Package: kwb.package (via r-universe)

September 2, 2024

Title Install / Unstall KWB Packages and Show Package Dependencies

Version 0.4.0

Description This package contains some helper functions for (un-)installing KWB packages and for showing package dependencies. The function of main interest may be updateKwbPackages() that checks for the latest package version on KWB's server and installs the packages from there if required.

License MIT + file LICENSE

URL https://github.com/kwb-r/kwb.package

BugReports https://github.com/kwb-r/kwb.package/issues

Imports gh, kwb.utils (>= 0.5.0), magrittr, mvbutils, networkD3, remotes, withr

Suggests covr, knitr, pkgmeta, rmarkdown, testthat (>= 3.0.0)

Remotes github::kwb-r/kwb.utils, github::kwb-r/pkgmeta

Encoding UTF-8

RoxygenNote 7.3.1

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/KWB-R/kwb.package

RemoteRef HEAD

RemoteSha e55bce9d65131d24b2967b7f58e3a0e7d7a3e3ba

Contents

addNodeLabels *Add Labels to the Nodes Drawn on a Circle Line*

Description

Add Labels to the Nodes Drawn on a Circle Line

Usage

```
addNodeLabels(nodes, cex = 1, distance.factor = 1)
```

Arguments

nodes	data frame as returned by toNodes
cex	character expansion factor as given to text
distance.factor	expansion factor applied to the x and y coordinates of the nodes to get the coordinates of the labels

anglesToPoints *Angle in Unit Circle to Point Coordinate*

Description

Convert angles in a unit circle to point coordinates (x, y)

Usage

```
anglesToPoints(angles.grad)
```

Arguments

angles.grad	vector of angles in degree

Value

matrix with columns *x* and *y* containing the coordinates of points corresponding to the given angles in a unit circle

Examples

```
plot(anglesToPoints(equidistantAngles(n = 10)), type = "b")
```

archivedCranVersions *Archived CRAN versions*

Description

Archived CRAN versions

Usage

```
archivedCranVersions(package, ref_date = NULL)
```

Arguments

package	package name
ref_date	default: NULL

Examples

```
packages <- c("ggplot2", "swmmr", "kwb.hantush")
archivedCranVersions(packages)
archivedCranVersions(packages, ref_date= "2012-12-01")
```

compareInstalledVersions *Compare Package Versions Between Libraries*

Description

Compare Package Versions Between Libraries

Usage

```
compareInstalledVersions(lib1, lib2)
```

Arguments

lib1	path to first R library
lib2	path to second R library

Value

data frame with columns `name` (package name), `version.1`, `version.2` (version string of package in `lib1` and `lib2`, respectively), `differs` (logical indicating whether the version is different in the two libraries)

copyBasePackages *Copy Base R Packages from the System Library to the Target Library*

Description

Copy Base R Packages from the System Library to the Target Library

Usage

```
copyBasePackages(
  target_lib,
  set_number = 2L,
  system_lib = utils::tail(.libPaths(), 1L),
  packages = systemPackages(set_number)
)
```

Arguments

target_lib	path to the target library
set_number	number defining the base packages to be copied, see systemPackages
system_lib	path to the system library from which to copy packages
packages	vector of names of packages to be copied

cranVersions *Get versions of CRAN packages*

Description

Get versions of CRAN packages

Usage

```
cranVersions(name, dbg = TRUE)
```

Arguments

name	package name
dbg	logical indicating whether or not to show debug messages. Default: TRUE

defaultPackageDir *Default Package Directory*

Description

Default Package Directory

Usage

```
defaultPackageDir()
```

detachAllNonSystemPackages *Detach all Non-System Packages*

Description

Detach all Non-System Packages

Usage

```
detachAllNonSystemPackages()
```

detachRecursively *Detach Packages Recursively*

Description

Detach a package and all the depending packages

Usage

```
detachRecursively(package, pattern = ".*", dbg = FALSE)
```

Arguments

package	name of package to be detached
pattern	pattern matching the names of depending packages that are actually to be detached, e.g. use pattern = "^kwb\." to only detach kwb packages. Default: ".*" (matching all package names)
dbg	if TRUE, debug messages are shown

downloadGitHubPackage *Download an R Package from GitHub*

Description

Download an R Package from GitHub

Usage

```
downloadGitHubPackage(repo, destdir = "~/../Downloads")
```

Arguments

repo	path to repository, relative to https://github,com, e.g. "kwb-r/kwb.utils"
destdir	path to download folder, default: "~/../Downloads"

