

# Package ‘rpymat’

February 19, 2022

**Type** Package

**Title** Easy to Configure an Isolated 'Python' Environment

**Version** 0.1.2

**Description** Aims to create a single isolated 'Miniconda' and 'Python' environment for reproducible pipeline scripts. The package provides utilities to run system command within the 'conda' environment, making it easy to install, launch, manage, and stop 'Jupyter-lab'.

**License** Apache License (>= 2)

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.2

**URL** <https://github.com/dipterix/rpymat>

**BugReports** <https://github.com/dipterix/rpymat/issues>

**Imports** utils, reticulate (>= 1.21), fastmap (>= 1.1.0), rappdirs (>= 0.3.3), glue (>= 1.4.2), IRkernel (>= 1.3), jsonlite (>= 1.7.3), rstudioapi (>= 0.13)

**NeedsCompilation** no

**Author** Zhengjia Wang [cph, aut, cre]

**Maintainer** Zhengjia Wang <dipterix.wang@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-02-19 17:20:01 UTC

## R topics documented:

conda-env . . . . .	2
jupyter . . . . .	4
run_command . . . . .	6
run_script . . . . .	8

<b>Index</b>	<b>10</b>
--------------	-----------

---

conda-env	<i>'Miniconda' environment</i>
-----------	--------------------------------

---

## Description

These functions/variables are used to configure 'Miniconda' environment.

## Usage

```

CONDAENV_NAME(env_name)

conda_path()

conda_bin()

env_path()

configure_matlab(matlab, python_ver = "auto")

configure_conda(
  python_ver = "auto",
  packages = NULL,
  matlab = NULL,
  update = FALSE,
  force = FALSE
)

remove_conda(ask = TRUE)

add_packages(packages = NULL, python_ver = "auto", ...)

ensure_rpymat(verbose = TRUE)

matlab_engine()

call_matlab(
  fun,
  ...,
  .options = getOption("rpymat.matlab_opt", "-nodesktop -nojvm"),
  .debug = getOption("rpymat.debug", FALSE)
)

```

## Arguments

env_name	alternative environment name to use; default is "rpymat-conda-env"
matlab	'Matlab' path to add to the configuration path; see 'Details'
python_ver	python version to use; see 'Configuration'

packages	additional python or conda packages to install
update	whether to update conda; default is false
force	whether to force install the 'Miniconda' even a previous version exists; default is false. Setting <code>false=TRUE</code> rarely works. Please see 'Configuration'.
ask	whether to ask for user's agreement to remove the repository. This parameter should be true if your functions depend on <code>remove_conda</code> (see 'CRAN Repository Policy'). This argument might be removed and <code>force</code> to be interactive in the future.
...	for <code>add_packages</code> , these are additional parameters passing to <code>conda_install</code> ; for <code>call_matlab</code> , ... are the parameters passing to <code>fun</code>
verbose	whether to print messages
fun	'Matlab' function name, character (experimental)
.options	'Matlab' compiler options
.debug	whether to enable debug mode

**Value**

None

**Background & Objectives**

Package `reticulate` provides sophisticated tool-sets that allow us to call python functions within R. However, the installation of 'Miniconda' and python can be tricky on many platforms, for example, the 'M1' chip, or some other 'ARM' machines. The package `rpymat` provides easier approach to configure on these machines with totally isolated environments. Any modifications to this environment will not affect your other set ups.

Since 2014, 'Matlab' has introduced its official compiler for python. The package `rpymat` provides a simple approach to link the compiler, provided that you have proper versions of 'Matlab' installed. [Here](#) is a list of 'Matlab' versions with official compilers and their corresponding python versions.

**Configuration**

If 'Matlab' compiler is not to be installed, In most of the cases, function `configure_conda` with default arguments automatically downloads the latest 'Miniconda' and configures the latest python. If any other versions of 'Miniconda' is ought to be installed, please set options `"reticulate.miniconda.url"` to change the source location.

