

# Package ‘scribe’

July 23, 2025

**Title** Command Argument Parsing

**Version** 0.3.0

**Maintainer** Jordan Mark Barbone <jmbarbone@gmail.com>

**Description** A base dependency solution with basic argument parsing for use with 'Rscript'.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.2.3

**Depends** R (>= 3.6)

**Imports** methods, utils

**Suggests** covr, knitr, rmarkdown, spelling, testthat (>= 3.1.0), withr

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** <https://jmbarbone.github.io/scribe/>,  
<https://github.com/jmbarbone/scribe>

**BugReports** <https://github.com/jmbarbone/scribe/issues>

**NeedsCompilation** no

**Author** Jordan Mark Barbone [aut, cph, cre] (ORCID:  
<<https://orcid.org/0000-0001-9788-3628>>)

**Repository** CRAN

**Date/Publication** 2023-10-21 21:50:02 UTC

## Contents

|                                   |   |
|-----------------------------------|---|
| command_args . . . . .            | 2 |
| new_arg . . . . .                 | 3 |
| scribeArg-class . . . . .         | 4 |
| scribeCommandArgs-class . . . . . | 6 |
| value_convert . . . . .           | 9 |

**Index****11**


---

|              |                               |
|--------------|-------------------------------|
| command_args | <i>Command line arguments</i> |
|--------------|-------------------------------|

---

**Description**

Make a new [scribeCommandArgs](#) object

**Usage**

```
command_args(
  x = NULL,
  include = getOption("scribe.include", c("help", "version", NA_character_)),
  string = NULL
)
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x, string | Command line arguments; see <a href="#">base::commandArgs()</a> for default. At least one parameter has to be NULL. When string is NULL, x is used, which defaults to <a href="#">commandArgs(trailingOnly = TRUE)</a> . Otherwise the value of x is converted to a character. If string is not NULL, <a href="#">scan()</a> will be used to split the value into a character vector. |
| include   | Special default arguments to included. See <a href="#">\$initialize()</a> in <a href="#">scribeCommandArgs</a> for more details.                                                                                                                                                                                                                                                      |

**Value**

A [scribeCommandArgs](#) object

**See Also**

Other scribe: [new\\_arg\(\)](#), [scribeArg-class](#), [scribeCommandArgs-class](#)

**Examples**

```
command_args()
command_args(c("-a", 1, "-b", 2))
command_args(string = "-a 1 -b 2")
```

---

|         |                             |
|---------|-----------------------------|
| new_arg | <i>New command argument</i> |
|---------|-----------------------------|

---

## Description

Make a new [scribeArg](#) object

## Usage

```
new_arg(  
  aliases = "",  
  action = arg_actions(),  
  default = NULL,  
  convert = scribe_convert(),  
  n = NA_integer_,  
  info = NULL,  
  options = list(),  
  stop = c("none", "hard", "soft"),  
  execute = invisible  
)
```

## Arguments

aliases, action, convert, options, default, info, n, stop, execute  
See `$initialize()` in [scribeArg](#).

## Value

A [scribeArg](#) object

## See Also

Other scribe: [command\\_args\(\)](#), [scribeArg-class](#), [scribeCommandArgs-class](#)

## Examples

```
new_arg()  
new_arg("values", action = "dots")  
new_arg(c("-f", "--force"), action = "flag")
```

---

|                 |                        |
|-----------------|------------------------|
| scribeArg-class | <i>scribe argument</i> |
|-----------------|------------------------|

---

## Description

ReferenceClass object for managing arguments

## Details

The [scribeArg](#) class sets specifications and controls for how command line arguments are to be parsed. These are meant to be used in conjunction with [scribeCommandArgs](#) and specifically with the [Rscript](#) utility. However, a use can define their own [scribeArg](#) separately.

## Fields

**aliases** [character]  
A vector to denote the argument's name

**action** [character]  
An action for resolving the argument (see default for note on using another [scribeArg](#) object)

**default** [ANY]  
A default value. This can be another [scribeArg](#) object. When that is the case, the default value and action are pass through from the other [scribeArg](#) object.

**convert** [ANY]  
Passed to the to argument in [value\\_convert\(\)](#)

**n** [integer]  
The length of the values

**info** [character]  
Additional information about the argument when printed

**options** [list]  
A named list of options (see **Options**)

**positional** [logical]  
Indicator if the argument is *positional* (i.e., not preceded by a - or -- command line argument)

**resolved** [logical]  
Has the object been resolved

**value** [ANY]  
The resolve value

**stop** [character]  
"none", "hard", or "soft"

**execute** [function]  
(For advanced use). A function to be evaluated along with the arg. The function can have no parameters, a single parameter for the [scribeArg](#) object, or accept the [scribeArg](#) object as its first argument, and the [scribeCommandArgs](#) object as its second. Both objects will be passed by position

## Methods

`get_action()` Retrieve action  
`get_aliases()` Retrieve aliases  
`get_default()` Retrieve the default value  
`get_help()` Retrieve help information as a character vector  
`get_name(clean = TRUE)` Retrieve names  
     `clean` When TRUE removes `-s` from text  
`get_value()` Retrieve the resolved value  
`help()` Print out formatted help information  
`initialize(aliases = "", action = arg_actions(), default = NULL, convert = scribe_convert(), n = NA_integer_)`  
     Initialize the [scribeArg](#) object  
     See **fields** for parameter information.  
`is_resolved()` Check if object has been resolved

## Options

Several available options

`action="list"` `choices` An explicit set of values that argument must be. If the value parsed is not one of these, an error will occur.

`action="flag"` `no` When TRUE included appends `--no` to aliases to invert results

### Example:

With the argument `new_arg("--test", options = list(no = TRUE))`, passing command arguments `--test` would set this to TRUE and `--no-test` explicitly set to FALSE.

## See Also

Other scribe: [command\\_args\(\)](#), [new\\_arg\(\)](#), [scribeCommandArgs-class](#)

## Examples

