12
pkg_status 12

Index **13**

`lib_status` *Status of packages in a library*

Description

Status of packages in a library

Usage

```
lib_status(lib = .libPaths()[1])
```

Arguments

`lib`	Path to library.

Value

Data frame (tibble) the contains data about the packages installed in the library.

local_install *Install a package tree*

Description

Installs a package tree (or source package file), together with its dependencies.

Usage

```
local_install(root = ".", lib = .libPaths()[1], upgrade = FALSE,
  ask = interactive())
```

Arguments

root	Path to the package tree.
lib	Package library to install the packages to. Note that *all* dependent packages will the be installed here, even if they are already installed in another library.
upgrade	Whether to upgrade already installed packages to the latest available version. If this is FALSE, then only packages that need updates to satisfy version requirements, will be updated. If it is TRUE, all specified or dependent packages will be updated to the latest available version.
ask	Whether to ask for confirmation.

Details

local_install() is equivalent to pkg_install("local::.").

Value

Data frame, with information about the installed package(s).

See Also

Other local package trees: local_install_deps, local_install_dev_deps, local_package_trees

local_install_deps *Install the dependencies of a package tree*

Description

Installs the hard dependencies of a package tree (or source package file), without installing the package tree itself.

Usage

```
local_install_deps(root = ".", lib = .libPaths()[1], upgrade = FALSE,
  ask = interactive())
```

Arguments

root	Path to the package tree.
lib	Package library to install the packages to. Note that *all* dependent packages will the be installed here, even if they are already installed in another library.
upgrade	Whether to upgrade already installed packages to the latest available version. If this is FALSE, then only packages that need updates to satisfy version requirements, will be updated. If it is TRUE, all specified or dependent packages will be updated to the latest available version.
ask	Whether to ask for confirmation.

Details

Note that development (and optional) dependencies, under Suggests in DESCRIPTION, are not installed. If you want to install them as well, use local_install_dev_deps().

Value

Data frame, with information about the installed package(s).

See Also

Other local package trees: local_install_dev_deps, local_install, local_package_trees

---

local_install_dev_deps

Install all dependencies of a package tree

Description

Installs all dependencies of a package tree (or source package file), without installing the package tree itself. It installs the development dependencies as well, specified in the Suggests field of DESCRIPTION.

Usage

```
local_install_dev_deps(root = ".", lib = .libPaths()[1],
  upgrade = FALSE, ask = interactive())
```

Arguments

root	Path to the package tree.
lib	Package library to install the packages to. Note that *all* dependent packages will the be installed here, even if they are already installed in another library.
upgrade	Whether to upgrade already installed packages to the latest available version. If this is FALSE, then only packages that need updates to satisfy version requirements, will be updated. If it is TRUE, all specified or dependent packages will be updated to the latest available version.
ask	Whether to ask for confirmation.

See Also

Other local package trees: local_install_deps, local_install, local_package_trees

---

local_package_trees *Local package trees*

Description

pak can install packages from local package trees. This is convenient for package development. See the following functions:

- local_install() installs a package from a package tree and all of its (hard) dependencies (i.e. Includes, Depends, LinkingTo.
- local_install_deps() installs all hard dependencies of a package.
- local_install_dev_deps() installs all hard and soft dependencies of a package. This function is intended for active package development.

Details

Note that the last two functions do not install the package in the specified package tree itself, only its dependencies. This is convenient if the package itself is loaded via some other means, e.g. devtools::load_all(), for development.

See Also

Other local package trees: local_install_deps, local_install_dev_deps, local_install

pak_cleanup *Clean up pak caches and/or the pak library*

Description

Clean up pak caches and/or the pak library

Usage

```
pak_cleanup(package_cache = TRUE, metadata_cache = TRUE,
  pak_lib = TRUE, force = FALSE)
```

Arguments

package_cache	Whether to clean up the cache of package files.
metadata_cache	Whether to clean up the cache of package meta data.
pak_lib	Whether to clean up the pak package library.
force	Do not ask for confirmation. Note that to use this function in non-interactive mode, you have to specify force = FALSE.

See Also

Other pak housekeeping: pak_private_library, pak_setup, pak_sitrep

pak_package_sources *Package sources*

Description

Package sources

Standard packages

pak can install packages from various package sources. By default, a package name without the specification of its source, refers to a CRAN or Bioconductor package. pak calls these *standard* packages. For example:

```
## CRAN package
pkg_install("glue")
## BioC package
pkg_install("limma")
```

When considering a standard package, the calling version of R is used to determine the available source and binary packages on CRAN and the Bioconductor repositories.

