Package 'condathis'

December 11, 2024

Title Run Any CLI Tool on a 'Conda' Environment

Version 0.1.0

Description Simplifies the execution of command line interface (CLI) tools within isolated and reproducible environments. It enables users to effortlessly manage 'Conda' environments, execute command line tools, handle dependencies, and ensure reproducibility in their data analysis workflows.

License MIT + file LICENSE

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

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

Depends R (>= 4.0)

Imports cli, fs, jsonlite, processx, rlang, stringr, tibble, tools, utils, withr

Suggests curl, dplyr, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Lucio Queiroz [aut, cre, cph] (<https://orcid.org/0000-0002-6090-1834>), Claudio Zanettini [aut, ctb] (<https://orcid.org/0000-0001-5043-8033>)

Maintainer Lucio Queiroz <luciorqueiroz@gmail.com>

Repository CRAN

Date/Publication 2024-12-11 16:00:01 UTC

Contents

`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"),
  additional_channels = NULL,
  platform = NULL,
  verbose = "silent",
  overwrite = FALSE
)
```

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`. This argument is **soft deprecated** as changing it don't really do anything.

`additional_channels` Character. Additional Channels to be added to the default ones.

`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 and any of the `packages` is not available for "osx-arm64".

`verbose` Character string specifying the verbosity level of the function's output. Acceptable values are:

- **"silent"**: Suppress all output from internal command-line tools. Equivalent to `FALSE`.
- **"cmd"**: Print the internal command(s) passed to the command-line tool.
- **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed.
- **"full"**: Print both the internal command(s) (`"cmd"`) and their standard output and error (`"output"`). Equivalent to `TRUE`. Logical values `FALSE` and `TRUE` are also accepted for backward compatibility but are *soft-deprecated*. Please use `"silent"` and `"full"` respectively instead.

`overwrite` Logical. Should environment always be overwritten? Defaults to `FALSE`.

Value

An object of class `list` representing the result of the command execution. Contains information about the standard output, standard error, and exit status of the command.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create a Conda environment and install the CLI `fastqc` in it.
  condathis::create_env(
    packages = "fastqc==0.12.1",
    env_name = "fastqc-env",
    verbose = "output"
  )
  #> ! Environment fastqc-env succesfully created.
})

## End(Not run)
```

env_exists *Check If Environment Already exists*

Description

This function checks whether a specified Conda environment already exists in the available environments. It returns TRUE if the environment exists and FALSE otherwise.

Usage

```
env_exists(env_name)
```

Arguments

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

Value

Boolean. TRUE if the environment exists and FALSE otherwise.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create the environment
  condathis::create_env(
    packages = "fastqc",
    env_name = "fastqc-env"
  )

  # Check if the environment exists
  condathis::env_exists("fastqc-env")
  #> [1] TRUE

  # Check for a non-existent environment
  condathis::env_exists("non-existent-env")
  #> [1] FALSE
})

## End(Not run)
```

get_env_dir *Retrieve Path To Environment*

Description

Retrieve path to where environment should be created. **Note**: It retrieves the Path even if the environment is **not** created yet.

Usage

```
get_env_dir(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'.

Value

A character string indicating the path where environments will be created.

Examples

```
condathis::with_sandbox_dir({
  # Get the default environment directory
  condathis::get_env_dir()
  #> "/path/to/condathis/envs/condathis-env"

  # Get the directory for a specific environment
  condathis::get_env_dir("my-env")
  #> "/path/to/condathis/envs/my-env"
})
```

get_install_dir *Retrieve and Create the* condathis *Data Directory*

Description

Retrieves the installation directory for the condathis package, creating it if it does not exist. This function ensures that the package data directory complies with the freedesktop's XDG Base Directory Specification. The base path can be controlled by the XDG_DATA_HOME environment variable. Additionally, on Windows, %LOCALAPPDATA% is also accepted as the base installation directory.

Usage

```
get_install_dir()
```

Details

If the directory does not exist, it will be created. On macOS, special handling is applied to avoid spaces in the path, as `micromamba run` fails if there are spaces in the path (e.g., in `~/Library/Application Support/condat` Therefore, Unix-style paths are used on macOS.

Value

A character string representing the normalized, real path to the `condathis` data directory.

