Package: condathis (via r-universe)

September 13, 2024

Title Condathis: run any CLI tool on a Conda Environment

Version 0.0.3.9023

Description Condathis is an R package that simplifies the execution of command line tools within isolated conda environments. It enables users to effortlessly manage Conda environments, execute command line tools, handle dependencies, and ensure reproducibility in their data analysis workflows.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/luciorq/condathis

BugReports https://github.com/luciorq/condathis/issues

Imports cli, fs, jsonlite, processx, rappdirs, stringr, utils, withr

Suggests testthat (>= 3.0.0), dockerthis, curl

Remotes github::luciorq/dockerthis

Config/testthat/edition 3

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

RemoteUrl https://github.com/luciorq/condathis

RemoteRef HEAD

RemoteSha 052898d85a6f945fbad304ac70b7d4a10959c05f

Contents

`build_container_image` *Build Micromamba Container Image*

Description

Build Micromamba Container Image

Usage

```
build_container_image(
  dockerfile_path = NULL,
  image_name = "luciorq/condathis-micromamba:latest",
  force = FALSE,
  method = "docker"
)
```

Arguments

`dockerfile_path`	Character. Path to Dockerfile.
`image_name`	Character. Image name used for the contanier image. Defaults to `"condathis-micromamba:latest"`.
`force`	Logical. Should image be removed before building. Defaults to FALSE.
`method`	Character. One of `c("docker", "singularity")`.

`build_container_image_singularity` *Build Singularity / Apptainer Image*

Description

Build Singularity / Apptainer Image

Usage

```
build_container_image_singularity(
  image_name = "luciorq/condathis-micromamba:latest",
  registry_name = "docker"
)
```

Arguments

`image_name`	Character. Image name used for the container image, containing the remote repository, separated by forward slash. Defaults to `"luciorq/condathis-micromamba:latest"`.
`registry_name`	Character. Container Registry where image already exists. Defaults to `"docker"` (Docker Hub).

`create_env` *Create a Conda Environment*

Description

Create Conda Environment with specific packages installed to be used by `run()`.

Usage

```
create_env(
  packages = NULL,
  env_file = NULL,
  env_name = "condathis-env",
  channels = c("bioconda", "conda-forge"),
  method = c("native", "auto", "docker", "singularity"),
  container_name = "condathis-micromamba-base",
  image_name = "luciorq/condathis-micromamba:latest",
  sif_image_path = NULL,
  additional_channels = NULL,
  gpu_container = FALSE,
  platform = NULL,
  verbose = TRUE
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`env_file`	Character. Path to the YAML file with Conda Environment description. If this argument is used, the `packages` argument should not be included in the command.
`env_name`	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`method`	Character. Beckend method to run `micromamba`, the default is "auto" running "native" with the `micromamba` binaries installed by `condathis`. Additional methods are supported for using Docker Linux Containers "docker" and Singularity Containers "singularity" as the beckends, those are especially useful if running on systems where the Conda recipes are not available for the OS or CPU architecture in place. The container-based backends leverage `dockerthis` R package.
`container_name`	Character. Name of the Container created by Docker. Defaults to `"condathis-micromamba-base"`.
`image_name`	Character. Name of the Docker Image used, it will try to pull it automatically if internet connection is available. Defaults to `"luciorq/condathis-micromamba:latest"`.
`sif_image_path`	Character. Path to SIF image file.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`gpu_container`	Logical. GPU support for Container Beckend `methods`. This argument is not necessary if running native. Default to FALSE.
`platform`	Character. Platform to search for `packages`. Defaults to `NULL` which will use the current platform. E.g. "linux-64", "linux-32", "osx-64", "win-64", "win-32", "noarch". Note: on Apple Silicon MacOS will use "osx-64" instead of "osx-arm64" if Rosetta 2 is available.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`create_env_internal_docker`
Create Environment Using Docker

Description

Create Environment Using Docker

Usage

