Package 'outsider'

December 9, 2020

Type Package

Title Install and Run Programs, Outside of R, Inside of R

Version 0.1.1

Maintainer Dom Bennett <dominic.john.bennett@gmail.com>

Description Install and run external command-line programs in R through use of 'Docker' <https://www.docker.com/> and online repositories.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

SystemRequirements docker (>=18.0.0)

URL https://github.com/ropensci/outsider#readme, https://docs.ropensci.org/outsider/

BugReports https://github.com/ropensci/outsider/issues/

Language en-GB

Depends R (>= 3.3.0)

Imports outsider.base, remotes (>= 2.0), crayon, jsonlite, tibble, curl, yaml

Suggests testthat (>= 2.1), knitr (>= 1.0), ape, rlang, ssh

NeedsCompilation no

Author Dom Bennett [aut, cre, cph] (<https://orcid.org/0000-0003-2722-1359>), Hannes Hettling [ctb] (<https://orcid.org/0000-0003-4144-2238>), Daniele Silvestro [ctb] (<https://orcid.org/0000-0003-0100-0961>), Rutger Vos [ctb] (<https://orcid.org/0000-0001-9254-7318>), Alexandre Antonelli [ctb] (<https://orcid.org/0000-0003-1842-9297>), Anna Krystalli [rev]

Repository CRAN

Date/Publication 2020-12-09 09:20:09 UTC

R topics documented:

`bitbucket_repo_search` *Search for repository*

Description

Return bitbucket API item for specific repository.

Usage

```
bitbucket_repo_search(repo)
```

Arguments

`repo`	bitbucket repo

Value

data.frame

`bitbucket_search` *Search for outsider modules in bitbucket*

Description

Returns bitbucket API item results for outsider module search.

Usage

```
bitbucket_search(...)
```

Arguments

`...`	Arguments

Details

Function is NOT available. This is a stub for when BitBucket API updates.

Value

data.frame

`bitbucket_tags` *Module tags from bitbucket*

Description

Return tbl_df of module tags for a list of outsider modules hosted on bitbucket.

Usage

```
bitbucket_tags(repos)
```

Arguments

`repos`	Character vector of outsider module repositories.

Value

tbl_df

github_repo_search *Search for repository*

Description

Return GitHub API item for specific repository.

Usage

```
github_repo_search(repo)
```

Arguments

repo	GitHub repo

Value

data.frame

github_search *Search for outsider modules in GitHub*

Description

Returns GitHub API item results for outsider module search.

Usage

```
github_search()
```

Value

data.frame

github_tags *Module tags from GitHub*

Description

Return tbl_df of module tags for a list of outsider modules hosted on GitHub.

Usage

```
github_tags(repos)
```

Arguments

repos	Character vector of outsider module repositories.

Value

tbl_df

gitlab_repo_search *Search for repository*

Description

Return gitlab API item for specific repository.

Usage

```
gitlab_repo_search(repo)
```

Arguments

repo	gitlab repo

Value

data.frame

gitlab_search *Search for outsider modules in GitLab*

Description

Returns GitLab API item results for outsider module search.

Usage

```
gitlab_search()
```

Value

data.frame

gitlab_tags *Module tags from GitLab*

Description

Return tbl_df of module tags for a list of outsider modules hosted on gitlab.

Usage

```
gitlab_tags(repo_ids)
```

Arguments

repo_ids	Character vector of outsider module repositories ids.

Value

tbl_df

`is_module_installed` *Is module installed?*

Description

Check if a module is installed on your system.

Usage

```
is_module_installed(repo)
```

Arguments

`repo`	Module repo

Details

Searches for `repo` among installed outsider modules. Returns TRUE if found, else FALSE.

Value

Logical

See Also

Other public: `module_details()`, `module_functions()`, `module_help()`, `module_import()`, `module_installed()`, `module_install()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
# NOT RUN (too slow for automated testing)
## Not run:
  if (is_outsider_ready()) {
    # simplest repo
    repo <- 'dombennett/om..hello.world'
    # install
    module_install(repo = repo, force = TRUE, update = 'never')
    # is module_installed?
    (is_module_installed(repo = repo))
    # uninstall
    module_uninstall(repo)
  }

## End(Not run)
```

is_outsider_ready *Is outsider ready to run?*

Description

Return TRUE if modules can be installed and run. Provides helpful messages on what may be missing.

Usage

```
is_outsider_ready()
```

Value

logical

module_details *Look up details on module(s)*

Description

Return a tbl_df of information for outsider module(s) for a given code-sharing service. If `repo` is NULL, will return details on all available modules.

Usage