Examples

```
condathis::with_sandbox_dir({
  condathis::get_install_dir()
  #> /home/username/.local/share/condathis
})
```

`get_sys_arch` *Retrieve Operating System and CPU Architecture*

Description

This function retrieves the operating system (OS) name and the CPU architecture of the current system. The output combines the OS and CPU architecture into a single string in the format `"<OS>-<Architecture>"`.

Usage

```
get_sys_arch()
```

Value

A character string indicating the operating system and CPU architecture, e.g., `"Darwin-x86_64"` or `"Linux-aarch64"`.

Examples

```
# Retrieve the system architecture
condathis::get_sys_arch()
#> [1] "Darwin-x86_64"
```

install_micromamba *Install Micromamba Binaries in the* condathis *Controlled Path*

Description

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(
  micromamba_version = "2.0.4-0",
  timeout_limit = 3600,
  download_method = "auto",
  force = FALSE
)
```

Arguments

micromamba_version	Character string specifying the version of Micromamba to download. Defaults to "2.0.4-0".
timeout_limit	Numeric value specifying the timeout limit for downloading the Micromamba binaries, in seconds. Defaults to 3600 seconds (1 hour).
download_method	Character string 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.

Details

This function checks if Micromamba is already installed in the condathis controlled path. If not, it downloads the specified version from the official GitHub releases and installs it. On Windows, it ensures the binary is downloaded correctly by setting the download mode to "wb". If the download fails, appropriate error messages are displayed.

Value

Invisibly returns the path to the installed Micromamba binary.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Install the default version of Micromamba
  condathis::install_micromamba()

  # Install a specific version of Micromamba
  condathis::install_micromamba(micromamba_version = "2.0.2-2")

  # Force reinstallation of Micromamba
  condathis::install_micromamba(force = TRUE)
})

## End(Not run)
```

---

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 = "silent"
)
```

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	Character string specifying the verbosity level of the function's output. Acceptable values are: • **"silent"**: Suppress all output from internal command-line tools. Equivalent to FALSE.

- **"cmd"**: Print the internal command(s) passed to the command-line tool.
- **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed.
- **"full"**: Print both the internal command(s) (`"cmd"`) and their standard output and error (`"output"`). Equivalent to `TRUE`. Logical values `FALSE` and `TRUE` are also accepted for backward compatibility but are *soft-deprecated*. Please use `"silent"` and `"full"` respectively instead.

Value

An object of class `list` representing the result of the command execution. Contains information about the standard output, standard error, and exit status of the command.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  condathis::create_env(
    packages = "fastqc",
    env_name = "fastqc-env"
  )
  # Install the package `python` in the `fastqc-env` environment.
  # It is not recommended to install multiple packages in the same environment,
  #  # as it defeats the purpose of isolation provided by separate environments.
  condathis::install_packages(packages = "python", env_name = "fastqc-env")
})

## End(Not run)
```

---

list_envs *List Installed Conda Environments*

Description

This function retrieves a list of Conda environments installed in the Condathis environment directory. The returned value excludes any environments unrelated to Condathis, such as the base Conda environment itself.

Usage

```
list_envs(verbose = "silent")
```

Arguments

verbose	A character string indicating the verbosity level for the command. Defaults to "silent". Options include "silent", "minimal", and "verbose". See run() for details.

Value

A character vector containing the names of installed Conda environments. If the command fails, the function returns the process exit status as a numeric value.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create environments
  condathis::create_env(
    packages = "fastqc",
    env_name = "fastqc-env"
  )
  condathis::create_env(
    packages = "python",
    env_name = "python-env"
  )

  # List environments
  condathis::list_envs()
  #> [1] "fastqc-env" "python-env"
})

## End(Not run)
```

---

list_packages *List Packages Installed in a Conda Environment*

Description

This function retrieves a list of all packages installed in the specified Conda environment. The result is returned as a tibble with detailed information about each package, including its name, version, and source details.

Usage

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

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().
verbose	Character string specifying the verbosity level of the function's output. Acceptable values are: • **"silent"**: Suppress all output from internal command-line tools. Equivalent to FALSE. • **"cmd"**: Print the internal command(s) passed to the command-line tool. • **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed. • **"full"**: Print both the internal command(s) ("cmd") and their standard output and error ("output"). Equivalent to TRUE. Logical values FALSE and TRUE are also accepted for backward compatibility but are *soft-deprecated*. Please use "silent" and "full" respectively instead.

Value

A tibble containing all the packages installed in the specified environment, with the following columns:

base_url The base URL of the package source.

build_number The build number of the package.

build_string The build string describing the package build details.

channel The channel from which the package was installed.

dist_name The distribution name of the package.

name The name of the package.

platform The platform for which the package is built.

version The version of the package.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Creates a Conda environment with the CLI `fastqc`
  condathis::create_env(
    packages = "fastqc",
    env_name = "fastqc-env"
  )
  # Lists the packages in env `fastqc-env`
  dat <- condathis::list_packages("fastqc-env")
  dim(dat)
  #> [1] 34  8
```

```
})

## End(Not run)
```