```
create_env_internal_docker(
  packages = NULL,
  env_file = NULL,
  env_name = "condathis-env",
  channels = c("bioconda", "conda-forge"),
  container_name = "condathis-micromamba-base",
  image_name = "luciorq/condathis-micromamba:latest",
  additional_channels = NULL,
  verbose = TRUE
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`env_file`	Character. Path to the YAML file with Conda Environment description. If this argument is used, the `packages` argument should not be included in the command.
`env_name`	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`container_name`	Character. Name of the Container created by Docker. Defaults to `"condathis-micromamba-base"`.
`image_name`	Character. Name of the Docker Image used, it will try to pull it automatically if internet connection is available. Defaults to `"luciorq/condathis-micromamba:latest"`.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`create_env_internal_singularity`

Create Environment Using Singularity / Apptainer

Description

Create Environment Using Singularity / Apptainer

Usage

```
create_env_internal_singularity(
  packages = NULL,
  env_file = NULL,
  env_name = "condathis-env",
  channels = c("bioconda", "conda-forge"),
  sif_image_path = NULL,
  additional_channels = NULL,
  verbose = TRUE
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`env_file`	Character. Path to the YAML file with Conda Environment description. If this argument is used, the `packages` argument should not be included in the command.
`env_name`	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`sif_image_path`	Character. Path to SIF image file.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`define_method_to_use` *Automatically Find Suitable Infrastrucure to Run CLI Tools*

Description

Automatically Find Suitable Infrastrucure to Run CLI Tools

Usage

```
define_method_to_use(
  packages = NULL,
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL,
  container_name = "condathis-micromamba-base",
  image_name = "luciorq/condathis-micromamba:latest",
  sif_image_path = NULL
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`container_name`	Character. Name of the Container created by Docker. Defaults to `"condathis-micromamba-base"`.
`image_name`	Character. Name of the Docker Image used, it will try to pull it automatically if internet connection is available. Defaults to `"luciorq/condathis-micromamba:latest"`.
`sif_image_path`	Character. Path to SIF image file.

`define_platform` *Define Platform to be used by* `create_env()`.

Description

Define Platform to be used by `create_env()`.

Usage

```
define_platform(
  packages,
  platform = NULL,
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`platform`	Character. Platform to search for `packages`. Defaults to `NULL` which will use the current platform. E.g. "linux-64", "linux-32", "osx-64", "win-64", "win-32", "noarch". Note: on Apple Silicon MacOS will use "osx-64" instead of "osx-arm64" if Rosetta 2 is available.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.

`env_exists` *Check If Environment Names Already exists*

Description

Check If Environment Names Already exists

Usage

```
env_exists(env_name = "condathis-env")
```

Arguments

`env_name`	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.

`format_channels_args` *Format Channels Arguments*

Description

Format Channels Arguments

Usage

```
format_channels_args(...)
```

Arguments

`...`	Character. Channels to be used by `micromamba`.

`format_user_arg_string` *Format user string for Docker*

Description

Format user string for Docker

Usage

```
format_user_arg_string()
```

`get_install_dir` *Retrieve* `condathis` *Data Creation Path*

Description

Retrieve `condathis` Data Creation Path

Usage

```
get_install_dir()
```

`get_sys_arch` *Return OS and CPU Architecture*

Description

Return OS and CPU Architecture

Usage

```
get_sys_arch()
```

`install_micromamba` *Install Micromamba binaries in a* `condathis` *controlled path.*

Description

This function downloads and installs the Micromamba binaries in the path managed by the `condathis` package. Micromamba is a lightweight implementation of the Conda package manager and provides an efficient way to create and manage conda environments.

Usage

```
install_micromamba(
  timeout_limit = 3600,
  download_method = "auto",
  force = FALSE
)
```

Arguments

`timeout_limit`	Numeric. Timeout limit for downloading the Micromamba binaries, in seconds. Defaults to 3600 seconds (1 hour).
`download_method`	Character. Argument passed to the `method` argument of the `utils::download.file()` function used for downloading the binaries. Defaults to "auto".
`force`	Logical. If set to TRUE, the download and installation of the Micromamba binaries will be forced, even if they already exist in the system or `condathis` controlled path. Defaults to FALSE.

`install_packages` *Install Packages in a Existing Conda Environment*

Description

Install Packages in a Existing Conda Environment

Usage