```
module_details(repo = NULL, service = c("github", "bitbucket", "gitlab"))
```

Arguments

`repo`	Vector of one or more outsider module repositories, default NULL.
`service`	Code-sharing service, e.g. GitHub

Details

Module details in tibble format include: repository name (user/repo), last time repo was updated, number of watchers (or stars in the case of GitLab), url to web presence, names of tagged versions.

Value

tbl_df

See Also

Other public: `is_module_installed()`, `module_functions()`, `module_help()`, `module_import()`, `module_installed()`, `module_install()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
# return table of ALL available modules on GitHub
# NOT RUN - takes too long
## Not run:
  (available_modules <- module_search())

## End(Not run)

# look-up specific modules
repo <- 'dombennett/om..goodbye.world'
(suppressWarnings(module_details(repo = repo))) # no module exists, expect warning
repo <- 'dombennett/om..hello.world'
(module_details(repo = repo))
```

`module_functions` *List the functions associated with a module*

Description

Return a vector of functions that can be imported from the module.

Usage

```
module_functions(repo)
```

Arguments

`repo`	Module repo

Value

character

See Also

Other public: `is_module_installed()`, `module_details()`, `module_help()`, `module_import()`, `module_installed()`, `module_install()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
if (is_outsider_ready()) {
  # simplest repo
  repo <- 'dombennett/om..hello.world'

  # is module_installed?
  if (is_module_installed(repo = repo)) {
```

```
    # get help for package
    module_help(repo = repo)

    # list functions available
    module_functions(repo = repo)

    # import
    hello_world <- module_import(fname = 'hello_world', repo = repo)

    # get help for function
    module_help(repo = repo, fname = 'hello_world')
    # also works
    ?hello_world

    # run function
    hello_world()

    # change verbosity settings

    # print nothing to console
    verbosity_set(show_program = FALSE, show_docker = FALSE)
    hello_world()

    # print everything to console
    verbosity_set(show_program = TRUE, show_docker = TRUE)
    hello_world()

    # write program output to a file
    log_file <- tempfile()
    verbosity_set(show_program = log_file, show_docker = FALSE)

    hello_world()

    (readLines(con = log_file))

    # Clean up
    file.remove(log_file)
  }
}
```

`module_help` *Get help for outsider modules*

Description

Look up help files for specific outsider module functions or whole modules.

Usage

```
module_help(repo, fname = NULL)
```

Arguments

repo	Module repo
fname	Function name

See Also

Other public: is_module_installed(), module_details(), module_functions(), module_import(), module_installed(), module_install(), module_search(), module_uninstall()

Examples

```
library(outsider)
if (is_outsider_ready()) {
  # simplest repo
  repo <- 'dombennett/om..hello.world'

  # is module_installed?
  if (is_module_installed(repo = repo)) {

    # get help for package
    module_help(repo = repo)

    # list functions available
    module_functions(repo = repo)

    # import
    hello_world <- module_import(fname = 'hello_world', repo = repo)

    # get help for function
    module_help(repo = repo, fname = 'hello_world')
    # also works
    ?hello_world

    # run function
    hello_world()

    # change verbosity settings

    # print nothing to console
    verbosity_set(show_program = FALSE, show_docker = FALSE)
    hello_world()

    # print everything to console
    verbosity_set(show_program = TRUE, show_docker = TRUE)
    hello_world()

    # write program output to a file
    log_file <- tempfile()
    verbosity_set(show_program = log_file, show_docker = FALSE)

    hello_world()
```

```
    (readLines(con = log_file))

    # Clean up
    file.remove(log_file)
  }
}
```

`module_import` *Import functions from a module*

Description

Import specific functions from an outsider module to the Global Environment.

Usage

```
module_import(fname, repo)
```

Arguments

`fname`	Function name to import
`repo`	Module repo

Value

Function

See Also

Other public: `is_module_installed()`, `module_details()`, `module_functions()`, `module_help()`, `module_installed()`, `module_install()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
if (is_outsider_ready()) {
  # simplest repo
  repo <- 'dombennett/om..hello.world'

  # is module_installed?
  if (is_module_installed(repo = repo)) {

    # get help for package
    module_help(repo = repo)

    # list functions available
    module_functions(repo = repo)
```

```
    # import
    hello_world <- module_import(fname = 'hello_world', repo = repo)

    # get help for function
    module_help(repo = repo, fname = 'hello_world')
    # also works
    ?hello_world

    # run function
    hello_world()

    # change verbosity settings

    # print nothing to console
    verbosity_set(show_program = FALSE, show_docker = FALSE)
    hello_world()

    # print everything to console
    verbosity_set(show_program = TRUE, show_docker = TRUE)
    hello_world()

    # write program output to a file
    log_file <- tempfile()
    verbosity_set(show_program = log_file, show_docker = FALSE)

    hello_world()

    (readLines(con = log_file))

    # Clean up
    file.remove(log_file)
  }
}
```

`module_install` *Install an outsider module*

Description

Install a module through multiple different methods: via a code sharing site such as GitHub, a URL, a git repository or local filepath. The function will first install the R package and then build the Docker image. Docker image version is determined by "tag". To avoid pulling the image from DockerHub set "manual" to TRUE.