Value

path to downloaded file in the `destdir` folder with attribute "origin" pointing to the original file in `tempdir()`.

downloadPackagesFromSnapshot *Download Package Archive from Microsoft R Archive Network*

Description

Download Package Archive from Microsoft R Archive Network

Usage

```
downloadPackagesFromSnapshot(
  packages,
  snapshot_date,
  destdir = NULL,
  type = c("source", "win.binary")[1L]
)
```

Arguments

packages	names of packages (vector of character)
snapshot_date	date of snapshot of CRAN package versions, as a string in yyyy-mm-dd format
destdir	path to download folder
type	one of `c("source", "win.binary")`

Value

paths to the downloaded files (vector of character)

`drawDependencies` *Draw Lines between Nodes*

Description

Draw Lines between Nodes

Usage

```
drawDependencies(nodes, dependencies, nodeColours, ...)
```

Arguments

`nodes`	data frame as returned by `toNodes`
`dependencies`	list of package dependencies as returned by `packageDependencies`
`nodeColours`	colours given to the lines starting at the same start node
`...`	arguments passed to `arrows`

`equidistantAngles` *Equidistantly Distributed Angles in Degrees Between 0 and 360*

Description

Equidistantly Distributed Angles in Degrees Between 0 and 360

Usage

```
equidistantAngles(n, from = 0)
```

Arguments

`n`	number of angles to be returned
`from`	start angle in degrees. Default: 0

Examples

```
x <- equidistantAngles(90)

plot(x, sin(gradToRad(x)), xlab = "angle in degree", ylab = "sin(angle)")
```

exampleLinksAndNodes *Example Links and Nodes*

Description

Example Links and Nodes

Usage

```
exampleLinksAndNodes()
```

getAuthors *Get Information on Package Authors*

Description

Get Information on Package Authors

Usage

```
getAuthors(package)
```

Arguments

package	name of (installed) package

getCranPackageDatabase *Get Matrix with Information on All CRAN Packages*

Description

Get Matrix with Information on All CRAN Packages

Usage

```
getCranPackageDatabase()
```

`getDependencyData` *Get Package Dependency Data from Package Database*

Description

Get Package Dependency Data from Package Database

Usage

```
getDependencyData(
  db,
  fields = c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")
)
```

Arguments

db	package data base as e.g. returned by `getCranPackageDatabase`
fields	types of dependencies to be considered. Default: `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`

`getPackageFilesToInstall` *Get Package Files to Install*

Description

Get paths/names of package files available in a directory

Usage

```
getPackageFilesToInstall(
  package.dir = defaultPackageDir(),
  packageNames = NULL,
  filepattern = "",
  full.names = TRUE,
  dbg = FALSE,
  warn = TRUE
)
```

Arguments

package.dir	full path to directory containing the package files
packageNames	optional vector of character containing the names of the packages to be installed. If NULL (default), all available packages in *package.dir* are installed,
filepattern	pattern matching names of files to be considered. Default: "^kwb\..*\.zip$"
full.names	if TRUE (default) the full paths to the package files are returned, else only the file names
dbg	if TRUE (default) debug messages are shown
warn	if TRUE (default) warnings are given if no corresponding package files are found

Value

character vector containing the full paths to or just the names of the available package files

---

getPackageLicences *Which Licences are Specified for the Packages?*

Description

Which Licences are Specified for the Packages?