`micromamba_bin_path` *Retrieve Path to the* `micromamba` *Executable*

Description

This function returns the file path to the `micromamba` executable managed by the `condathis` package. The path is determined based on the system's operating system and architecture.

Usage

```
micromamba_bin_path()
```

Value

A character string representing the full path to the `micromamba` executable. The path differs depending on the operating system:

Windows `<install_dir>/micromamba/Library/bin/micromamba.exe`

Other OS (e.g., Linux, macOS) `<install_dir>/micromamba/bin/micromamba`

Examples

```
condathis::with_sandbox_dir({
  # Retrieve the path to where micromamba executable is searched
  micromamba_path <- condathis::micromamba_bin_path()
  print(micromamba_path)
})
```

`parse_output` *Parse the output of a Condathis command*

Description

This function processes the result of a `run()` call by parsing the specified output stream (`stdout` or `stderr`) into individual, trimmed lines.

Usage

```
parse_output(res, stream = c("stdout", "stderr"))
```

Arguments

res	A list containing the result of run(), typically including stdout and stderr as character strings.
stream	A character string specifying the data stream to parse. Must be either "stdout" or "stderr". Defaults to "stdout".

Value

A character vector where each element is a trimmed line from the specified stream.

Examples

```
# Example result object from condathis::run()
res <- list(
  stdout = "line1\nline2\nline3\n",
  stderr = "error1\nerror2\n"
)

# Parse the standard output
parse_output(res, stream = "stdout")

# Parse the standard error
parse_output(res, stream = "stderr")
```

---

remove_env *Remove a Conda Environment*

Description

Remove a Conda environment previously created by create_env().

Usage

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

Arguments

env_name	Character. Name of the Conda environment where the packages are going to be installed. Defaults to 'condathis-env'.
verbose	Character string specifying the verbosity level of the function's output. Acceptable values are: • **"silent"**: Suppress all output from internal command-line tools. Equivalent to FALSE. • **"cmd"**: Print the internal command(s) passed to the command-line tool.

- **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed.
- **"full"**: Print both the internal command(s) (`"cmd"`) and their standard output and error (`"output"`). Equivalent to `TRUE`. Logical values `FALSE` and `TRUE` are also accepted for backward compatibility but are *soft-deprecated*. Please use `"silent"` and `"full"` respectively instead.

Value

An object of class `list` representing the result of the command execution. Contains information about the standard output, standard error, and exit status of the command.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  condathis::create_env(
    packages = "fastqc",
    env_name = "fastqc-env"
  )
  condathis::remove_env(env_name = "fastqc-env")
})

## End(Not run)
```

---

run *Run Command-Line Tools in a Conda Environment*

Description

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

Usage

```
run(
  cmd,
  ...,
  env_name = "condathis-env",
  method = c("native", "auto"),
  verbose = c("silent", "cmd", "output", "full", FALSE, TRUE),
  error = c("cancel", "continue"),
  stdout = "|",
  stderr = "|"
)
```

Arguments

cmd	Character. The main command to be executed in the Conda environment.
...	Additional arguments to be passed to the command. These arguments will be passed directly to the command executed in the Conda environment. File paths should not contain special characters 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().
method	Character string. The method to use for running the command. Options are "native", "auto". Defaults to "native". This argument is **soft deprecated** as changing it don't really do anything.
verbose	Character string specifying the verbosity level of the function's output. Acceptable values are: • **"silent"**: Suppress all output from internal command-line tools. Equivalent to FALSE. • **"cmd"**: Print the internal command(s) passed to the command-line tool. • **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed. • **"full"**: Print both the internal command(s) ("cmd") and their standard output and error ("output"). Equivalent to TRUE. Logical values FALSE and TRUE are also accepted for backward compatibility but are *soft-deprecated*. Please use "silent" and "full" respectively instead.
error	Character string. How to handle errors. Options are "cancel" or "continue". Defaults to "cancel".
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".
stderr	Default: "|" keep stderr to the R object returned by run(). A character string can be used to define a file path to be used as standard error. e.g: "error.txt".

Details

The run() function provides a flexible way to execute command-line tools within Conda environments. This is particularly useful for reproducible research and ensuring that specific versions of tools are used.

Value

An object of class list representing the result of the command execution. Contains information about the standard output, standard error, and exit status of the command.

See Also

install_micromamba, create_env

Examples