Usage

```
module_install(
  repo = NULL,
  url = NULL,
  filepath = NULL,
```

```
  git = NULL,
  service = c("github", "bitbucket", "gitlab"),
  tag = "latest",
  manual = FALSE,
  verbose = FALSE,
  force = FALSE,
  update = c("default", "ask", "always", "never")
)
```

Arguments

repo	Module repo, character.
url	URL to downloadable compressed (zip, tar or bzipped/gzipped) folder of a module, character.
filepath	Filepath to uncompressed directory of module, character.
git	URL to git repository
service	Code-sharing service. Character.
tag	Module version, default latest. Character.
manual	Build the docker image? Default FALSE. Logical.
verbose	Be verbose? Default FALSE.
force	Ignore warnings and install anyway? Default FALSE.
update	Update dependent R packages?

Details

All installation options depend on the installation functions of `remotes`. E.g. GitHub packages are installed with `install_github`. See these functions for more details on the R package installation process.

Value

Logical

See Also

Other public: `is_module_installed()`, `module_details()`, `module_functions()`, `module_help()`, `module_import()`, `module_installed()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
# NOT RUN (too slow for automated testing)
## Not run:
  if (is_outsider_ready()) {
    # simplest repo
    repo <- 'dombennett/om..hello.world'
    # install
```

```
  module_install(repo = repo, force = TRUE, update = 'never')
  # is module_installed?
  (is_module_installed(repo = repo))
  # uninstall
  module_uninstall(repo)
}

## End(Not run)
```

`module_installed` *Which outsider modules are installed?*

Description

Returns tbl_df of details for all outsider modules installed on the user's computer.

Usage

```
module_installed()
```

Value

tbl_df

See Also

Other public: `is_module_installed()`, `module_details()`, `module_functions()`, `module_help()`, `module_import()`, `module_install()`, `module_search()`, `module_uninstall()`

Examples

```
library(outsider)
# check what modules are installed
if (is_outsider_ready()) {
  (module_installed())
}
```

`module_search` *Search for available outsider modules*

Description

Return a list of available outsider modules. (Not possible for BitBucket.)

Usage

```
module_search(service = c("github", "gitlab"))
```

Arguments

service	Code-sharing service, e.g. GitHub

Details

Note: To search GitLab an access token is required. To create one: 1. Visit the personal access tokens section of your GitLab profile https://about.gitlab.com/ 2. Create a new token with api scope 3. Save the generated token to .Renviron (try `usethis::edit_r_environ()`) with the line "GITLAB_PAT=your access token"

For increased search relaiability, a token can be created for GitHub as well. Visit https://github.com/settings/tokens to create a token and save it to .Renviron as "GITHUB_PAT".

Value

Character vector

See Also

Other public: `is_module_installed()`, `module_details()`, `module_functions()`, `module_help()`, `module_import()`, `module_installed()`, `module_install()`, `module_uninstall()`

Examples

```
library(outsider)
# return table of ALL available modules on GitHub
# NOT RUN - takes too long
## Not run:
  (available_modules <- module_search())

## End(Not run)

# look-up specific modules
repo <- 'dombennett/om..goodbye.world'
(suppressWarnings(module_details(repo = repo))) # no module exists, expect warning
repo <- 'dombennett/om..hello.world'
(module_details(repo = repo))
```

---

module_uninstall *Uninstall and remove a module*

Description

Uninstall outsider module and removes it from your docker

Usage

```
module_uninstall(repo)
```

Arguments

repo	Module repo

Details

If program is successfully removed from your system, TRUE is returned else FALSE.

Value

Logical

See Also

Other public: is_module_installed(), module_details(), module_functions(), module_help(), module_import(), module_installed(), module_install(), module_search()

Examples

```
library(outsider)
# NOT RUN (too slow for automated testing)
## Not run:
  if (is_outsider_ready()) {
    # simplest repo
    repo <- 'dombennett/om..hello.world'
    # install
    module_install(repo = repo, force = TRUE, update = 'never')
    # is module_installed?
    (is_module_installed(repo = repo))
    # uninstall
    module_uninstall(repo)
  }

