Package 'groundhog'

September 7, 2021

Title The Simplest Solution to Version-Control for CRAN Packages

Version 1.5.0

Description Make R scripts that rely on packages reproducible, by ensuring that every time a given script is run, the same version of the used packages are loaded (instead of whichever version the user running the script happens to have installed). This is achieved by using the new command groundhog.library() instead of the base command library(), and including a date in the call. The date is used to call on the same version of the package every time (the most recent version available on CRAN at that date).

URL https://groundhogr.com/, https://github.com/CredibilityLab/groundhog

BugReports https://github.com/CredibilityLab/groundhog/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author Uri Simonsohn [aut, cre] (<https://orcid.org/0000-0002-8601-7211>), Hugo Gruson [ctb, aut] (<https://orcid.org/0000-0002-4094-1476>)

Maintainer Uri Simonsohn <urisohn@gmail.com>

Repository CRAN

Date/Publication 2021-09-07 15:50:02 UTC

R topics documented:

cross.toc 2
get.groundhog.folder 3
get.snowball 3
groundhog.library 4
meta.groundhog 6
reinstall.conflicts 6
see.unistalled.conflicts 7

set.groundhog.folder 8
toc 9

Index **10**

`cross.toc` *Show toc table with multiple packages*

Description

Show toc table with multiple packages

Usage

```
cross.toc(pkgs, date1 = "1970-1-1", date2 = Sys.Date())
```

Arguments

`pkgs`	character vector containing the package names.
`date1, date2`	date range to consider (in the format "%Y-%m-%d").

Value

A `data.frame` with 3 columns:

Version The package version number

Published The date at which the specific version was published

Package The package name

See Also

`toc()` for the same function for a single package.

Examples

```
## Not run:
cross.toc(c("magrittr", "R"))
cross.toc(c("magrittr", "rlang"), date1 = "2012-02-01", date2 = "2020-02-01")

## End(Not run)
```

get.groundhog.folder *Get current local path to groundhog folder*

Description

Get current local path to groundhog folder

Usage

```
get.groundhog.folder()
```

Value

the path to the groundhog folder, the meta-library where `groundhog.library()` downloads and stores packages that can be loaded

Note

you can change the location of this folder with the command `set.groundhog.folder("path")`.

See Also

`set.groundhog.folder()`

Examples

```
## Not run:
get.groundhog.folder()

## End(Not run)
```

get.snowball *Generates dataframe with all dependencies needed to install a package, in the order they will be loaded*

Description

Generates dataframe with all dependencies needed to install a package, in the order they will be loaded

Usage

```
get.snowball(pkg, date, include.suggests = FALSE, force.source = FALSE)
```

Arguments

pkg	character string, name of target package to load (and install if needed),
date	character string (yyyy-mm-dd), or date value, with the date which determines the version of the package, and all dependencies, to be loaded (and installed if needed).
include.suggests	logical, defaults to FALSE. When set to TRUE, includes dependencies classified in the DESCRIPTION file as suggested.
force.source	logical (defaults to FALSE). When set to TRUE, will not attempt installing binary from CRAN or MRAN and instead download source file and install it.

Value

a dataframe with all packages that need to be installed, their version , whether they are installed, where to obtain them if not locally available (CRAN vs MRAN), which date to use for MRAN, installation time from source (in seconds), and local path for storage

Examples

```
## Not run:
get.snowball("rio", "2020-07-12")

## End(Not run)
```

---

groundhog.library *Load CRAN packages, and their dependencies, as current on given date*

Description

Groundhog maintains a separate local package library where it stores version-controlled packages, with multiple versions of the same package saved side-by-side. The date argument in the groundhog.library() function determines the version of the package that is loaded (the most recently published version on CRAN on that date). If that version of the package is not available in the local groundhog library, it is automatically installed. groundhog.library() thus substitutes both library() and install.packages(). No changes to how R manages packages are made (e.g., no change to .libPaths(), to .Rprofile, or to R Studio global settings). Therefore, to discontinue relying on groundhog for package management, all you do is go back to executing the install.packages() and library() functions, instead of the groundhog.library() function.

Usage

```
groundhog.library(
  pkg,
  date,
  quiet.install = TRUE,
```

```
  include.suggests = FALSE,
  ignore.deps = c(),
  force.source = FALSE,
  force.install = FALSE,
  tolerate.R.version = ""
)
```

Arguments

pkg	character string or vector with name of target package(s). Single package names need not be in quotes.
date	character string (yyyy-mm-dd), or date value, with the date which determines the version of the package, and all dependencies, to be loaded (and installed if needed). The most recent date accepted is 2 days prior to when the code is executed.
quiet.install	logical, defaults to TRUE. When set to FALSE, displays output generated by `install.packages()` when installing from source
include.suggests	logical, defaults to FALSE. When set to TRUE, installs/ loads dependencies classified in the DESCRIPTION file as suggested.
ignore.deps	an optional character vector containing dependencies which may be already loaded in the R session and even if the loaded version does not match the version implied by the entered date, groundhog.library() will proceed and ignore this conflict. If one version of a package is loaded, and a different is needed for groundhog, the default behavior is to stop the request and ask the user to restart the R session to unload all packages. This will bypass that requirement.
force.source	logical (defaults to FALSE). When set to TRUE, will not attempt installing binary from CRAN or MRAN and instead download source file and install it.
force.install	logical (defaults to FALSE). When set to TRUE, will delete existing package files in groundhog folder, and install anew.
tolerate.R.version	optional character string containing an R version which `groundhog.library()` will not throw an error for using, even if the date entered corresponds to a more recent major R release.