Usage

```
getPackageLicences(packages, db = utils::installed.packages())
```

Arguments

packages	names of (installed) packages
db	optional. Package database, similar to what is returned by installed.packages. Default: installed.packages()

Value

data frame

getRVersionMajorMinor *Helper: Get R major minor version string*

Description

Helper: Get R major minor version string

Usage

```
getRVersionMajorMinor()
```

Value

returns R version major.minor string (e.g. 4.0), used by standard R libraries for grouping all R packages into one folder

Examples

```
getRVersionMajorMinor()
```

getServername *Get KWB Servername*

Description

Get KWB Servername

Usage

```
getServername()
```

githubVersions *Get Versions of Packages on GitHub*

Description

Get Versions of Packages on GitHub

Usage

```
githubVersions(name, github_user = "KWB-R")
```

Arguments

name	package name
github_user	name of github account, default: "KWB-R"

Value

data frame with one row per available version

Examples

```
githubVersions("kwb.utils")
```

---

gradToRad *Angle in Degree to Angle in rad*

Description

Angle in Degree to Angle in rad

Usage

```
gradToRad(grad)
```

Arguments

grad	vector of angles in degrees

Examples

```
gradToRad(c(0, 90, 180, 270, 360)) / pi
```

---

hasGplLicence *Do Packages have a GPL Licence?*

Description

Do Packages have a GPL Licence?

Usage

```
hasGplLicence(packages)
```

Arguments

packages	package name(s) as a vector of character

Value

vector of logical

`initLocalCRAN` *Create the folder structure for a local CRAN-like repository*

Description

Create the folder structure for a local CRAN-like repository

Usage

```
initLocalCRAN(local_cran)
```

Arguments

`local_cran`	full path to the folder representing the local CRAN

`installedDependencies` *What Versions of Package Dependencies are Installed?*

Description

What Versions of Package Dependencies are Installed?

Usage

```
installedDependencies(package, recursive = TRUE)
```

Arguments

`package`	name of the package of which to check the dependencies
`recursive`	whether to look recursively for dependencies or only for the direct dependencies of package. Passed to `packageDependencies`, defaults to `TRUE`

Examples

```
installedDependencies(package = "kwb.package")
installedDependencies(package = "kwb.package", recursive = FALSE)
```

installedKwbPackages *Installed KWB-Packages*

Description

Installed KWB-Packages

Usage

```
installedKwbPackages()
```

Value

vector of names of installed kwb-packages

installGithubPackages *Install GitHub Packages*

Description

Install GitHub Packages

Usage

```
installGithubPackages(
  lib,
  repos,
  dependencies = TRUE,
  upgrade = "never",
  auth_token = Sys.getenv("GITHUB_PAT")
)
```

Arguments

lib	path to R library where packages should be installed
repos	vector of relative paths to GitHub repositories containing R packages (e.g. "kwb-r/kwb.utils")
dependencies	passed to remotes::install_github(). TRUE is shorthand for "Depends", "Imports", "LinkingTo" and "Suggests" NA is shorthand for "Depends", "Imports" and "LinkingTo" and is the default. FALSE is shorthand for no dependencies (i.e. just check this package, not its dependencies), (default: TRUE)
upgrade	passed to install_github, (default: "never")
auth_token	GitHub Personal Access token, required with scope "private" if access to non-public R packages is required (default: Sys.getenv("GITHUB_PAT"))

Value

installs multiple GitHub R packages into one R library

Examples

```
## Not run:
remotes::install_github("kwb-r/pkgmeta")
pkgs <- pkgmeta::get_github_packages()
paths_list <- list(
  r_version = kwb.packages::getRVersionMajorMinor(),
  lib_linux = "/usr/lib/R/site-library",
  lib_win = "<win_root_dir>/kwbran/<r_version>"
)

paths <- kwb.utils::resolve(paths_list, win_root_dir = tempdir())

pkgs <- pkgmeta::get_github_packages()

installGithubPackages(lib = paths$lib_win, pkgs$full_name)
installGithubPackages(lib = paths$lib_linux, pkgs$full_name)

## End(Not run)
```

---

installRemotes *Install the remotes Package to the Given Library*

Description

Install the remotes Package to the Given Library

Usage

```
installRemotes(lib)
```

Arguments

lib	path to the library to which to install the remotes package

packageDependencies *Package Dependencies*

Description

This is just a wrapper around package_dependencies with some defaults defined.