```
# new_arg() is recommended over direct use of scribeArg$new()

# arguments with `--` indicators
new_arg("--verbose", action = "flag")
new_arg(c("-f", "--force"), action = "flag")
new_arg("--values", action = "list")

# positional
new_arg("verbose", action = "flag")
new_arg("value", action = "list", n = 1)

# special `...` action which absorbs left-over arguments
new_arg("values", action = "dots", info = "list of values")
new_arg("...", info = "list of values") # defaults when alias is "..."
```

---

scribeCommandArgs-class

*scribe command arguments*


---

## Description

Reference class object for managing command line arguments.

## Details

This class manages the command line argument inputs when passed via the [Rscript](#) utility. Take the simple script below which adds two numbers, which we will save in an executable file called `add.R`,

```
#!/usr/bin/env Rscript

library(scribe)
ca <- command_args()
ca$add_argument("--value1", default = 0L)
ca$add_argument("--value2", default = 0L)
args <- ca$parse()
writeLines(args$value1 + args$value2)
```

When called by a terminal, we can pass arguments and return a function.

```
add.R --value1 10 --value2 1
11
```

When testing, you can simulate command line arguments by passing them into the input field. By default, this will grab values from `base::commandArgs()`, so use with the [Rscript](#) utility doesn't require any extra steps.

Most methods are designed to return `.self`, or the [scribeCommandArgs](#) class. The exceptions to these are the `$get_*`() methods, which return their corresponding values, and `$parse()` which returns a named list of the parsed input values.

## Fields

input [character]

A character vector of command line arguments. See also [command\\_args\(\)](#)

values [list]

A named list of values. Empty on initialization and populated during argument resolving.

args [list]

a List of [scribeArgs](#)

description [character]

Additional help information

included [character]  
     Default [scribeArgs](#) to include  
 examples [character]  
     Examples to print with help  
 comments [character]  
     Comments printed with  
 resolved [logical]  
     A logical value indicated if the `$resolve()` method has been successfully executed.  
 working [character]  
     A copy of input. Note: this is used to track parsing progress and is not meant to be accessed directly.  
 stop [character]  
     Determines parsing

## Methods

`add_argument( ..., action = arg_actions(), options = NULL, convert = scribe_convert(), default = NULL, n = ...)`  
     Add a [scribeArg](#) to args  
     ... Either aliases or a [scribeArg](#). If the latter, all other arguments are ignored. Note that only the first value (`..1`) is used.  
     action, options, convert, default, n, info See [new\\_arg\(\)](#)  
`add_description(..., sep = "")` Add a value to description  
     ... Information to paste into the description  
     sep character separate for ...  
`add_example(x, comment = "", prefix = "$ ")` Add a value to examples  
     x A code example as a character  
     comment An optional comment to append  
     prefix An optional prefix for the example  
`get_args(included = TRUE)` Retrieve args  
     included If TRUE also returns included default [scribeArgs](#) defined in `$initialize()`  
`get_description()` Retrieve description  
`get_examples()` Retrieve examples  
`get_input()` Retrieve input  
`get_values()` Retrieve values  
`help()` Print the help information  
`initialize(input = "", include = c("help", "version", NA_character_))` Initialize the [scribeCommandArgs](#) object. The wrapper [command\\_args\(\)](#) is recommended rather than calling this method directly.  
     input A character vector of command line arguments to parse  
     include A character vector denoting which default [scribeArgs](#) to include in args  
`parse()` Return a named list of parsed values of from each [scribeArg](#) in args

`resolve()` Resolve the values of each [scribeArg](#) in `args`. This method is called prior to `$parse()`

`set_description(..., sep = "")` Set the value of description

- ... Information to paste into the description
- sep character separate for ...

`set_example(x = character(), comment = "", prefix = "$ ")` Set the value of examples

- x A code example as a character
- comment An optional comment to append
- prefix An optional prefix for the example

`set_input(value)` Set input. Note: when called, resolved is (re)set to FALSE and values need to be parsed again.

- value Value to set

`set_values(i = TRUE, value)` Set values

- i Index value of working to set
- value The value to set

`version()` Print the [scribe-package](#) version

### See Also

Other scribe: [command\\_args\(\)](#), [new\\_arg\(\)](#), [scribeArg-class](#)

### Examples