The full specification of standard packages is simply

```
[standard::]<package>
```

If you know the exact source of the package, you can also write

```
cran::<package>
bioc::<package>
```

GitHub packages

pak can install packages from GitHub repositories. Any package that is specified in the `user/repo` notation is taken to be a GitHub package. For example:

```
## Package from GitHub
pkg_install("r-lib/glue")
```

The full specification of GitHub packages is

```
[<package>=][github::]<username>/<repo>[/<subdir>]
    [@<committish> | #<pull> | @[*]release]
```

- `<package>` is the name of the package. If this is missing, the name of the package must match the name of the repository.
- `<username>`: GitHub user or organization name.
- `<repo>`: repository name.
- `<subdir>`: If the R package is in a subdirectory within the repository.
- `<commitish>`: A branch name, git tag or SHA hash, to specify the branch, tag or commit to download or install.
- `<pull>`: Pull request number, to install the branch that corresponds to a pull request.
- The `@*release` string can be used to install the latest release.

Local package trees

pak can install packages from package trees. You can either use the `local_install()` function for this, or specify the `local::` package source. E.g. these are equivalent:

```
local_install("/path/to/my/package")
pkg_install("local::/path/to/my/package")
```

The `local::` form is handy if you want to mix it with other package specifications, e.g. to install a local package, and another standard package:

```
pkg_install(c("local://path/to/my/package", "testthat"))
```

The `Remotes` field

You can mark any regular dependency defined in the `Depends`, `Imports`, `Suggests` or `Enhances` fields as being installed from a remote location by adding the remote location to `Remotes` in your `DESCRIPTION` file. This will cause pak to download and install them from the specified location, instead of CRAN.

The remote dependencies specified in `Remotes` is a comma separated list of package sources:

```
Remotes: <pkg-source-1>, <pkg-source-2>, [ ... ]
```

Note that you will still need add the package to one of the regular dependency fields, i.e. `Imports`, `Suggests`, etc. Here is a concrete example that specifies the `r-lib/glue` package:

```
Imports: glue
Remotes: `r-lib/glue,
  r-lib/httr@v0.4,
  klutometis/roxygen#142,
  r-lib/testthat@c67018fa4970
```

The CRAN and Bioconductor repositories do not support the `Remotes` field, so you need to remove this field, before submitting your package to either of them.

The package dependency solver

pak contains a package dependency solver, that makes sure that the package source and version requirements of all packages are satisfied, before starting an installation. For CRAN and BioC packages this is usually automatic, because these repositories are generally in a consistent state. If packages depend on other other package sources, however, this is not the case.

Here is an example of a conflict detected:

```
> pak::pkg_install(c("r-lib/pkgcache@conflict", "r-lib/cli@message"))
Error: Cannot install packages:
  * Cannot install `r-lib/pkgcache@conflict`.
    - Cannot install dependency r-lib/cli@master
  * Cannot install `r-lib/cli@master`.
- Conflicts r-lib/cli@message
```

`r-lib/pkgcache@conflict` depends on the master branch of `r-lib/cli`, whereas, we explicitly requested the `message` branch. Since it cannot install both versions into a single library, pak quits.

When pak considers a package for installation, and the package is given with its name only, (e.g. as a dependency of another package), then the package may have *any* package source. This is necessary, because one R package library may contain only at most one version of a package with a given name.

pak's behavior is best explained via an example. Assume that you are installing a local package (see below), e.g. `local::.`, and the local package depends on `pkgA` and `user/pkgB`, the latter being a package from GitHub (see below), and that `pkgA` also depends on `pkgB`. Now pak must install `pkgB` *and* `user/pkgB`. In this case pak interprets `pkgB` as a package from any package source, instead of a standard package, so installing `user/pkgB` satisfies both requirements.

Note that that `cran::pkgB` and `user/pkgB` requirements result a conflict that pak cannot resolve. This is because the first one *must* be a CRAN package, and the second one *must* be a GitHub package, and two different packages with the same cannot be installed into an R package library.

See Also

Other package functions: `pkg_install`, `pkg_remove`, `pkg_status`

`pak_private_library` *The pak private library*

Description

pak is an R package, and needs other R packages to do its job. These dependencies should be kept separate from the user's "regular" package libraries, to avoid the situation when pak needs a different version of a package than the one in the regular library.

