Package: nfcore.utils (via r-universe)

June 23, 2026

Version 0.0.2

Date 2026-03-30

Title Utilities for Nf-Core Modules

Depends R (>= 4.4.0),

Imports utils

Description Provides utility functions to facilitate the use of R within nf-core modules. The package helps parse Nextflow inputs and perform validation checks to ensure correct parameter handling and reproducible execution.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Suggests testthat (>= 3.0.0), rmarkdown, BiocStyle, knitr, withr, yaml

Config/testthat/edition 3

BugReports https://github.com/nf-core/r-nf-core-utils/issues

URL https://github.com/nf-core/r-nf-core-utils

Repository https://nf-core.r-universe.dev

Date/Publication 2026-06-08 13:45:19 UTC

RemoteUrl https://github.com/nf-core/r-nf-core-utils

RemoteRef HEAD

RemoteSha 6c13765fed25d54b05d0e0fd6ba1764c5442c8f3

Contents

`nfcore.utils-package` *The nf-core utils R package for Nextflow processes*

Description

This package aims to ease the link between Nextflow inputs and outputs to the R variable needed in the script.

Functions

Below are listed some of the main functions available:

`process_inputs()`: Given a list of expected options with their default this function will parse the arguments given and validate their value based on the expected rules.

`process_end()`: Create a `versions.yml` and `R_sessionInfo.log` file in the directory provided. The version file will be populated with the R version, the version of nfcore.utils and the version of the additional packages given.

Author(s)

Maintainer: Louis Le Nezet <louislenezet@gmail.com> (ORCID) [copyright holder]

Authors:

- nf-core Community

See Also

Useful links:

- https://github.com/nf-core/r-nf-core-utils
- Report bugs at https://github.com/nf-core/r-nf-core-utils/issues

Examples

```
library(nfcore.utils)
td <- withr::local_tempdir()
test_file_path <- file.path(td, "test_file.txt")
file.create(test_file_path)
options <- list(
  input_file = test_file_path,
  output_file = "prefix",
  threshold = 0.5
)
args <- c("--threshold 0.7")
processed_options <- process_inputs(
  options, args,
  keys_to_nullify = c("input_file", "output_file"),
  expected_files = c("input_file"),
  expected_double = c("threshold"),
  required_opts = c("input_file", "threshold")
)
## Not run:
process_end(
  packages = list(
    "r-stats" = "stats"
  ),
  task_name = 'MY_PROCESS'
)

## End(Not run)
```

---

create_log_session_info

Log R session info

Description

This function logs the R session info to a file named `R_sessionInfo.log` in the specified output directory.

Usage

```
create_log_session_info(out_dir = ".")
```

Arguments

`out_dir`	Output directory where the R session info log will be written to. Default is the current directory.

Value

R session info log

Examples

```
td <- withr::local_tempdir()
create_log_session_info(td)
```

`create_versions_yml` *Create versions.yml file*

Description

This function allows to automatically create the versions.yml file used to store the packages used. `r-base` as well as `r-nfcore.utils` versions will automatically be added.

Usage

```
create_versions_yml(packages, task_name, out_dir = ".")
```

Arguments

`packages`	Named list of packages to add to the versions.yml. The items names should be the conda package name, while the items value should be the package name used in R.
`task_name`	Name of the nextflow process. Typically `${task.process}`
`out_dir`	Output directory where the versions.yml will be written to. Default is the current directory.

Value

versions.yml file

Examples

```
td <- withr::local_tempdir()
create_versions_yml(list("r-stats" = "stats"), "MY_PROCESS", td)
```

`is_null_or_invalid` *Check if is* `NULL` *or invalid*

Description

Check if is `NULL` or invalid

Usage

```
is_null_or_invalid(x, variable_name = NULL)
```

Arguments

x	String to check
variable_name	Name of the variable to print if error occurs

Value

TRUE if is null, ERROR if invalid and FALSE otherwise

Examples

```
is_null_or_invalid(NULL) # TRUE
try(is_null_or_invalid(" ")) # ERROR
is_null_or_invalid("Hello World !") # FALSE
```

---

is_valid_string *Check for Non-Empty, Non-Whitespace String*

Description

This function checks if the input is non-NULL and contains more than just whitespace. It returns TRUE if the input is a non-empty, non-whitespace string, and FALSE otherwise.