```
# command_args() is recommended over direct use of scribeCommandArgs$new()

ca <- command_args(c(1, 2, 3, "--verbose"))
ca$add_argument("--verbose", action = "flag")
ca$add_argument("...", "values", info = "values to add", default = 0.0)
args <- ca$parse()

if (args$verbose) {
  message("Adding ", length(args$values), " values")
}

sum(args$values)

# $parse() returns a named list, which means scribeCommandArgs can function
# as a wrapper for calling R functions inside Rscript

ca <- command_args(c("mean", "--size", 20, "--absolute"))
ca$add_argument("fun", action = "list")
ca$add_argument("--size", default = 5L)
ca$add_argument("--absolute", action = "flag")
args <- ca$parse()

my_function <- function(fun, size, absolute = FALSE) {
  fun <- match.fun(fun)
  x <- sample(size, size, replace = TRUE)
```



```

    res <- fun(x)
    if (absolute) res <- abs(res)
    res
  }

do.call(my_function, args)

```

---

|               |                           |
|---------------|---------------------------|
| value_convert | <i>Simple conversions</i> |
|---------------|---------------------------|

---

## Description

Convert character to data types

## Usage

```

value_convert(x, to = default_convert)

scribe_convert(method = c("default", "evaluate", "none"))

```

## Arguments

|        |                                                                                                                                                                                                                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x      | A vector of character values                                                                                                                                                                                                                                                                     |
| to     | What to convert x to (see details for more)                                                                                                                                                                                                                                                      |
| method | The conversion method: <ul style="list-style-type: none"> <li>• TRUE or "default": uses <a href="#">value_convert()</a></li> <li>• "evaluate" executes the string as an expression</li> <li>• FALSE or NA does nothing</li> <li>• When passed a function, simply returns the function</li> </ul> |

## Details

to can be one of several values. Firstly the default of default calls several additional functions that attempt to resolve a transformation from a character vector to a different type. It is recommended for users to enter their own specifications instead. Secondly, a function (with a single argument) can be passed which will then be applied directly to x. Third, a *prototype* value can be passed. This might be risky for special types. Here, the values of [mode\(\)](#), [storage.mode\(\)](#), [attributes\(\)](#), and [class\(\)](#) are captured and reassigned from to to x. A special check is implemented for factors to more safely convert. Lastly, NULL will do nothing and will simply return x.

## Value

- [value\\_convert\(\)](#): A parsed value from x
- [scribe\\_convert\(\)](#): A function that takes a argument x and converts it

**Examples**

```
str(value_convert("2023-03-05", as.Date))  
value_convert("a", factor(letters))
```

# Index

- \* **scribe**
  - command\_args, 2
  - new\_arg, 3
  - scribeArg-class, 4
  - scribeCommandArgs-class, 6
  - ..1, 7
- attributes(), 9
- base::commandArgs(), 2, 6
- class(), 9
- command\_args, 2, 3, 5, 8
- command\_args(), 6, 7
- mode(), 9
- new\_arg, 2, 3, 5, 8
- new\_arg(), 7
- Rscript, 4, 6
- scan(), 2
- scribe-package, 8
- scribe\_convert(value\_convert), 9
- scribe\_convert(), 9
- scribeArg, 3–8
- scribeArg(scribeArg-class), 4
- scribeArg-class, 4
- scribeCommandArgs, 2, 4, 6, 7
- scribeCommandArgs
  - (scribeCommandArgs-class), 6
- scribeCommandArgs-class, 6
- storage.mode(), 9
- value\_convert, 9
- value\_convert(), 4, 9