```
install_packages(
  packages,
  env_name = "condathis-env",
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL,
  verbose = TRUE
)
```

Arguments

`packages`	Character vector with the names of the packages and version strings if necessary.
`env_name`	Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`is_cmd_available` *Check If Command Line Tool Is Available*

Description

Test the required methods to check if command line tool is available on PATH, Before trying to execute it.

Usage

```
is_cmd_available(cmd)
```

Arguments

`cmd`	Character. The main command to be executed in the Conda environment.

`is_micromamba_available_for_arch` *Check if Micromamba is Available for OS and CPU architecture*

Description

Check if Micromamba is Available for OS and CPU architecture

Usage

```
is_micromamba_available_for_arch(sys_arch = NULL)
```

Arguments

`sys_arch`	Default: NULL.

`is_singularity_available` *Are Singularity or Apptainer CLIs available*

Description

Test if Singularity or Apptainer CLIs are available on PATH.

Usage

```
is_singularity_available()
```

`list_envs` *List Installed Environments*

Description

List Installed Environments

Usage

```
list_envs(verbose = FALSE)
```

Arguments

`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`list_packages` *List Packages Installed Inside Environment*

Description

List Packages Installed Inside Environment

Usage

```
list_packages(env_name = "condathis-env", verbose = FALSE)
```

Arguments

`env_name`	Character. The name of the Conda environment where the tool will be run. Defaults to 'condathis-env'. If the specified environment does not exist, it will be created automatically using create_env() function from the `condathis` package.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`micromamba_bin_path` *Retrieve the path to the* `condathis` *managed installation of* `micromamba`*.*

Description

Retrieve the path to the `condathis` managed installation of `micromamba`.

Usage

```
micromamba_bin_path()
```

`micromamba_user_installed` *Retrieve Micromamba Installed Path*

Description

Checks if user already have `micromamba` binaries available on `PATH`.

Usage

```
micromamba_user_installed()
```

`native_cmd` *Run Micromamba Command*

Description

Run a command using micromamba executable in the native backend.

Usage

```
native_cmd(conda_cmd, conda_args = NULL, ..., verbose = TRUE, stdout = "|")
```

Arguments

`conda_cmd`	Character. Conda subcommand to be run. E.g. "create", "install", "env", "–help", "–version".
`conda_args`	Character vector. Additional arguments passed to the Conda command.
`...`	Additional arguments to be used in the command. These arguments will be passed directly to the command executed in the Conda environment. File paths should not contain special character or spaces.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.
`stdout`	Default: "|" keep stdout to the R object returned by `run()`. A character string can be used to define a file path to be used as standard output. e.g: "output.txt".

`packages_search_docker` *Search for Packages in Channels using Docker*

Description

Search for Packages in Channels using Docker

Usage

```
packages_search_docker(
  packages,
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL,
  container_name = "condathis-micromamba-base",
  image_name = "luciorq/condathis-micromamba:latest"
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`container_name`	Character. Name of the Container created by Docker. Defaults to `"condathis-micromamba-base"`.
`image_name`	Character. Name of the Docker Image used, it will try to pull it automatically if internet connection is available. Defaults to `"luciorq/condathis-micromamba:latest"`.

`packages_search_native` *Search For Package in Channels*

Description

Search if Package is available in required Channels

Usage

```
packages_search_native(
  packages,
  channels = c("bioconda", "conda-forge"),
  method = c("native", "docker", "singularity"),
  platform = NULL,
  additional_channels = NULL
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`method`	Character. Beckend method to run `micromamba`, the default is "auto" running "native" with the `micromamba` binaries installed by `condathis`. Additional methods are supported for using Docker Linux Containers "docker" and Singularity Containers "singularity" as the beckends, those are especially useful if running on systems where the Conda recipes are not available for the OS or CPU architecture in place. The container-based backends leverage `dockerthis` R package.
`platform`	Character. Platform to search for `packages`. Defaults to `NULL` which will use the current platform. E.g. "linux-64", "linux-32", "osx-64", "win-64", "win-32", "noarch". Note: on Apple Silicon MacOS will use "osx-64" instead of "osx-arm64" if Rosetta 2 is available.
`additional_channels`	Character. Additional Channels to be added to the default ones.

`packages_search_singularity`

Search for packages in Channels using Singularity

Description

Search for packages in Channels using Singularity

Usage