Details

For more information about groundhog check out groundhogr.com

Value

a character vector containing all active packages for the session, with their version number, under the format pkg_vrs.

Examples

```
## Not run:
groundhog.library("magrittr", "2020-07-12")
```

```
pkgs <- c('pwr','metafor')
groundhog.library(pkgs, "2020-02-12")

#Allow using R 3.6.3 despite entering a date that corresponds to R 4.0
groundhog.library('rio', '2021-04-12', tolerate.R.version='3.6.3')


## End(Not run)
```

`meta.groundhog` *Load a specific version of groundhog, as available on a given date*

Description

Load a specific version of groundhog, as available on a given date

Usage

```
meta.groundhog(date)
```

Arguments

`date`	character string (yyyy-mm-dd), or date value, with the date which determines the version of groundhog to load

Examples

```
## Not run:
#Load groundhog as available on 2021-03-12 (v1.3.2)
meta.groundhog("2021-03-12")

## End(Not run)
```

`reinstall.conflicts` *Re-install packages that created a conflict*

Description

Re-install uninstalled packages (from the non-groundhog local library) previously uninstalled to prevent conflict with installation via groundhog.library()

Usage

```
reinstall.conflicts(since = "1970-01-01")
```

Arguments

since	(optional) character string (yyyy-mm-dd), or date value, to specify that only packages uninstalled since that date will be reinstalled. If not set then all packages ever uninstalled this way will be re-installed.

Details

When installing packages with groundhog.library(), if another version of a needed package or dependency is already loaded the process ends. This is usually solved by restarting the R session thus unloading the conflicting package. But on occasion the problem persists, often because R Studio loads the undesired version of the package automatically from the local (non-groundhog) library. When this occurs, groundhog.library() will prompt users with a set of options, one of which is to uninstall these packages from the non-groundhog library. The set of packages uninstalled in this way are recorded and saved to a data.frame, allowing easily undoing this action, even months later. Specifically, a user running the function `reinstalled.conflicts()` will loop over that data.frame, reinstalling all packages listed if they are not currently available (into the original path they were previously installed). Again, these are not packages in the groundhog library, but rather, in the default library R uses to install packages. IF the optiona parameter `since` is used, only packages uninstalled since that date will be reinstalled.

Note

Because these packages are not managed by groundhog, there is no version control. The package that is re-installed may be a newer version than that previously available (or the installation may fail if the package becomes archived).

Examples

```
## Not run:
#Re-install all packages ever uninstalled due to conflict
reinstall.conflicts()
#Re-install all packages uninstalled since 2021-02-04 due to conflict
reinstall.conflicts("2021-02-04")

## End(Not run)
```

`see.unistalled.conflicts` *View uninstalled conflicting packages by groundhog*

Description

List all packages that have been uninstalled with groundhog, by user, after creating a conflict with package attempted to be loaded with groundhog.library(). These packages can be reinstalled into the local non-groundhog library with `reinstall.conflicts()`

Usage

```
see.unistalled.conflicts()
```

---

set.groundhog.folder *Set groundhog folder location*

Description

Set groundhog folder location

Usage

```
set.groundhog.folder(path)
```

Arguments

path	Character. The path to the groundhog folder containing the library where packages are downloaded and installed.

Value

(invisibly) TRUE upon success.

See Also

get.groundhog.folder()

Examples

```
## Not run:
set.groundhog.folder("~/.R_groundhog")

## End(Not run)
```

toc *Show CRAN publication dates for all versions of a given package*

Description

Show CRAN publication dates for all versions of a given package

Usage

```
toc(pkg, dependencies = FALSE)
```

Arguments

pkg	(required) package name
dependencies	logical (defaults to FALSE). Should the output contain package dependencies (Imports, Depends and Suggests) for pkg.

Value

a data.frame where each row corresponds to one version of pkg, a date column contains the publication date, and when dependecies=TRUE, columns show package dependencies over time as well.

Examples

```
## Not run:
toc("R")
toc("magrittr")
toc("rio",dependencies = TRUE)

## End(Not run)
```

Index

cross.toc, 2

get.groundhog.folder, 3
get.groundhog.folder(), *8*
get.snowball, 3
groundhog.library, 4
groundhog.library(), *3*

meta.groundhog, 6

reinstall.conflicts, 6

see.unistalled.conflicts, 7
set.groundhog.folder, 8
set.groundhog.folder(), *3*

toc, 9
toc(), *2*