If 'Matlab' is to be installed, please specify the 'Matlab' path when running `configure_conda`. If the environment has been setup, `configure_matlab` can link the 'Matlab' compilers without removing the existing environment. For 'ARM' users, unfortunately, there will be no 'Matlab' support as the compilers are written for the 'Intel' chips.

**Initialization**

Once conda and python environment has been installed, make sure you run `ensure_rpymat()` before running any python code. This function will make sure correct compiler is linked to your current R session.

## Examples

```
# The script will interactively install \code{conda} to `R_user_dir`
## Not run:

# Install conda and python 3.9
configure_conda(python_ver = '3.9')

# Add packages h5py, pandas, jupyter
add_packages(c('h5py', 'pandas', 'jupyter'))

# Add pip packages
add_packages("itk", pip = TRUE)

# Initialize the isolated environment
ensure_rpymat()

# Remove the environment
remove_conda()

## End(Not run)
```

---

jupyter

*Install, register, launch 'Jupyter' notebook to the virtual environment*

---

## Description

Install, register, launch 'Jupyter' notebook to the virtual environment

## Usage

```
add_jupyter(..., register_R = TRUE)

jupyter_bin()

jupyter_register_R(
  user = NULL,
  name = "ir",
  displayname = "R",
  rprofile = NULL,
```

```

    prefix = NULL,
    sys_prefix = NULL,
    verbose = getOption("verbose")
)

jupyter_options(
  root_dir,
  host = "127.0.0.1",
  port = 8888,
  open_browser = FALSE,
  token = rand_string()
)

jupyter_launch(
  host = "127.0.0.1",
  port = 8888,
  open_browser = TRUE,
  workdir = getwd(),
  async = FALSE,
  ...,
  dry_run = FALSE
)

jupyter_check_launch(
  port = 8888,
  host = "127.0.0.1",
  open_browser = TRUE,
  workdir = getwd(),
  async = "auto",
  ...
)

jupyter_server_list()

jupyter_server_stop(port, ...)

jupyter_server_stop_all(...)

```

### Arguments

...	for <code>add_jupyter</code> , these are additional parameters passed to <code>jupyter_register_R</code> ; for <code>jupyter_launch</code> , these are additional parameters passed to <code>jupyter_options</code>
<code>register_R</code>	whether to register IRkernel to the notebook
<code>user</code> , <code>name</code> , <code>displayname</code> , <code>rprofile</code> , <code>prefix</code> , <code>sys_prefix</code> , <code>verbose</code>	see <a href="#">installspec</a>
<code>root_dir</code> , <code>workdir</code>	default root directory of the notebook
<code>host</code> , <code>port</code>	'IP' and port of the hosting 'URL'

open_browser	whether to open the browser once launched
token	access token of the notebook
async	whether to open the notebook in the background
dry_run	whether to display the command instead of executing them; used to debug the code

### Value

jupyter\_bin returns the 'Jupyter' notebook binary path; jupyter\_options returns the 'Jupyter' configuration in strings; jupyter\_server\_list returns a table of existing local 'Jupyter' server hosts, ports, and tokens; jupyter\_check\_launch returns true if a new server has been created, or false if there has been an existing server at the port; other functions return nothing.

### Examples

```
## Not run:

# Requires installation of conda
library(rpymat)

# Install conda, if you have done so, skip
configure_conda()

# Install Jupyter notebook
add_jupyter(register_R = TRUE)

# Utility functions
jupyter_bin()

# Please install `dipsaus` package to enable `async=TRUE` with
# better experience
jupyter_launch(async = FALSE, open_browser = TRUE)

## End(Not run)
```

---

run\_command

*Execute command with additional environments*

---

### Description

Enables 'conda' environment

**Usage**

```

cmd_create(command, shell, use_glue = TRUE)

cmd_set_env(command, key, value, quote = TRUE, quote_type = "cmd")

cmd_set_workdir(command, workdir)

cmd_set_conda(command, conda_path, env_path)

cmd_build(command, .env = parent.frame(), ...)

detect_shell(suggest = NULL)

run_command(
  command,
  shell = detect_shell(),
  use_glue = FALSE,
  enable_conda = TRUE,
  stdout = "",
  stderr = "",
  stdin = "",
  input = NULL,
  env_list = list(),
  wait = TRUE,
  timeout = 0,
  ...,
  workdir = getwd(),
  dry_run = FALSE,
  print_cmd = dry_run,
  glue_env = parent.frame()
)

