Package 'whirl'

January 21, 2026

Title Log Execution of Scripts

Version 0.3.2

Description Logging of scripts suitable for clinical trials using 'Quarto' to create nice human readable logs. 'whirl' enables execution of scripts in batch, while simultaneously creating logs for the execution of each script, and providing an overview summary log of the entire batch execution.

License Apache License (>= 2)

URL https://novonordisk-opensource.github.io/whirl/, https://github.com/novonordisk-opensource/whirl

BugReports https://github.com/NovoNordisk-OpenSource/whirl/issues

Depends R (>= 4.1)

Imports callr, cli, dplyr, jsonlite, kableExtra, knitr, purrr, quarto, R6 (>= 2.4.0), renv, reticulate (>= 1.23), rlang, sessioninfo, stringr, tibble, unglue, utils, withr, yaml, zephyr (>= 0.1.1)

Suggests ggplot2, rstudioapi, testthat (>= 3.0.0), usethis

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

SystemRequirements Quarto command line tool (<https://github.com/quarto-dev/quarto-cli>).

NeedsCompilation no

Author Aksel Thomsen [aut, cre], Lovemore Gakava [aut], Cervan Girard [aut], Kristian Troejelsgaard [aut], Steffen Falgreen Larsen [aut], Vladimir Obucina [aut], Michael Svingel [aut],

Skander Mulder [aut],
Oliver Lundsgaard [aut],
Novo Nordisk A/S [cph]

Maintainer Aksel Thomsen <oath@novonordisk.com>

Repository CRAN

Date/Publication 2026-01-21 11:00:02 UTC

Contents

`custom_logging` *Helper function to log custom messages*

Description

Useful for e.g. read and write operations on databases etc. that are not automatically captured.

Usage

```
log_read(file, log = Sys.getenv("WHIRL_LOG_MSG"))

log_write(file, log = Sys.getenv("WHIRL_LOG_MSG"))

log_delete(file, log = Sys.getenv("WHIRL_LOG_MSG"))
```

Arguments

`file`	`character()` description of the file that was read, written or deleted.
`log`	`character()` path to the log file.

Details

The default environment variable `WHIRL_LOG_MSG` is set in the session used to log scripts, and input is automatically captured in the resulting log.

If run outside of whirl, meaning when the above environment variable is unset, the operations are streamed to `stdout()`. By default the console.

Examples

```
# Stream logs to console since `WHIRL_LOG_MSG` is not set:
log_read("my/folder/input.txt")
log_write("my/folder/output.txt")
log_delete("my/folder/old_output.txt")
```

---

run *Execute single or multiple R, R Markdown, and Quarto scripts*

Description

Executes and logs the execution of the scripts. Logs for each script are stored in the same folder as the script.

The way the execution is logged is configurable through several options for e.g. the verbosity of the logs. See whirl-options on how to configure these.

Usage

```
run(
  input = "_whirl.yml",
  steps = NULL,
  summary_file = "summary.html",
  n_workers = zephyr::get_option("n_workers", "whirl"),
  check_renv = zephyr::get_option("check_renv", "whirl"),
  track_files = zephyr::get_option("track_files", "whirl"),
  out_formats = zephyr::get_option("out_formats", "whirl"),
  log_dir = zephyr::get_option("log_dir", "whirl"),
  with_options = zephyr::get_option("with_options", "whirl")
)
```

Arguments

input	A character vector of file path(s) to R, R Markdown, Quarto scripts, or files in a folder using regular expression, or to to a whirl config file. The input can also be structured in a list where each element will be executed sequentially, while scripts within each element can be executed in parallel.
steps	An optional argument that can be used if only certain steps within a config files (or list) is to be executed. Should be equivalent to the names of the steps found in the config file. If kept as NULL (default) then all steps listed in the config file will be executed.
summary_file	A character string specifying the file path where the summary log will be stored.
n_workers	Number of simultaneous workers used in the run function. A maximum of 128 workers is allowed.. Default: 1.
check_renv	Should the projects renv status be checked?. Default: FALSE.

track_files	Should files read and written be tracked? Currently only supported on Linux.. Default: FALSE.
out_formats	Which log format(s) to produce. Possibilities are html, json, and markdown formats: gfm, commonmark, and markua.. Default: "html".
log_dir	The output directory of the log files. Default is the folder of the executed script. log_dir can be a path as a character or it can be a function that takes the script path as input and returns the log directory. For more information see the examples of run() or vignette('whirl').. Default: function (x) dirname(x).
with_options	List of options to set in the child sessions executing the scripts.. Default: list().

