Package: piamenv (via r-universe)

August 29, 2024

Type Package

Title Package environment support for PIAM

Version 0.5.4

Date 2024-08-29

Description Enables easier management of package environments, based on renv and Python venv.

License LGPL-3

URL https://github.com/pik-piam/piamenv

Imports desc, methods, renv, withr

Suggests covr, testthat

Encoding UTF-8

RoxygenNote 7.3.2

Repository https://pik-piam.r-universe.dev

RemoteUrl https://github.com/pik-piam/piamenv

RemoteRef HEAD

RemoteSha 887da981cb78bfda6164d04640e2921e45c694c6

Contents

`piamenv-package` *piamenv: Package environment support for PIAM*

Description

Enables easier management of package environments, based on renv and Python venv.

Author(s)

Maintainer: Pascal Sauer <pascal.sauer@pik-potsdam.de>

See Also

Useful links:

- https://github.com/pik-piam/piamenv

`archiveRenv` *archiveRenv*

Description

Writes a new <timestamp>_renv.lock based on the current package environment into renv::project()/renv/archive.

Usage

```
archiveRenv()
```

Value

Invisibly, the absolute path to the created lockfile.

Author(s)

Pascal Sauer

checkDeps *Check Dependencies*

Description

Check if package requirements specified in the given DESCRIPTION are met.

Usage

```
checkDeps(
  descriptionFile = ".",
  dependencyTypes = c("Depends", "Imports", "LinkingTo"),
  action = "stop"
)
```

Arguments

`descriptionFile`
: Path to a DESCRIPTION file or a path that belongs to a source package project. If /DEPENDENCIES exists that will be used instead.

`dependencyTypes`
: The types of dependencies to check. Must be a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`.

`action`
: Action to take on unmet dependencies:

- `"stop"`: Issue an error with the unmet dependencies. (Default.)
- `"warn"`: Issue a warning with the unmet dependencies.
- `"note"`: Issue a message with the unmet dependencies.
- `"pass"`: Do nothing, just return invisibly.
- `"ask"`: Ask the user whether to auto-fix missing dependencies. Requires an active renv. Will also write renv.lock. If no active renv is found, stops instead.

Value

Invisibly, a named list of strings indicating whether each package requirement is met (`"TRUE"`) or not, in which case the reason is stated.

Author(s)

Pascal Sauer, Michaja Pehl

Examples

```
checkDeps(system.file("DESCRIPTION", package = "piamenv"))
```

createResultsfolderPythonVirtualEnv *createResultsfolderPythonVirtualEnv*

Description

Create a copy of the source virtual environment in the given results folder in the subfolder '.venv'.

Usage

```
createResultsfolderPythonVirtualEnv(resultsfolder, sourceVenv = ".venv")
```

Arguments

resultsfolder	Path to the resultsfolder where the new virtual environment will be created.
sourceVenv	Path to the virtual environment folder which will be copied.

Author(s)

Mika Pflüger

fixDeps *Fix Dependencies*

Description

Automatically install package versions as suggested by piamenv::checkDeps into an renv.

Usage

```
fixDeps(
  ask = FALSE,
  requirementMet = checkDeps(action = if (ask) "note" else "pass")
)
```

Arguments

ask	Whether to ask before fixing dependencies.
requirementMet	The output of piamenv::checkDeps.

Value

Invisibly, the return value of renv::install.

Author(s)

Pascal Sauer

pythonBinPath *pythonBinPath*

Description

Returns the proper path to the Python binary in a given virtual environment.

Usage

```
pythonBinPath(venv)
```

Arguments

venv	Path to Python virtual environment folder.

Value

The full path to the Python binary in the virtual environment folder.

Author(s)

Mika Pflüger

restoreRenv *restoreRenv*

Description

Restores the current renv to the state described in the given lockfile. If multiple lockfiles are given, ask the user which one to restore.

Usage

```
restoreRenv(
  lockfile = Sys.glob(c("output/*/*renv.lock", "renv/archive/*renv.lock"))
)
```

Arguments

lockfile	One or more paths to lockfiles. The default value assumes an output folder with run folders containing renv.lock files, and/or an renv/archive folder with renv.lock files.

Value

Invisibly, the return value of renv::restore.

Author(s)

Pascal Sauer

`revertDevelopmentVersions` *Revert Development Versions of Packages*

Description

All PIK-PIAM packages in the current renv that are development versions, i.e. that have a non-zero fourth version number component (e.g. `0.4.3.9001`), are reverted to the highest version lower than the development versions (e.g. `0.4.3`).

Usage

```
revertDevelopmentVersions()
```

Value

Invisibly the return value of `renv::install()`.

`showUpdates` *showUpdates*

Description

Print available updates of the given packages.

Usage

```
showUpdates(packages = piamPackages())
```

Arguments

`packages`	A character vector of package names.

Value

Invisibly, a data.frame as returned by utils::old.packages

stopIfLoaded *stopIfLoaded*

Description

Throw an error if any of the given packages is loaded.

Usage

```
stopIfLoaded(updatedPackage)
```

Arguments

updatedPackage	One or more names of packages that were just updated.

Details

This is useful after updating packages. If any of the updated packages was loaded before the update R might crash when trying to lazy load a function from the updated package.

Author(s)

Pascal Sauer

Examples

```
## Not run:
updates <- piamenv::fixDeps()
piamenv::stopIfLoaded(names(updates))

updates <- piamenv::updateRenv()
piamenv::stopIfLoaded(names(updates))

## End(Not run)
```

---

updatePythonVirtualEnv *updatePythonVirtualEnv*

Description

Installs newly added requirements into the Python virtual environment.

Usage

```
updatePythonVirtualEnv(venv = ".venv", requirements = "requirements.txt")
```

Arguments

venv	Path to Python virtual environment folder.
requirements	Path to a requirements.txt file containing the current requirements.

Author(s)

Mika Pflüger

---

updateRenv *updateRenv*

Description

Update all PIK-PIAM packages in the current renv, write renv.lock into archive.

Usage

```
updateRenv(exclude = NULL)
```

Arguments

exclude	vector of packages not to be updated

Value

Invisibly, the return value of renv::update.

Author(s)

Pascal Sauer

---

writePythonVirtualEnvLockFile *writePythonVirtualEnvLockFile*

Description

Write Python virtual environment lock file specifying package versions currently installed in the given virtual environment. Using the lock file, a new virtual environment can be built containing exactly the same package versions.

Usage

```
writePythonVirtualEnvLockFile(filePath, venv = ".venv")
```

Arguments

`filePath`	Path where the virtual environment lock file will be created.
`venv`	Path to Python virtual environment folder.

Author(s)

Mika Pflüger

Index