## End(Not run)
```

---

outsider *outsider: Install and run programs, outside of R, inside of R*

Description

The outsider package facilitates the installation and running of external software by interfacing with docker (https://www.docker.com/). External software are contained within mini-R-packages, called "outsider modules" and can be installed directly to a user's computer through online code-sharing services such as GitHub (https://github.com/). The outsider package comes with a series of functions for identifying, installing and importing these outsider modules.

Details

For more information visit the outsider website (https://docs.ropensci.org/outsider/).

pkgnm_guess *Guess package name*

Description

Return package name from a repo name.

Usage

```
pkgnm_guess(repo, call_error = TRUE)
```

Arguments

repo	Repository (e.g. "username/repo") or package name associated with module
call_error	Call error if no package found? Default, TRUE.

Value

character(1)

ssh_setup *Setup SSH*

Description

Send all outsider commands to an external host. Provide an ssh session to this function and all subsequent commands will be run on the host rather than the local machine. When finished it is always good practice to disconnect from the remote host by running ssh_teardown. It is required that the remote host has Docker running.

Usage

```
ssh_setup(session)
```

Arguments

session	ssh session, see ssh_connect

Value

logical

Examples

```
library(outsider)

# To forward all Docker commands to a remote host:
# 1. Gain ssh access to a remote host
# 2. Ensure Docker is running on the remote machine
# 3. Supply the IP address and authentication args to ssh::ssh_connect
ip_address <- NULL

if (!is.null(ip_address)) {
  # Create an ssh session
  session <- ssh::ssh_connect(host = ip_address)

  # Setup the session for running outsider
  ssh_setup(session)
}

# After setup, run outsider as normal

# simplest repo
repo <- 'dombennett/om..hello.world'

if (is_module_installed(repo = repo)) {
  # import
  hello_world <- module_import(fname = 'hello_world', repo = repo)

  # run function
  hello_world()
}

# Always ensure to disconnect after a session
if (!is.null(ip_address)) {
  ssh_teardown()
}
```

`ssh_teardown` *Teardown SSH*

Description

Disconnect from a remote host and stop commands being transferred.

Usage

```
ssh_teardown()
```

Value

logical

Examples

```
library(outsider)

# To forward all Docker commands to a remote host:
# 1. Gain ssh access to a remote host
# 2. Ensure Docker is running on the remote machine
# 3. Supply the IP address and authentication args to ssh::ssh_connect
ip_address <- NULL

if (!is.null(ip_address)) {
  # Create an ssh session
  session <- ssh::ssh_connect(host = ip_address)

  # Setup the session for running outsider
  ssh_setup(session)
}

# After setup, run outsider as normal

# simplest repo
repo <- 'dombennett/om..hello.world'

if (is_module_installed(repo = repo)) {
  # import
  hello_world <- module_import(fname = 'hello_world', repo = repo)

  # run function
  hello_world()
}

# Always ensure to disconnect after a session
if (!is.null(ip_address)) {
  ssh_teardown()
}
```

---

travis_build_status *Check Travis build status*

Description

Is build passing on travis? Returns either TRUE or FALSE.

Usage

```
travis_build_status(repo)
```

Arguments

repo	repo

Details

For GitHub-based repositories only.

Value

Logical

---

user_warn *Warn users on the dangers of outsider modules*

Description

Warn users on the dangers of installing an outsider module whose origin is potentially unknown.

Usage

```
user_warn(pkgnm)
```

Arguments

pkgnm	Package name

Details

Prints additional info to screen based on module YAML file.

Value

Logical

---

verbosity_set *Set the verbosity of modules*

Description

Control console messages of running outsider modules. Allow either the external program messages to run, the Docker messages or both.

Usage

```
verbosity_set(show_program = TRUE, show_docker = FALSE)
```

Arguments

show_program	Show external program messages? Default TRUE.
show_docker	Show docker messages? Default FALSE.

Details

For more control see log_set

Value

data.frame

Examples

```
library(outsider)
# NOT RUN (too slow for automated testing)
## Not run:
  if (is_outsider_ready()) {
    # simplest repo
    repo <- 'dombennett/om..hello.world'
    # install
    module_install(repo = repo, force = TRUE, update = 'never')
    # is module_installed?
    (is_module_installed(repo = repo))
    # uninstall
    module_uninstall(repo)
  }

## End(Not run)
```

---

yaml_fetch *Safely fetch om.yaml*

Description

Return list of 'program' and 'details'.

Usage

```
yaml_fetch(url)
```

Arguments

url	URL to repo

Value

list

`yaml_read` *Module YAML information*

Description

Return tbl_df of all YAML information of given outsider module repos.

Usage

```
yaml_read(repos, service = c("github", "gitlab", "bitbucket"))
```

Arguments

`repos`	Character vector of outsider module repositories on GitHub, GitLab or BitBucket.
`service`	Code-sharing service. Character.

Value

tbl_df

Index