Value

A tibble containing the execution results for all the scripts.

Examples

```
# Copy example scripts:
file.copy(
  from = system.file("examples", c("success.R", "warning.R", "error.R"),
    package = "whirl"
  ),
  to = tempdir()
)

# Run a single script and create log:
run(file.path(tempdir(), "success.R"))

# Run several scripts in parallel on up to 2 workers:
run(
  input = file.path(tempdir(), c("success.R", "warning.R", "error.R")),
  n_workers = 2
)

# Run several scripts in two steps by providing them as list elements:
run(
  list(
    file.path(tempdir(), c("success.R", "warning.R")),
    file.path(tempdir(), "error.R")
  )
)

# Re-directing the logs to a sub-folder by utilizing the log_dir argument in
# run(). This will require that the sub-folder exists.

# Specifying the path using a manually defined character
run(file.path(tempdir(), "success.R"), log_dir = tempdir())

# Specifying the path with a generic function that can handle the scripts
# individually.
run(
  input = file.path(tempdir(), "success.R"),
```

```
  log_dir = function(x) {paste0(dirname(x), "/logs")}
)
```

`use_biocompute` *Use whirl to create biocompute logs*

Description

Utility function to setup execution with whirl in your project suitable for creating biocompute logs with `write_biocompute()`:

1. Creates configuration file (default `_whirl.yml`) with default values for the `biocompute` metadata.
2. Updates `.gitignore` to not include log files

See `vignette("whirl")` for how to specify paths inside the configuration file.

Usage

```
use_biocompute(
  config_file = "_whirl.yml",
  parametrics_file = "_parametrics.yml"
)
```

Arguments

`config_file`	Path to the whirl config file, relative to the project
`parametrics_file`	Path to the biocompute parametrics file, relative to the project

---

`use_whirl` *Use whirl*

Description

Utility function to setup execution with whirl in your project:

1. Creates configuration file (default `_whirl.yml`)
2. Updates `.gitignore` to not include log files

See `vignette("whirl")` for how to specify paths inside the configuration file.

Usage

```
use_whirl(config_file = "_whirl.yml")
```

Arguments

`config_file`	Path to the whirl config file, relative to the project

`whirl-options` *Options for whirl*

Description

verbosity_level:

Verbosity level for functions in whirl. See zephyr::verbosity_level for details.

- Default: `NA_character_`
- Option: `whirl.verbosity_level`
- Environment: `R_WHIRL_VERBOSITY_LEVEL`

out_formats:

Which log format(s) to produce. Possibilities are `html`, `json`, and markdown formats: `gfm`, `commonmark`, and `markua`.

- Default: `"html"`
- Option: `whirl.out_formats`
- Environment: `R_WHIRL_OUT_FORMATS`

track_files:

Should files read and written be tracked? Currently only supported on Linux.

- Default: `FALSE`
- Option: `whirl.track_files`
- Environment: `R_WHIRL_TRACK_FILES`

check_renv:

Should the projects renv status be checked?

- Default: `FALSE`
- Option: `whirl.check_renv`
- Environment: `R_WHIRL_CHECK_RENV`

track_files_discards:

List of file naming patterns not be tracked when track_files = TRUE

- Default: `c("^/lib", "^/etc", "^/lib64", "^/usr", "^/var", "^/opt", "^/sys", "^/proc", "^/tmp", "^/null", "^/urandom", "^/.cache")`
- Option: `whirl.track_files_discards`
- Environment: `R_WHIRL_TRACK_FILES_DISCARDS`

track_files_keep:

List of file naming patterns always to be tracked when track_files = TRUE

- Default: `NULL`
- Option: `whirl.track_files_keep`
- Environment: `R_WHIRL_TRACK_FILES_KEEP`

approved_packages:
List of approved R packages and their version in the format: {name}@{version}

- Default: `NULL`
- Option: `whirl.approved_packages`
- Environment: `R_WHIRL_APPROVED_PACKAGES`

approved_python_packages:
List of approved Python packages and their version in the format: {name}@{version}

- Default: `NULL`
- Option: `whirl.approved_python_packages`
- Environment: `R_WHIRL_APPROVED_PYTHON_PACKAGES`

n_workers:
Number of simultaneous workers used in the run function. A maximum of 128 workers is allowed.

