

# Package ‘optparse’

January 16, 2024

**Encoding** UTF-8

**Type** Package

**Title** Command Line Option Parser

**Version** 1.7.4

**Description** A command line parser inspired by Python's 'optparse' library to be used with Rscript to write ``#!" shebang scripts that accept short and long flag/options.

**License** GPL (>= 2)

**Copyright** See file (inst/)COPYRIGHTS.

**URL** <https://github.com/trevorld/r-optparse>

**BugReports** <https://github.com/trevorld/r-optparse/issues>

**LazyLoad** yes

**Depends** R (>= 3.6.0)

**Imports** methods, getopt (>= 1.20.2)

**Suggests** knitr (>= 1.15.19), stringr, testthat

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Trevor L Davis [aut, cre] (<<https://orcid.org/0000-0001-6341-4639>>),  
Allen Day [ctb] (Some documentation and examples ported from the getopt package.),  
Python Software Foundation [ctb] (Some documentation from the optparse Python module.),  
Steve Lianoglou [ctb],  
Jim Nikelski [ctb],  
Kirill Müller [ctb],  
Peter Humburg [ctb],  
Rich FitzJohn [ctb],  
Gyu Jin Choi [ctb]

**Maintainer** Trevor L Davis <trevor.l.davis@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-01-16 10:30:02 UTC

## R topics documented:

optparse-package . . . . .	2
IndentedHelpFormatter . . . . .	3
make_option . . . . .	4
OptionParser . . . . .	6
OptionParser-class . . . . .	7
OptionParserOption-class . . . . .	8
parse_args . . . . .	9
print_help . . . . .	11

**Index** 13

---

optparse-package	<i>Command line option parser</i>
------------------	-----------------------------------

---

## Description

Goal is to create an R package of a command line parser inspired by Python’s “optparse” library.

## Details

optparse is primarily intended to be used with “Rscript”. It facilitates writing “#!” shebang scripts that accept short and long flags/options. It can also be used from directly, but is probably less useful in this context.

See package vignette for a more detailed example.

Notes on naming convention in package: 1. An option is one of the shell-split input strings. 2. A flag is a type of option. a flag can be defined as having no argument (defined below), a required argument, or an optional argument. 3. An argument is a type of option, and is the value associated with a flag. 4. A long flag is a type of flag, and begins with the string “-”. If the long flag has an associated argument, it may be delimited from the long flag by either a trailing =, or may be the subsequent option. 5. A short flag is a type of flag, and begins with the string “-”. If a short flag has an associated argument, it is the subsequent option. short flags may be bundled together, sharing a single leading “-”, but only the final short flag is able to have a corresponding argument. %%%

## Author(s)

Trevor Davis.

Some documentation and unit tests ported from Allen Day’s getopt package.

The documentation for Python’s optparse library, which this package is based on, is Copyright 1990-2009, Python Software Foundation.

## References

Python's optparse library, which this package is based on, is described here: <https://docs.python.org/3/library/optparse.html>

## See Also

[getopt](#)

## Examples

```
example_file <- system.file("exec", "example.R", package = "optparse")
example_file_2 <- system.file("exec", "display_file.R", package = "optparse")
## Not run:
  readLines(example_file)
  readLines(example_file_2)

## End(Not run)
```

---

IndentedHelpFormatter *Builtin help text formatters*

---

## Description

IndentedHelpFormatter() is the default help text formatter. TitledHelpFormatter() is an alternative help text formatter.

## Usage

```
IndentedHelpFormatter(object)
```

```
TitledHelpFormatter(object)
```

## Arguments

object            An [OptionParser\(\)](#) object.

## Value

NULL invisibly. As a side effect prints out help text.

**Examples**

```

parser <- OptionParser(formatter = IndentedHelpFormatter)
parser <- add_option(parser, "--generator", help = "Generator option")
parser <- add_option(parser, "--count", help = "Count option")
print_help(parser)

parser <- OptionParser(formatter = TitledHelpFormatter)
parser <- add_option(parser, "--generator", help = "Generator option")
parser <- add_option(parser, "--count", help = "Count option")
print_help(parser)

```

---

make_option	<i>Functions to enable our OptionParser to recognize specific command line options.</i>
-------------	---

---

**Description**

add\_option adds a option to a preexisting OptionParser instance whereas make\_option is used to create a list of OptionParserOption instances that will be used in the option\_list argument of the OptionParser function to create a new OptionParser instance.

**Usage**

```

make_option(
  opt_str,
  action = NULL,
  type = NULL,
  dest = NULL,
  default = NULL,
  help = "",
  metavar = NULL,
  callback = NULL,
  callback_args = NULL
)

```

```

add_option(
  object,
  opt_str,
  action = NULL,
  type = NULL,
  dest = NULL,
  default = NULL,
  help = "",
  metavar = NULL,
  callback = NULL,
  callback_args = NULL
)

```

**Arguments**

opt_str	A character vector containing the string of the desired long flag comprised of “-” followed by a letter and then a sequence of alphanumeric characters and optionally a string of the desired short flag comprised of the “-” followed by a letter.
action	A character string that describes the action optparse should take when it encounters an option, either “store”, “store_true”, “store_false”, or “callback”. An action of “store” signifies that optparse should store the specified following value if the option is found on the command string. “store_true” stores TRUE if the option is found and “store_false” stores FALSE if the option is found. “callback” stores the return value produced by the function specified in the callback argument. If callback is not NULL then the default is “callback” else “store”.
type	A character string that describes specifies which data type should be stored, either “logical”, “integer”, “double”, “complex”, or “character”. Default is “logical” if action %in% c(“store_true”, “store_false”), typeof(default) if action == “store” and default is not NULL and “character” if action == “store” and default is NULL. “numeric” will be converted to “double”.
dest	A character string that specifies what field in the list returned by parse_args should optparse store option values. Default is derived from the long flag in opt_str.
default	The default value optparse should use if it does not find the option on the command line.
help	A character string describing the option to be used by print_help in generating a usage message. %default will be substituted by the value of default.
metavar	A character string that stands in for the option argument when printing help text. Default is the value of dest.
callback	A function that executes after the each option value is fully parsed. It’s value is assigned to the option and its arguments are the option S4 object, the long flag string, the value of the option, the parser S4 object, and . . . .
callback_args	A list of additional arguments passed to callback function (via do.call).
object	An instance of the OptionParser class

**Value**

Both make\_option and add\_option return instances of class OptionParserOption.

