

Package: piamenv (via r-universe)

October 16, 2026

Type Package
Title Package environment support for PIAM
Version 0.8.1
Date 2026-08-11
Description Enables easier management of package environments, based on renv and conda.
License LGPL-3
URL <https://github.com/pik-piam/piamenv>
Imports desc, methods, renv, withr
Suggests covr, testthat
Encoding UTF-8
RoxygenNote 7.3.3
Config/Needs/website tidyverse/tidytemplate
Repository <https://pik-piam.r-universe.dev>
Date/Publication 2026-08-17 21:28:49 UTC
RemoteUrl <https://github.com/pik-piam/piamenv>
RemoteRef HEAD
RemoteSha 9ce3ce01d7fd84e7310a6426e8192bfb55c62b97

Contents

piamenv-package	2
archiveRenv	2
checkDeps	3
condaInit	4
condaPackageVersion	4
condaRun	5
fixDeps	5
installDeps	6
restoreRenv	6

revertDevelopmentVersions	7
showUpdates	7
stopIfLoaded	8
updateRenv	9
Index	10

piamenv-package	<i>piamenv: Package environment support for PIAM</i>
-----------------	--

Description

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

Author(s)

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

- Tonn Rüter <tonn.rueter@pik-potsdam.de>

See Also

Useful links:

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

archiveRenv	<i>archiveRenv</i>
-------------	--------------------

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"))
```

condaInit

condaInit

Description

Determine initialization commands to make conda available

Usage

```
condaInit(how = "", log = NULL, verbose = 0)
```

Arguments

how	Method to activate conda if not found (e.g., "pik-cluster")
log	Path to log file or NULL to use stdout
verbose	Verbosity level (0 for silent, higher for more verbose)

Value

List containing 'setup' and 'teardown' commands, and environment variables ('env').

condaPackageVersion

condaPackageVersion

Description

Query the version of a Python package installed in a conda environment, as recorded in the package metadata ('importlib.metadata'). Reads the environment's own Python interpreter directly, so it requires neither an activated conda nor 'module load' (unlike [condaRun]). Fails hard with a descriptive error if the version cannot be established.

Usage

```
condaPackageVersion(package, path)
```

Arguments

package	Distribution/import name of the Python package to query, e.g. "climate_assessment"
path	Path to the conda environment prefix

Value

Trimmed, single-line version string as reported by 'importlib.metadata'

`condaRun`*condaRun*

Description

Run a command in a specified conda environment with given environment variables.

Usage

```
condaRun(  
  cmd,  
  path,  
  env = c(),  
  init = list(setup = "", teardown = ""),  
  log = NULL,  
  verbose = 0  
)
```

Arguments

<code>cmd</code>	Command to run in the conda environment
<code>path</code>	Path to the conda environment
<code>env</code>	Named vector of environment variables to set, i.e., <code>c("VAR0" = "value0", "VAR1" = "value1")</code>
<code>init</code>	List containing ‘setup’ and ‘teardown’ commands to run before and after the command
<code>log</code>	Path to log file or NULL to use stdout
<code>verbose</code>	Verbosity level (0 for silent, higher for more verbose).

Value

Invisible NULL

`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

<code>installDeps</code>	<i>installDeps</i>
--------------------------	--------------------

Description

Install dependencies, identified by `renv::dependencies` for a given path, into the current `renv`.

Usage

```
installDeps(path)
```

Arguments

`path` The path to scan for dependencies.

Author(s)

Pascal Sauer

<code>restoreRenv</code>	<i>restoreRenv</i>
--------------------------	--------------------

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`

<code>stopIfLoaded</code>	<i>stopIfLoaded</i>
---------------------------	---------------------

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)
```

`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

Index

archiveRenv, [2](#)

checkDeps, [3](#)

condaInit, [4](#)

condaPackageVersion, [4](#)

condaRun, [5](#)

fixDeps, [5](#)

installDeps, [6](#)

piamenv (piamenv-package), [2](#)

piamenv-package, [2](#)

renv::install(), [7](#)

restoreRenv, [6](#)

revertDevelopmentVersions, [7](#)

showUpdates, [7](#)

stopIfLoaded, [8](#)

updateRenv, [9](#)