```
## Not run:
condathis::with_sandbox_dir({
  ## Create env
  create_env("samtools", env_name = "samtools-env")

  ## Run a command in a specific Conda environment
 samtools_res <- run("samtools", "view", fs::path_package("condathis", "extdata", "example.bam"),
    env_name = "samtools-env"
  )
  parse_output(samtools_res)[1]
  #> [1] "SOLEXA-1GA-1_6_FC20ET7:6:92:473:531\t0\tchr1\t10156..."
})

## End(Not run)
```

`run_bin` *Run a Binary from a Conda Environment Without Environment Activation*

Description

Executes a binary command from a specified Conda environment without activating the environment or using its environment variables. This function temporarily clears Conda and Mamba-related environment variables to prevent interference, ensuring that the command runs in a clean environment. Usually this is not what the user wants as this mode of execution does not load environment variables and scripts defined in the environment `activate.d`, check `run()` for the stable function to use.

Usage

```
run_bin(
  cmd,
  ...,
  env_name = "condathis-env",
  verbose = "silent",
  error = c("cancel", "continue"),
  stdout = "|",
  stderr = "|"
)
```

Arguments

`cmd`	Character. The main command to be executed in the Conda environment.
`...`	Additional arguments to be passed to the command. These arguments will be passed directly to the command executed in the Conda environment. File paths should not contain special characters 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().

verbose — Character string specifying the verbosity level of the function's output. Acceptable values are:

- **"silent"**: Suppress all output from internal command-line tools. Equivalent to FALSE.
- **"cmd"**: Print the internal command(s) passed to the command-line tool.
- **"output"**: Print the standard output and error from the command-line tool to the screen. Note that the order of the standard output and error lines may not be correct, as standard output is typically buffered. If the standard output and/or error is redirected to a file or they are ignored, they will not be echoed.
- **"full"**: Print both the internal command(s) ("cmd") and their standard output and error ("output"). Equivalent to TRUE. Logical values FALSE and TRUE are also accepted for backward compatibility but are *soft-deprecated*. Please use "silent" and "full" respectively instead.

error — Character string. How to handle errors. Options are "cancel" or "continue". Defaults to "cancel".

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".

stderr — Default: "|" keep stderr to the R object returned by run(). A character string can be used to define a file path to be used as standard error. e.g: "error.txt".

Value

An object of class list representing the result of the command execution. Contains information about the standard output, standard error, and exit status of the command.

Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Example assumes that 'my-env' exists and contains 'python'
  # Run 'python' with a script in 'my-env' environment
  condathis::run_bin(
    "python", "-c", "import sys; print(sys.version)",
    env_name = "my-env",
    verbose = "output"
  )

  # Run 'ls' command with additional arguments
  condathis::run_bin("ls", "-la", env_name = "my-env")
})

## End(Not run)
```

`with_sandbox_dir` *Execute Code in a Temporary Directory*

Description

Runs user-defined code inside a temporary directory, setting up a temporary working environment. This function is intended for use in examples and tests and ensures that no data is written to the user's file space. Environment variables such as `HOME`, `APPDATA`, `R_USER_DATA_DIR`, `XDG_DATA_HOME`, `LOCALAPPDATA`, and `USERPROFILE` are redirected to temporary directories.

Usage

```
with_sandbox_dir(code, .local_envir = base::parent.frame())
```

Arguments

`code`	expression An expression containing the user-defined code to be executed in the temporary environment.
`.local_envir`	environment The environment to use for scoping.

Details

This function is not designed for direct use by package users. It is primarily used to create an isolated environment during examples and tests. The temporary directories are created automatically and cleaned up after execution.

Value

Returns `NULL` invisibly.

Examples

```
condathis::with_sandbox_dir(print(fs::path_home()))
condathis::with_sandbox_dir(print(tools::R_user_dir("condathis")))
```

Index