Package: natmanager (via r-universe)

August 24, 2024

Title Install the 'Natverse' Packages from Scratch

Version 0.5.1

URL https://github.com/natverse/natmanager,
http://natverse.org/natmanager/

BugReports https://github.com/natverse/natmanager/issues

Description Provides streamlined installation for packages from the
'natverse', a suite of R packages for computational
neuroanatomy built on top of the 'nat' 'NeuroAnatomy Toolbox'
package. Installation of the complete 'natverse' suite requires
a 'GitHub' user account and personal access token 'GITHUB_PAT'.
'natmanager' will help the end user set this up if necessary.

License GPL-3

Imports gh (>= 1.2.1), utils, remotes, pak, usethis (>= 2.0.0), curl,
withr

Suggests testthat (>= 2.1.0), knitr, mockery, rstudioapi, covr,
spelling, rappdirs

Encoding UTF-8

Language en-GB

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/natverse/natmanager

RemoteRef HEAD

RemoteSha e283247acebbda41fcba233a1c336b292c3f2d80

Contents

install — *Install natverse packages from GitHub*

Description

`install` allows you to install one of two collections of nat packages

- `core` a minimal install that can help users to get started with nat and already solve many problems (the default)
- `natverse` a powerful "batteries included" distribution with all mature packages in the natverse.

Since the `natverse` option will install many packages from GitHub, you need to have a GitHub account and personal access token (GITHUB_PAT). Install will check to see if you have a `GITHUB_PAT` already and, if not, walk you through the steps of setting one up. A fall-back PAT is built into the package but we strongly recommend that you sign up to GitHub and get your own if you start using the natverse regularly.

`check_pat` can be used to check if you have a GITHUB_PAT set and will advise on how to do this if necessary.

Usage

```
install(
  collection = c("core", "natverse"),
  pkgs = NULL,
  dependencies = TRUE,
  upgrade.dependencies = FALSE,
  method = c("pak", "remotes"),
  ...
)

check_pat(create = TRUE)
```

Arguments

`collection`	The collection of natverse packages that you would like to install. The current options are `core`, the default, or `natverse`. See Description for more information.
`pkgs`	A character vector of package names specifying natverse packages to install. When present overrides the `collection` argument.
`dependencies`	Which dependencies you want to install. The default value (`TRUE`) will install all dependencies, `NA` will install only hard (essential) dependencies, while `F` will not install any dependencies (not recommended). See `pak::pkg_install` or `install_github` for further details.

`upgrade.dependencies`	Whether to upgrade dependencies of requested packages See the `upgrade` argument of `pak::pkg_install` or `install_github` for details. The default value (`FALSE`) will do the minimum amount to enable you to install the package(s) you have requested. In contrast `TRUE` will go ahead and upgrade all dependencies to the latest version; pak will potentially install source packages to do this.
`method`	Whether to use the `pak` (now the default) or `install_github` package for installation.
`...`	extra arguments to pass to `pak::pkg_install` or `remotes::install_github`.
`create`	Whether to help you create a personal GITHUB_PAT if you do not have one set. When `create=FALSE` a default PAT will be used if you have not set your own. This could cause trouble if other people are using the same PAT.

Value

`check_pat` returns the PAT invisibly or errors out if `create=TRUE` and none can be set.

Examples

```
## Not run:
# install core packages to try out the core natverse
if(is.interactive()) {
natmanager::install('core')
}
# Full "batteries included" installation with all packages
if(is.interactive()) {
natmanager::install('natverse')
# same but upgrading all dependencies to latest version
natmanager::install('natverse', upgrade.dependencies = T)
}
# Install natverse, non-natverse package
# for natverse packages no need to specify the repo
if(is.interactive()) {
natmanager::install(pkgs=c('nat.jrcbrains','flyconnectome/hemibrainr'))
}

## End(Not run)
## Not run:
# Check status of GitHub PAT and create one if required
natmanager::check_pat(create=TRUE)
# Check status of GitHub PAT and use default if no personal one available
natmanager::check_pat(create=FALSE)

## End(Not run)
```

`list_repo` *List all the repos inside a particular GitHub organisation*

Description

by default this will list all the repositories inside the ‘natverse‘ organization.

Usage

```
list_repo(orgname = "natverse")
```

Arguments

`orgname`	Name of the GitHub organization

Value

Character vector of repository names

Examples

```
## Not run:
natmanager::list_repo()

## End(Not run)
```

`selfupdate` *Update the ‘natmanager‘ package itself.*

Description

Update the ‘natmanager‘ package itself.

Usage

```
selfupdate(
  source = c("GITHUB", "CRAN"),
  upgrade.dependencies = TRUE,
  force = FALSE,
  method = c("pak", "remotes"),
  ...
)
```

Arguments

`source`	Location from which to obtain a newer version of natmanager. Defaults to GITHUB since this may well have a newer version than the CRAN package repository.
`upgrade.dependencies`	Whether to upgrade dependencies of requested packages See the `upgrade` argument of `pak::pkg_install` or `install_github` for details. The default value (`FALSE`) will do the minimum amount to enable you to install the package(s) you have requested. In contrast `TRUE` will go ahead and upgrade all dependencies to the latest version; pak will potentially install source packages to do this.
`force`	Force self update even if there doesn't seem to be an update (default `FALSE`)
`method`	Whether to use the `pak` (now the default) or `install_github` package for installation.
`...`	extra arguments to pass to `pak::pkg_install` or `remotes::install_github`.

Value

Logical indicating whether an update was required (invisibly).

See Also

`install`

Examples

```
## Not run:
natmanager::selfupdate()

## End(Not run)
```

Index