```

**Arguments**

command	system command
shell	shell type
use_glue	whether to <a href="#">glue</a> the command; default is false
key, value	environment variable key and value
quote, quote_type	whether to quote the environment variables and what quote type should use; see <a href="#">shQuote</a>
workdir	the working directory
conda_path	'conda' path; default is <a href="#">conda_path</a>
env_path	'conda' environment path; default is <a href="#">env_path</a>
suggest	suggested shell type; default is 'cmd' on windows, or 'bash' on others
enable_conda	whether to activate 'conda'

stdout, stderr, stdin, input, wait, timeout, ...  
 passed to [system2](#)

env\_list a key-value pairs of environment variables

dry\_run whether to dry-run the command (do not execute, simply returns the command),  
 useful to debug

print\_cmd whether to print the command out

glue\_env, .env the environment to evaluate variables when use\_glue is true

### Value

All the functions return a list with class `rpymat_system_command` except for `run_command`, which returns the exit code by [system2](#).

### Examples

```
run_command("conda install -y numpy", dry_run = TRUE)

a <- "This is a message"
run_command('echo "{a}"', dry_run = TRUE,
            enable_conda = FALSE, use_glue = TRUE)

## Not run:

# Use `jupyter_launch()` instead. This is just a demonstration
run_command("{jupyter_bin()} server list", use_glue = TRUE)

## End(Not run)
```

---

run\_script

*Run 'Python' script*

---

### Description

A wrapper of [py\\_run\\_file](#), but with `rpymat` enabled

### Usage

```
run_script(x, work_dir = NULL, local = FALSE, convert = FALSE)
```

### Arguments

x script path

work\_dir working directory of the script

local, convert passed to [py\\_run\\_file](#)



**Value**

The values returned by `py_run_file`

**Examples**

```
## Not run:  
  
# Please configure conda environment first  
  
x <- tempfile()  
writeLines(c(  
  "import re",  
  "zipcode = re.findall(r'[0-9]{5,6}', r.address)"  
) , con = x)  
  
address <- '2341 Main St., 72381'  
rpymat::run_script(x)  
  
py$zipcode  
  
## End(Not run)
```

# Index

`add_jupyter` (`jupyter`), 4  
`add_packages` (`conda-env`), 2

`call_matlab` (`conda-env`), 2  
`cmd_build` (`run_command`), 6  
`cmd_create` (`run_command`), 6  
`cmd_set_conda` (`run_command`), 6  
`cmd_set_env` (`run_command`), 6  
`cmd_set_workdir` (`run_command`), 6  
`conda-env`, 2  
`conda_bin` (`conda-env`), 2  
`conda_install`, 3  
`conda_path`, 7  
`conda_path` (`conda-env`), 2  
`CONDAENV_NAME` (`conda-env`), 2  
`configure_conda` (`conda-env`), 2  
`configure_matlab` (`conda-env`), 2

`detect_shell` (`run_command`), 6

`ensure_rpymat` (`conda-env`), 2  
`env_path`, 7  
`env_path` (`conda-env`), 2

`glue`, 7

`installspec`, 5

`jupyter`, 4  
`jupyter_bin` (`jupyter`), 4  
`jupyter_check_launch` (`jupyter`), 4  
`jupyter_launch` (`jupyter`), 4  
`jupyter_options` (`jupyter`), 4  
`jupyter_register_R` (`jupyter`), 4  
`jupyter_server_list` (`jupyter`), 4  
`jupyter_server_stop` (`jupyter`), 4  
`jupyter_server_stop_all` (`jupyter`), 4

`matlab_engine` (`conda-env`), 2

`py_run_file`, 8, 9

`remove_conda` (`conda-env`), 2  
`run_command`, 6  
`run_script`, 8

`shQuote`, 7  
`system2`, 8