Usage

```
packageDependencies(
  packages = NULL,
  db = utils::installed.packages(),
  which = c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")[1:3],
  recursive = TRUE,
  reverse = FALSE,
  verbose = FALSE
)
```

Arguments

packages	a character vector of package names.
db	character matrix as from available.packages() (with the default NULL the results of this call) or data frame variants thereof. Alternatively, a package database like the one available from https://cran.r-project.org/web/packages/packages.rds.
which	a character vector listing the types of dependencies, a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"). Character string "all" is shorthand for that vector, character string "most" for the same vector without "Enhances", character string "strong" (default) for the first three elements of that vector.
recursive	a logical indicating whether (reverse) dependencies of (reverse) dependencies (and so on) should be included, or a character vector like which indicating the type of (reverse) dependencies to be added recursively.
reverse	logical: if FALSE (default), regular dependencies are calculated, otherwise reverse dependencies.
verbose	logical indicating if output should monitor the package search cycles.

packageDependenciesByType
Package Dependencies by Type

Description

Package Dependencies by Type

Usage

```
packageDependenciesByType(
  packages = NULL,
  db = utils::installed.packages(),
  which = c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")[1:3],
  recursive = TRUE,
  reverse = FALSE,
  verbose = FALSE
)
```

Arguments

packages	a character vector of package names.
db	character matrix as from available.packages() (with the default NULL the results of this call) or data frame variants thereof. Alternatively, a package database like the one available from https://cran.r-project.org/web/packages/packages.rds.
which	a character vector listing the types of dependencies, a subset of c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances"). Character string "all" is shorthand for that vector, character string "most" for the same vector without "Enhances", character string "strong" (default) for the first three elements of that vector.
recursive	a logical indicating whether (reverse) dependencies of (reverse) dependencies (and so on) should be included, or a character vector like which indicating the type of (reverse) dependencies to be added recursively.
reverse	logical: if FALSE (default), regular dependencies are calculated, otherwise reverse dependencies.
verbose	logical indicating if output should monitor the package search cycles.

packageString *Package String*

Description

Package String

Usage

```
packageString(package)
```

Arguments

package	Package name

Value

package, preceded by package:

plotAllDependencies *Plot all Package Dependencies*

Description

Plot all Package Dependencies

Usage

```
plotAllDependencies(dependencies, r = 1.5, for.each = TRUE, ...)
```

Arguments

dependencies	list of package dependencies as returned by packageDependencies
r	radius of the unit circle in which to arrange the package names. Passed to plotNodes
for.each	if TRUE (default) not only an overview plot showing all dependencies but also one plot per package of which dependency information are contained in dependencies is created.
...	arguments passed to plotDependencies

Examples

```
kwb.packages <- installedKwbPackages()

# Plot all (direct and indirect) dependencies of installed kwb packages
plotAllDependencies(packageDependencies(kwb.packages))

# Plot only direct dependencies
plotAllDependencies(packageDependencies(kwb.packages, recursive = FALSE))
```

---

plotDependencies *Plot Dependencies Between Nodes on a Circle Line*

Description

Plot Dependencies Between Nodes on a Circle Line

Usage

```
plotDependencies(
  nodes,
  dependencies,
  main = "",
  r = 1.5,
  nodeColours = grDevices::rainbow(nrow(nodes)),
  ...
)
```

Arguments

nodes	data frame as returned by toNodes
dependencies	list of package dependencies as returned by packageDependencies
main	plot title
r	radius of the circle
nodeColours	colours to be given to the nodes
...	passed to drawDependencies

plotNodes *Plot Nodes*

Description

Plot Nodes

Usage

```
plotNodes(nodes, r = 1, col = "red", ...)
```

Arguments

nodes	data frame as returned by toNodes
r	Radius of a circle that fits into the plot range: xlim is set to c(-r, r)
col	colour used to plot the nodes (default: "red")
...	arguments passed to plot

plotPackageVersions *Plot Package Versions*

Description

Plot Package Versions

Usage

```
plotPackageVersions(
  versions,
  r_range = c(1, 10),
  rmax = 1.1 * r_range[2L],
  dphi = NULL,
  ticklen = 1
)
```