Details

To accomplish this, pak keeps all of its dependencies in a separate library. This library is usually in the user's cache directory.

pak creates and updates its private library, as needed: every time pak cannot load a package from the private library, including the obvious case when the user does not have a private library, pak will create one.

You can use `pak_sitrep()` to list the location of the pak private library, and `pak_cleanup()` to clean it up.

See Also

Other pak housekeeping: `pak_cleanup`, `pak_setup`, `pak_sitrep`

`pak_setup` *Install pak's dependencies into its private library*

Description

To avoid interference between your regular R packages and pak's dependencies, pak works off a private library, which can be created by `pak_setup()`.

Usage

```
pak_setup(mode = c("auto", "download", "copy"), quiet = FALSE)
```

Arguments

mode	Where to get the packages from. "download" will try to download them from CRAN. "copy" will try to copy them from your current "regular" package library. "auto" will try to copy first, and if that fails, then it tries to download.
quiet	Whether to omit messages.

Value

The path to the private library, invisibly.

See Also

Other pak housekeeping: pak_cleanup, pak_private_library, pak_sitrep

---

pak_sitrep *pak SITuation REPort*

Description

It prints

- pak version,
- the current library path,
- location of the private library,
- whether the pak private library exists,
- whether the pak private library is functional.

Usage

```
pak_sitrep()
```

See Also

Other pak housekeeping: pak_cleanup, pak_private_library, pak_setup

pkg_install *Install a package*

Description

Install a package and its dependencies, into a single package library.

Usage

```
pkg_install(pkg, lib = .libPaths()[[1L]], upgrade = FALSE,
  ask = interactive())
```

Arguments

pkg	Package names or remote package specifications to install.
lib	Package library to install the packages to. Note that *all* dependent packages will the be installed here, even if they are already installed in another library.
upgrade	Whether to upgrade already installed packages to the latest available version. If this is FALSE, then only packages that need updates to satisfy version requirements, will be updated. If it is TRUE, all specified or dependent packages will be updated to the latest available version.
ask	Whether to ask for confirmation.

Value

Data frame, with information about the installed package(s).

See Also

Other package functions: pak_package_sources, pkg_remove, pkg_status

Examples

```
## Not run:
pkg_install("dplyr")
pkg_install("dplyr", upgrade = TRUE)

## Package from GitHub
pkg_install("r-lib/pkgconfig")

## End(Not run)
```

pkg_remove *Remove installed packages*

Description

Remove installed packages

Usage

```
pkg_remove(pkg, lib = .libPaths()[[1L]])
```

Arguments

pkg	A character vector of packages to remove.
lib	library to remove packages from

See Also

Other package functions: pak_package_sources, pkg_install, pkg_status

pkg_status *Display installed locations of a package*

Description

Display installed locations of a package

Usage

```
pkg_status(pkg, lib = .libPaths())
```

Arguments

pkg	Name of an installed package to display status for.
lib	One or more library paths to lookup package status in.

See Also

Other package functions: pak_package_sources, pkg_install, pkg_remove

Examples

```
## Not run:
pkg_status("MASS")

## End(Not run)
```

Index

∗ **library functions**
    lib_status, 2
∗ **local package trees**
    local_install, 3
    local_install_deps, 3
    local_install_dev_deps, 4
    local_package_trees, 5
∗ **package functions**
    pak_package_sources, 6
    pkg_install, 11
    pkg_remove, 12
    pkg_status, 12
∗ **pak housekeeping**
    pak_cleanup, 6
    pak_private_library, 9
    pak_setup, 9
    pak_sitrep, 10

lib_status, 2
local_install, 3, *4, 5*
local_install(), *5*, *7*
local_install_deps, *3*, 3, *5*
local_install_deps(), *5*
local_install_dev_deps, *3, 4*, 4, *5*
local_install_dev_deps(), *4, 5*
local_package_trees, *3–5*, 5

pak_cleanup, 6, *9, 10*
pak_cleanup(), *9*
pak_package_sources, 6, *11, 12*
pak_private_library, *6*, 9, *10*
pak_setup, *6*, *9*, 9, *10*
pak_sitrep, *6*, *9, 10*, 10
pak_sitrep(), *9*
pkg_install, *9*, 11, *12*
pkg_remove, *9*, *11, 12*, 12
pkg_status, *9*, *11, 12*, 12