# Package 'condathis'

June 20, 2026

**Title** Run Any CLI Tool on a 'Conda' Environment

**Version** 0.1.4

**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, https://luciorq.github.io/condathis/

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

**Depends** R (>= 4.3)

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

**Suggests** curl, knitr, quarto, testthat (>= 3.0.0)

**VignetteBuilder** quarto

**Config/Needs/dev** devtools, pkgload, remotes, rcmdcheck, covr, testthat, usethis, pak, knitr, quarto, pkgdown, roxygen2, roxygen2md, lintr, styler

**Config/Needs/website** quarto

**Config/roxygen2/markdown** TRUE

**Config/roxygen2/version** 8.0.0

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Config/testthat/start-first** create_env, rethrow_error, parse_match_spec, run_verbose_levels

**Encoding** UTF-8

**NeedsCompilation** no

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

**Maintainer** Lucio Queiroz <luciorqueiroz@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-06-19 22:10:02 UTC

# Contents



---

`clean_cache` *Clean Conda cache*

---

## Description

Removes cached packages and archives from the `condathis` Conda root. Also removes files from the package cache directory returned by `tools::R_user_dir(package = "condathis", which = "cache")`.

## Usage

```
clean_cache(verbose = c("output", "silent", "cmd", "spinner", "full"))
```

## Arguments

| | |
|---|---|
| `verbose` | Character string controlling console output. Supported values are `"output"`, `"silent"`, `"cmd"`, `"spinner"`, and `"full"`. Defaults to `"output"`. |

## Details

Package files still referenced by existing environments may not be removed. To maximize cleanup, remove environments first with `list_envs()` and `remove_env()`.

## Value

A process result list (from `processx::run()`) with command output, error output, exit status, and timeout information.

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  clean_cache(verbose = "output")
})

## End(Not run)
```

---

`create_env` *Create a Conda environment*

---

## Description

Creates a Conda environment managed by `condathis` and installs dependencies from package specs or from an environment file.

## Usage

```
create_env(
  packages = NULL,
  env_file = NULL,
  env_name = "condathis-env",
  channels = c("conda-forge", "bioconda"),
  method = c("native", "auto"),
  channel_priority = c("disabled", "strict", "flexible"),
  additional_channels = NULL,
  platform = NULL,
  verbose = c("output", "silent", "cmd", "spinner", "full"),
  overwrite = FALSE
)
```

## Arguments

| | |
|---|---|
| `packages` | Character vector of package MatchSpec strings. Examples: `"python=3.13"`, `"bioconda::fastqc==0.12.1"`. Defaults to `NULL`. |
| `env_file` | Character string with the path to an environment YAML file. Defaults to `NULL`. When provided, it is passed to `micromamba create -f`. |

`env_name` Character string with the target environment name. Defaults to `"condathis-env"`.

`channels` Character vector with channel names used for dependency resolution. Defaults to `c("conda-forge", "bioconda")`.

`method` Character string with the backend execution strategy. Supported values are `"native"` and `"auto"`. Defaults to `"native"`. This argument is soft-deprecated and currently does not change behavior.

`channel_priority` Character string with channel priority mode. Supported values are `"disabled"`, `"strict"`, and `"flexible"`. Defaults to `"disabled"`.

`additional_channels` Character vector of additional channels appended to `channels`. Defaults to `NULL`.

`platform` Character string with the platform used for dependency solving (for example, `"linux-64"`, `"osx-64"`, `"osx-arm64"`, `"win-64"`, `"noarch"`). Defaults to `NULL`. On Apple Silicon, `condathis` may fall back to `"osx-64"` when Rosetta 2 is available and packages are not available for `"osx-arm64"`.

`verbose` Character string controlling console output. Supported values are `"output"`, `"silent"`, `"cmd"`, `"spinner"`, and `"full"`. Defaults to `"output"`. Logical values are accepted for backward compatibility: `TRUE` maps to `"output"` and `FALSE` maps to `"silent"`.

`overwrite` Logical value that controls whether an existing environment should always be recreated. Defaults to `FALSE`.

## Value

A process result list (from `processx::run()`) with command output, error output, exit status, and timeout information.

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create a Conda environment and install the CLI `fastqc` in it.
  # Explicitly using the channel `bioconda` and version `0.12.1`.
  condathis::create_env(
    packages = "bioconda::fastqc==0.12.1",
    env_name = "fastqc-env",
    verbose = "output"
  )
})

## End(Not run)
```

env_exists *Check whether a Conda environment exists*

## Description

Checks whether an environment name is present in the environments managed by condathis.

## Usage

```
env_exists(env_name, verbose = "silent")
```

## Arguments

| | |
|---|---|
| env_name | Character string with the environment name to check. |
| verbose | Character string controlling console output passed to list_envs(). Defaults to "silent". |

## Value

TRUE when the environment exists and FALSE otherwise.

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create the environment
  condathis::create_env(
    packages = "bioconda::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 *Get an environment directory path*

### Description

Returns the absolute path where an environment is expected under the condathis installation root. The path is returned even if the environment has not been created yet.