- Default: `1`
- Option: `whirl.n_workers`
- Environment: `R_WHIRL_N_WORKERS`

log_dir:
The output directory of the log files. Default is the folder of the executed script. log_dir can be a path as a character or it can be a function that takes the script path as input and returns the log directory. For more information see the examples of `run()` or `vignette('whirl')`.

- Default: `function (x) dirname(x)`
- Option: `whirl.log_dir`
- Environment: `R_WHIRL_LOG_DIR`

execute_dir:
The working directory of the process executing each script. Default us to execute R files from the working directory when calling `run()` and all other functions from the directory of the script. To change provide a character path (used for all scripts) or a function that takes the script as input and returns the execution directory.

- Default: `NULL`
- Option: `whirl.execute_dir`
- Environment: `R_WHIRL_EXECUTE_DIR`

wait_timeout:
Timeout for waiting for the R process from callr::r_session to start, in milliseconds.

- Default: `9000`
- Option: `whirl.wait_timeout`
- Environment: `R_WHIRL_WAIT_TIMEOUT`

environment_secrets:
Secret environment variable patterns. Any variables matching will not be included in the logs.

- Default: `c("BASH_FUNC", "_SSL_CERT", "_KEY", "_PAT", "_TOKEN")`
- Option: `whirl.environment_secrets`

- Environment: R_WHIRL_ENVIRONMENT_SECRETS

with_options:
List of options to set in the child sessions executing the scripts.

- Default: list()
- Option: whirl.with_options
- Environment: R_WHIRL_WITH_OPTIONS

---

write_biocompute *Create biocompute logs*

Description

BioCompute is a standard for logs of programs for for Bioinformatics Computational Analyses.

The BioCompute object is a json log that can be created based on the output of run().

Usage

```
write_biocompute(queue = run("_whirl.yml"), path = "bco.json", ...)
```

Arguments

queue	Result from run().
path	A character string specifying the file path to write BioCompute log to.
...	Additional arguments parsed to jsonlite::write_json(). Note always uses auto_unbox = TRUE.

Details

The object consists of the following domains:

- **Specifications**:
  - *spec_version*: Version of BioCompute used ('https://w3id.org/biocompute/1.3.0/")
  - *object_id*: Unique project id
  - *type*: Your project type
  - *etag*: Your etag id from the BioCompute Object Portal
- Provenance Domain
  - This is used to track the history of the BCO. Review and signatures go here.
- Usability Domain
  - This is used to improve searchability by allowing a free-text description of the BCO.
  - Provide external document.
- Extension Domain
  - This is used to add any additional structured information that is not directly covered by the BCO.

- Description Domain
  - Contains a structured field for the description of external references, the pipeline steps, and the relationship of I/O objects.
  - Provide external document.
  - **Note**: Use of `keywords` and `External_Reference` entries are not yet implemented. To use fill out the entries manually after creating the BioCompute object.‘
- Execution Domain
  - Contains fields for the execution of the BCO.
  - **Note**: Use of `external_data_endpoints` not implemented. Fill out manually afterwards if needed.
- Parametric Domain
  - Represents the list of parameters customizing the computational flow which can affect the output of the calculations.
- IO Domain
  - Represents the list of global input and output files created by the computational workflow.
- Error Domain
  - Defines the empirical and algorithmic limits and error sources of the BCO.
  - **Note**: Use of this domain is not clearly defined. It is therefore always left empty in the current implementation. If you want to add content do so manually after creating the BCO.

See the BioCompute Object Portal and the BioCompute Objects Wiki for more information.

Value

(`invisible`) `list` of the biocompute domains and their content.

Index