```
packages_search_singularity(
  packages,
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL,
  sif_image_path = NULL
)
```

Arguments

`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`sif_image_path`	Character. Path to SIF image file.

`remove_env` *Remove a Conda Environment*

Description

Remove a Conda environment previously created by `create_env()`.

Usage

```
remove_env(env_name = "condathis-env", verbose = TRUE)
```

Arguments

`env_name`	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.

`run` *Run Command Line tools in a Conda environment.*

Description

This function allows the execution of command line tools within a specific Conda environment. It runs the provided cmd command in the designated Conda environment, using the Micromamba binaries managed by the `condathis` package.

Usage

```
run(
  cmd,
  ...,
  env_name = "condathis-env",
  method = "native",
  container_name = "condathis-micromamba-base",
  image_name = "luciorq/condathis-micromamba:latest",
  mount_paths = NULL,
  packages = NULL,
  channels = c("bioconda", "conda-forge"),
  additional_channels = NULL,
  sif_image_path = NULL,
  gpu_container = FALSE,
  verbose = TRUE,
  stdout = "|"
)
```

Arguments

`cmd`	Character. The main command to be executed in the Conda environment.
`...`	Additional arguments to be used in the command. These arguments will be passed directly to the command executed in the Conda environment. File paths should not contain special character or spaces.
`env_name`	Character. The name of the Conda environment where the tool will be run. Defaults to 'condathis-env'. If the specified environment does not exist, it will be created automatically using create_env() function from the `condathis` package.
`method`	Character. Beckend method to run `micromamba`, the default is "auto" running "native" with the `micromamba` binaries installed by `condathis`. Additional methods are supported for using Docker Linux Containers "docker" and Singularity Containers "singularity" as the beckends, those are especially useful if running on systems where the Conda recipes are not available for the OS or CPU architecture in place. The container-based backends leverage `dockerthis` R package.
`container_name`	Character. Name of the Container created by Docker. Defaults to `"condathis-micromamba-base"`.
`image_name`	Character. Name of the Docker Image used, it will try to pull it automatically if internet connection is available. Defaults to `"luciorq/condathis-micromamba:latest"`.
`mount_paths`	Character vector. Host paths to be mounted in container.
`packages`	Character vector. Names of the packages, and version strings if necessary, e.g. 'python=3.11'. The use of the `packages` argument assumes that env_file is not used.
`channels`	Character vector. Names of the channels to be included. By default 'c("bioconda", "conda-forge")' are used for solving dependencies.
`additional_channels`	Character. Additional Channels to be added to the default ones.
`sif_image_path`	Character. Path to SIF image file.

`gpu_container`	Logical. GPU support for Container Beckend `methods`. This argument is not necessary if running native. Default to FALSE.
`verbose`	Logical. Should command and messages be print to the console. defaults to TRUE.
`stdout`	Default: "|" keep stdout to the R object returned by `run()`. A character string can be used to define a file path to be used as standard output. e.g: "output.txt".

See Also

`install_micromamba`, `create_env`

Examples

```
## Not run:
## Run a simple command in the default Conda environment
run("ls", "-l")

## Run a command in a specific Conda environment
run("python script.py", env_name = "my-conda-env")

## Run a command with additional arguments
run("my-command", "--arg1", "--arg2=value", env_name = "my-conda-env")

## End(Not run)
```

`stop_if_not_installed` *Stop execution if* `dockerthis` *package is not installed.*

Description

Stop execution if `dockerthis` package is not installed.

Usage

```
stop_if_not_installed(pkg_name = "dockerthis", org_name = "luciorq")
```

Arguments

`pkg_name`	Character. Name of the R package to check.
`org_name`	Character. Name of the Remote organization where development version of package is hosted.

Index