### Usage

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

### Arguments

| | |
|---|---|
| env_name | Character string with the environment name. Defaults to "condathis-env". |

### Value

A character string with the expected environment directory path.

### 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 *Get the* condathis *data directory*

### Description

Returns the data directory used by condathis, creating it when needed. The base path follows the platform-specific user data directory rules used by tools::R_user_dir().

### Usage

```
get_install_dir()
```

## Details

On macOS, `condathis` uses a path without spaces when possible because `micromamba run` can fail on paths that contain spaces.

## Value

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

## Examples

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

---

# `get_sys_arch` *Get operating system and CPU architecture*

---

## Description

Returns the current operating system and CPU architecture as a single string in the format `"<OS>-<Architecture>"`.

## Usage

```
get_sys_arch()
```

## Value

A character string such as `"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 managed condathis path*

## Description

Downloads and installs the `micromamba` executable used by `condathis`.

## Usage

```
install_micromamba(
  micromamba_version = "2.8.1-0",
  timeout_limit = 3600,
  download_method = "auto",
  force = FALSE,
  verbose = c("output", "silent", "cmd", "spinner", "full")
)
```

## Arguments

| | |
|---|---|
| `micromamba_version` | Character string with the micromamba version. Defaults to `"2.8.1-0"`. |
| `timeout_limit` | Numeric download timeout in seconds. Defaults to `3600`. |
| `download_method` | Character string passed as the download method when `utils::download.file()` is used. Defaults to `"auto"`. |
| `force` | Logical value that controls forced reinstallation. Defaults to `FALSE`. |
| `verbose` | Character string controlling console output. Supported values are `"output"`, `"silent"`, `"cmd"`, `"spinner"`, and `"full"`. Defaults to `"output"`. |

## Details

Download mirrors are tried in order until one succeeds. When system `tar` and `bzip2` are available, a compressed archive may be used first. Otherwise, or when extraction fails, a standalone binary is downloaded.

## Value

The installed micromamba binary path, invisibly.

## 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 Conda environment*

---

## Description

Installs packages into an existing `condathis` environment. If the target environment does not exist, it is created first.

## Usage

```
install_packages(
  packages,
  env_name = "condathis-env",
  channels = c("conda-forge", "bioconda"),
  channel_priority = c("disabled", "strict", "flexible"),
  additional_channels = NULL,
  verbose = c("output", "silent", "cmd", "spinner", "full")
)
```

## Arguments

| | |
|---|---|
| `packages` | Character vector of package MatchSpec strings to install. |
| `env_name` | Character string with the target environment name. Defaults to `"condathis-env"`. |
| `channels` | Character vector with channel names used for dependency resolution. Defaults to `c("conda-forge", "bioconda")`. |
| `channel_priority` | Character string with channel priority mode. Supported values are `"disabled"`, `"strict"`, and `"flexible"`. Defaults to `"disabled"`. |
| `additional_channels` | Character vector of additional channels appended to `channels`. Defaults to `NULL`. |
| `verbose` | Character string controlling console output. Supported values are `"output"`, `"silent"`, `"cmd"`, `"spinner"`, and `"full"`. Defaults to `"output"`. |

## Value

A process result list (from `processx::run()`) with command output, error output, exit status, and timeout information.

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  condathis::create_env(
    packages = "bioconda::fastqc",
    env_name = "fastqc-env"
  )
  # Install the package `python` in the `fastqc-env` environment.
  # NOTE: 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 Conda environments managed by condathis*

---

## Description

Returns environment names located under the `condathis` installation root. Environments not managed by `condathis` are excluded.

## Usage

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

## Arguments

| | |
|---|---|
| `verbose` | Character string controlling console output. Defaults to `"silent"`. |

## Value

A character vector of environment names. If the command fails, returns the process exit status as a numeric value.

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  # Create environments
  condathis::create_env(
    packages = "bioconda::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 in a Conda environment*

---

## Description

Returns package metadata for a Conda environment as a tibble.

## Usage

```
list_packages(
  env_name = "condathis-env",
  verbose = c("output", "silent", "cmd", "spinner", "full")
)
```

## Arguments

| | |
|---|---|
| `env_name` | Character string with the target environment name. Defaults to `"condathis-env"`. |
| `verbose` | Character string controlling console output. Supported values are `"output"`, `"silent"`, `"cmd"`, `"spinner"`, and `"full"`. Defaults to `"output"`. |

## Value

A data frame (`tibble`) with installed packages and the 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 = "bioconda::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` *Get the managed* `micromamba` *binary path*

---

## Description

Returns the expected path to the `micromamba` executable managed by `condathis` for the current operating system.

## Usage

```
micromamba_bin_path()
```

## Value

A character string with the full executable path. On Windows this points to `micromamba.exe` under `Library/bin`. On other platforms this points to `micromamba` under `bin`.

## Examples

```
condathis::with_sandbox_dir({
  # Retrieve the path used by condathis for micromamba
  micromamba_path <- condathis::micromamba_bin_path()
  print(micromamba_path)
})
```

`parse_output` *Parse command output text*

## Description

Parses output from a `run()` result into trimmed text lines.