Usage

```
is_valid_string(input)
```

Arguments

input	A variable to check.

Value

A logical value: TRUE if the input is a valid, non-empty, non-whitespace string; FALSE otherwise.

Examples

```
is_valid_string("Hello World") # Returns TRUE
is_valid_string(" Hello World ") # Returns TRUE
is_valid_string("   ")         # Returns FALSE
is_valid_string(NULL)          # Returns FALSE
```

nullify *Turn "null" or empty strings into actual NULL*

Description

Turn "null" or empty strings into actual NULL

Usage

```
nullify(x)
```

Arguments

x	Input option

Value

NULL or x

Examples

```
nullify("null") # Returns NULL
nullify("NULL") # Returns NULL
nullify("   ")  # Returns NULL
nullify("")     # Returns NULL
nullify(NULL)   # Returns NULL
nullify(NA)     # Returns NULL
nullify("Hello") # Returns "Hello"
```

parse_arguments *Parse out options from a string without recourse to optparse*

Description

If only the key is given, the value will be set to TRUE

Usage

```
parse_arguments(x)
```

Arguments

x	Long-form argument list like –opt1 val1 –opt2 val2

Value

named list of options and values similar to optparse

Examples

```
parse_arguments("--opt1 val1 --opt2 val2")
parse_arguments(' --opt1-extra "value with space" --opt2 val2 ')
```

`process_end` *Process end of the workflow*

Description

Create the versions.yml file and log session info at the end of the process.

Usage

```
process_end(packages, task_name, out_dir = ".")
```

Arguments

`packages`	Named list of packages to add to the versions.yml. The items names should be the conda package name, while the items value should be the package name used in R.
`task_name`	Name of the nextflow process. Typically `${task.process}`
`out_dir`	Output directory where the versions.yml will be written to. Default is the current directory.

Value

versions.yml file and R session info log

Examples

```
td <- withr::local_tempdir()
process_end(list("r-stats" = "stats"), "MY_PROCESS", td)
```

`process_inputs` *Process input parameters and files*

Description

This function processes input parameters and files, applying parameter overrides, validating expected keys and files, and ensuring that the provided file paths are valid.

Usage

```
process_inputs(
  opt,
  args = NULL,
  keys_to_nullify = NULL,
  expected_files = NULL,
  expected_folders = NULL,
  expected_double = NULL,
  expected_integer = NULL,
  expected_boolean = NULL,
  required_opts = NULL
)
```

Arguments

opt	A list of default options.
args	A character vector of command-line arguments to override defaults.
keys_to_nullify	A character vector of keys that should be nullified.
expected_files	A character vector of keys in opt that are expected to be file paths.
expected_folders	A character vector of keys in opt that are expected to be folder paths.
expected_double	A character vector of keys in opt that are expected to be float values.
expected_integer	A character vector of keys in opt that are expected to be integer values.
expected_boolean	A character vector of keys in opt that are expected to be boolean values.
required_opts	A character vector of keys in opt that are required and must not be null or empty.

Value

A list of processed options with overrides applied and validated.