Arguments

versions	versions
r_range	r_range (default: c(1, 10))
rmax	rmax (default: 1.1 * r_range[2L])
dphi	dphi (default: NULL)
ticklen	ticklen (default: 1)

plotSankeyNetwork *Plot Sankey Network*

Description

Plot Sankey Network

Usage

```
plotSankeyNetwork(functionName, where = 1, ...)
```

Arguments

functionName	name of function from which to start the network
where	passed to foodweb
...	additional arguments passed to foodweb

provideInLocalCRAN *Provide a Source Package in the Local Cran*

Description

Provide a Source Package in the Local Cran

Usage

```
provideInLocalCRAN(
  package,
  rebuild = TRUE,
  local_cran = defaultLocalCRAN(drive_letter = TRUE)
)
```

Arguments

package	name of the package to be looked up in either of these locations: <home>/Documents/R-Development/RP <home>/Desktop/R_Development/RPackages
rebuild	logical. If TRUE the package is rebuild before all .tar.gz-files from the parent folder of the package folder are copied to the local CRAN folder structure
local_cran	full path to the folder representing the local CRAN

readGithubPackageDescription *readGithubPackageDescription*

Description

Read DESCRIPTION File for R Package on GitHub

Usage

```
readGithubPackageDescription(
  repo,
  sha,
  auth_token = remotes_github_pat(),
  destdir = tempdir()
)
```

Arguments

repo	GitHub repository, e.g. "kwb-r/kwb.utils"
sha	SHA (hash) of the commit
auth_token	GitHub token
destdir	path to destination directory, i.e. directory to which the DESCRIPTION file is copied. Default: `tempdir()`

setOptionsForPackrat *Set Options for Using Packrat*

Description

Add the path to the local repository to the option "repos" and set the option "pkgType" to "source".

Usage

```
setOptionsForPackrat()
```

Value

The old options are returned invisibly.

sortedDependencies *Sorted Package Dependencies*

Description

Names of depending packages in the order of their dependency

Usage

```
sortedDependencies(package, dbg = FALSE)
```

Arguments

package	name of package(s) of which dependencies are to be found
dbg	if TRUE, debug messages are shown and the user is asked to press Enter each time the body of the main loop is passed!

Value

vector of package names. The first element is the package itself, followed by the names of depending packages. You should be able to detach the packages in this order without any "package ... is required by ..." error

stopIfNotInstalled *Is a Package Installed?*

Description

Is a Package Installed?

Usage

```
stopIfNotInstalled(package)
```

Arguments

package	package name (character vector of length one)

systemPackages *Names of Base R Packages*

Description

Names of Base R Packages

Usage

```
systemPackages(set_number = 1L)
```

Arguments

set_number	integer number specifying a set of packages: 1 or 2.

Value

vector of character representing package names

toLinksAndNodes *Convert Links to List of Links and Nodes*

Description

Convert Links to List of Links and Nodes

Usage

```
toLinksAndNodes(links)
```

Arguments

links	list with elements source and target

Value

list with elements links (input list links with new elements value, source, target) and nodes (data frame with column name)

Examples

```
kwb.package:::toLinksAndNodes(list(
  source = c("s1", "s1"), target = c("t1", "t2")
))
```

toNodes *Nodes in a Unit Circle*

Description

Node names to node coordinates in a unit circle

Usage

```
toNodes(nodeNames)
```

Arguments

nodeNames	character vector of nodes to be arranged in a unit circle

Value

data frame with columns x and y giving the coordinates of the nodes arranged in a circle. The row names represent the node names.

Examples

```
nodes <- toNodes(LETTERS)

plot(nodes)
text(nodes, labels = row.names(nodes), adj = c(0, 0))
```

updateKwbPackages *Update or Install KWB-Packages*

Description

Update installed KWB-packages or install KWB-packages for the first time

Usage

```
updateKwbPackages(
  packageNames = sort(installedKwbPackages()),
  skip = character(),
  package.dir = defaultPackageDir()
)
```

Arguments

packageNames	vector of packages to be installed. Default: names of all installed KWB-packages
skip	vector of packages not to be installed, even if they are listed in *packageNames*
package.dir	full path to the folder containing the binary package files

Index