## Usage

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

## Arguments

| | |
|---|---|
| `res` | Either a process result list (with `stdout` and/or `stderr`) or a character vector when `stream = "plain"`. |
| `stream` | Character string selecting the output source. Supported values are `"stdout"`, `"stderr"`, `"both"`, and `"plain"`. Defaults to `"stdout"`. |

## Value

A character vector with one trimmed line per element.

## 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")

# Merge both
parse_output(res, stream = "both")

# Parse plain text
plain_text <- "This is line one.\nThis is line two.\nThis is line three."
parse_output(plain_text, stream = "plain")
```

remove_env *Remove a Conda environment*

## Description

Removes an environment managed by condathis.

## Usage

```
remove_env(
  env_name = "condathis-env",
  verbose = c("silent", "cmd", "output", "spinner", "full")
)
```

## Arguments

| | |
|---|---|
| env_name | Character string with the environment name to remove. Defaults to "condathis-env". |
| verbose | Character string controlling console output. Supported values are "silent", "cmd", "output", "spinner", and "full". Defaults to "silent". |

## Value

A process result list (from processx::run()) with command output, error output, exit status, and timeout information.

## Examples

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

## End(Not run)
```

run *Run a command inside a Conda environment*

## Description

Executes a command in a Conda environment managed by condathis. The command is run through micromamba run using the internal Conda root.

## Usage

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

## Arguments

| | |
|---|---|
| cmd | Character string with the command to execute. |
| ... | Additional unnamed command arguments passed to cmd. |
| env_name | Character string with the target environment name. Defaults to "condathis-env". If the default environment does not exist, it is created automatically. |
| method | Character string with the backend execution strategy. Supported values are "native" and "auto". Defaults to "native". This argument is soft-deprecated and currently does not change behavior. |
| verbose | Character string controlling console output. Supported values are "output", "silent", "cmd", "spinner", and "full". Defaults to "output". Logical values are accepted for backward compatibility: TRUE maps to "output" and FALSE maps to "silent". |
| error | Character string that controls error behavior. Supported values are "cancel" and "continue". Defaults to "cancel". |
| stdout | Standard output target. Defaults to "\|" (capture stdout in the returned object). Provide a file path to redirect stdout to a file. |
| stderr | Standard error target. Defaults to "\|" (capture stderr in the returned object). Provide a file path to redirect stderr to a file. |
| stdin | Standard input source. Defaults to NULL (no stdin stream). Provide a file path to use file contents as stdin. |

## Details

This function is the main execution entry point in `condathis`. Use it to run CLI tools with reproducible dependencies isolated in Conda environments.

## Value

A process result list (from `processx::run()`) with command output, error output, exit status, and timeout information.

## See Also

`install_micromamba`, `create_env`

## Examples

```
## Not run:
condathis::with_sandbox_dir({
  ## Create env
  create_env("bioconda::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",
    verbose = "silent"
  )
  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 without environment activation*

---

## Description

Executes a binary using files from a target Conda environment, but without running environment activation scripts. This is a lower-level execution mode than `run()`.

## Usage

```
run_bin(
  cmd,
  ...,
  env_name = "condathis-env",
```

```
  verbose = c("output", "silent", "cmd", "spinner", "full"),
  error = c("cancel", "continue"),
  stdout = "|",
  stderr = "|",
  stdin = NULL
)
```

## Arguments

| | |
|---|---|
| cmd | Character string with the command to execute. |
| ... | Additional unnamed command arguments passed to cmd. |
| env_name | Character string with the target environment name. Defaults to "condathis-env". |
| verbose | Character string controlling console output. Supported values are "output", "silent", "cmd", "spinner", and "full". Defaults to "output". |
| error | Character string that controls error behavior. Supported values are "cancel" and "continue". Defaults to "cancel". |
| stdout | Standard output target. Defaults to "\|" (capture stdout in the returned object). Provide a file path to redirect stdout to a file. |
| stderr | Standard error target. Defaults to "\|" (capture stderr in the returned object). Provide a file path to redirect stderr to a file. |
| stdin | Standard input source. Defaults to NULL (no stdin stream). Provide a file path to use file contents as stdin. |

## Value

A process result list (from processx::run()) with command output, error output, exit status, and timeout information.

## 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"
  )

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

## End(Not run)
```

`with_sandbox_dir` *Execute code in an isolated temporary environment*

## Description

Evaluates code with temporary home, data, and cache directories. This is mainly intended for examples and tests so no user files are written.

## Usage

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

## Arguments

| | |
|---|---|
| `code` | Expression to evaluate in the sandboxed environment. |
| `.local_envir` | Environment used for local scoping and evaluation. Defaults to `parent.frame()`. |

## Value

`NULL`, invisibly.

## Examples

```
## Not run:
condathis::with_sandbox_dir(print(fs::path_home()))
condathis::with_sandbox_dir(print(tools::R_user_dir("condathis")))

## End(Not run)
```

# Index