Examples

```
td <- withr::local_tempdir()
test_file_path <- file.path(td, "test_file.txt")
file.create(test_file_path)
options <- list(
  input_file = test_file_path,
  output_file = "prefix",
  threshold = 0.5,
  "is-a-test" = NULL
)
args <- c("--threshold 0.7 --is-a-test")
processed_options <- process_inputs(
  options, args,
```

```
  keys_to_nullify = c("input_file", "output_file"),
  expected_files = c("input_file"),
  expected_double = c("threshold"),
  expected_boolean = c("is-a-test"),
  required_opts = c("input_file", "threshold")
)
```

`read_delim_flexible` *Flexibly read CSV or TSV files*

Description

The following rules are applied:

- extension == txt or tsv, then separator is set to tabulation
- extension == csv, then separator is set to comma

Usage

```
read_delim_flexible(file, ...)
```

Arguments

`file`	Input file
`...`	Arguments passed on to `utils::read.delim`
	`header` a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: `header` is set to `TRUE` if and only if the first row contains one fewer field than the number of columns.
	`quote` the set of quoting characters. To disable quoting altogether, use `quote = ""`. See `scan` for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless `colClasses` is specified.
	`dec` the character used in the file for decimal points.
	`fill` logical. If `TRUE` then in case the rows have unequal length, blank fields are implicitly added. See 'Details'.
	`comment.char` character: a character vector of length one containing a single character or an empty string. Use `""` to turn off the interpretation of comments altogether.

Value

output Data frame

valid_string *Check for Non-Empty, Non-Whitespace String*

Description

This function checks if the input is non-NULL and contains more than just whitespace.

Usage

```
valid_string(input)
```

Arguments

input	A variable to check.

Value

input with whitespace trimmed or throw error if not a valid string

Examples

```
valid_string("Hello World") # Returns "Hello World"
valid_string("  Hello World ") # Returns "Hello World"
try(valid_string("   ")) # Error
try(valid_string(NULL)) # Error
```

validate_boolean *Check if value is a boolean*

Description

Convert to TRUE or FALSE with accepted lowered values as:

- TRUE: 1, yes, true
- FALSE: 0, no, false

Usage

```
validate_boolean(x, variable_name = NULL)
```

Arguments

x	String to check and convert
variable_name	Name of the variable to print if error occurs

Value

x as TRUE/FALSE, or NULL if is null and ERROR if x isn't convertable to a boolean

Examples

```
validate_boolean(NULL) # NULL
try(validate_boolean(" ")) # ERROR
try(validate_boolean("Hello World !")) # ERROR
try(validate_boolean(12.4)) # ERROR
validate_boolean("1") # TRUE
validate_boolean(0) # FALSE
```

---

validate_double *Check if value is a double number*

Description

Check if value is a double number

Usage

```
validate_double(x, variable_name = NULL)
```

Arguments

x	String to check and convert
variable_name	Name of the variable to print if error occurs

Value

x as double, or NULL if is null and ERROR if x isn't convertable to a double

Examples

```
validate_double(NULL) # NULL
try(validate_double(" ")) # ERROR
try(validate_double("Hello World !")) # ERROR
validate_double("12.4") # 12.4
validate_double("12") # 12.0
```

validate_file *Check if value is an existing file*

Description

Check if value is an existing file

Usage

```
validate_file(x, variable_name = NULL)
```

Arguments

x	String to check
variable_name	Name of the variable to print if error occurs

Value

x, or NULL if is null and ERROR if x isn't an existing file

Examples

```
try(validate_file("test")) # ERROR
```

validate_folder *Check if value is an existing folder*

Description

Check if value is an existing folder

Usage

```
validate_folder(x, variable_name = NULL)
```

Arguments

x	String to check
variable_name	Name of the variable to print if error occurs

Value

x, or NULL if is null and ERROR if x isn't an existing folder

Examples

```
try(validate_folder("test")) # ERROR
```

validate_integer *Check if value is an integer*

Description

Check if value is an integer

Usage

```
validate_integer(x, variable_name = NULL)
```

Arguments

x	String to check and convert
variable_name	Name of the variable to print if error occurs

Value

x as integer, or NULL if is null and ERROR if x isn't convertable to an integer

Examples

```
validate_integer(NULL) # NULL
try(validate_integer(" ")) # ERROR
try(validate_integer("Hello World !")) # ERROR
try(validate_integer("12.4")) # ERROR
validate_integer("12") # 12
```

Index