**Author(s)**

Trevor Davis.

**References**

Python’s optparse library, which inspires this package, is described here: <https://docs.python.org/3/library/optparse.html>

**See Also**

[parse\\_args OptionParser](#)

**Examples**

```

make_option("--longflag")
make_option(c("-l", "--longflag"))
make_option("--integer", type = "integer", default = 5)
make_option("--integer", default = as.integer(5)) # same as previous

# examples from package vignette
make_option(c("-v", "--verbose"), action = "store_true", default = TRUE,
  help = "Print extra output [default]")
make_option(c("-q", "--quietly"), action = "store_false",
  dest = "verbose", help = "Print little output")
make_option(c("-c", "--count"), type = "integer", default = 5,
  help = "Number of random normals to generate [default %default]",
  metavar = "number")
make_option("--generator", default = "rnorm",
  help = "Function to generate random deviates [default \"%default\"]")
make_option("--mean", default = 0,
  help = "Mean if generator == \"rnorm\" [default %default]")
make_option("--sd", default = 1, metavar = "standard deviation",
  help = "Standard deviation if generator == \"rnorm\" [default %default]")

```

---

OptionParser

*A function to create an instance of a parser object*

---

**Description**

This function is used to create an instance of a parser object which when combined with the `parse_args`, `make_option`, and `add_option` methods is very useful for parsing options from the command line.

**Usage**

```

OptionParser(
  usage = "usage: %prog [options]",
  option_list = list(),
  add_help_option = TRUE,
  prog = NULL,
  description = "",
  epilogue = "",
  formatter = IndentedHelpFormatter
)

```

**Arguments**

usage	The program usage message that will be printed out if <code>parse_args</code> finds a help option, <code>%prog</code> is substituted with the value of the <code>prog</code> argument.
option_list	A list of <code>OptionParserOption</code> instances that will define how <code>parse_args</code> reacts to command line options. <code>OptionParserOption</code> instances are usually created by <code>make_option</code> and can also be added to an existing <code>OptionParser</code> instance via the <code>add_option</code> function.
add_help_option	Whether a standard help option should be automatically added to the <code>OptionParser</code> instance.
prog	Program name to be substituted for <code>%prog</code> in the usage message (including description and epilogue if present), the default is to use the actual Rscript file name if called by an Rscript file and otherwise keep <code>%prog</code> .
description	Additional text for <code>print_help</code> to print out between usage statement and options statement
epilogue	Additional text for <code>print_help</code> to print out after the options statement
formatter	A function that formats usage text. The function should take only one argument (an <code>OptionParser()</code> object). Default is <code>IndentedHelpFormatter()</code> . The other builtin formatter provided by this package is <code>TitledHelpFormatter()</code> .

**Value**

An instance of the `OptionParser` class.

**Author(s)**

Trevor Davis.

**References**

Python's `optparse` library, which inspired this package, is described here: <https://docs.python.org/3/library/optparse.html>

**See Also**

[parse\\_args](#) [make\\_option](#) [add\\_option](#)

---

OptionParser-class      *Option Parser*

---

**Description**

Option Parser

**Slots**

- usage** The program usage message that will be printed out if `parse_args` finds a help option, `%prog` is substituted with the value of the `prog` argument.
- options** A list of `OptionParserOption` instances that will define how `parse_args` reacts to command line options. `OptionParserOption` instances are usually created by `make_option` and can also be added to an existing `OptionParser` instance via the `add_option` function.
- description** Additional text for `print_help` to print out between usage statement and options statement
- epilogue** Additional text for `print_help` to print out after the options statement
- formatter** A function that `print_help` will use to print out after the options statement. Default is `IndentedHelpFormatter()`. This package also provides the builtin formatter `TitledHelpFormatter()`.

**Author(s)**

Trevor Davis.

**See Also**

[OptionParserOption](#)

---

OptionParserOption-class

*Class to hold information about command-line options*

---

**Description**

Class to hold information about command-line options

**Slots**

- short\_flag** String of the desired short flag comprised of the “-” followed by a letter.
- long\_flag** String of the desired long flag comprised of “-” followed by a letter and then a sequence of alphanumeric characters.
- action** A character string that describes the action `optparse` should take when it encounters an option, either “store”, “store\_true”, or “store\_false”. The default is “store” which signifies that `optparse` should store the specified following value if the option is found on the command string. “store\_true” stores `TRUE` if the option is found and “store\_false” stores `FALSE` if the option is found.
- type** A character string that describes specifies which data type should be stored, either “logical”, “integer”, “double”, “complex”, or “character”. Default is “logical” if `action in %c("store_true", store_false)`, `typeof(default)` if `action == "store"` and default is not `NULL` and “character” if `action == "store"` and default is `NULL`. “numeric” will be converted to “double”.
- dest** A character string that specifies what field in the list returned by `parse_args` should `optparse` store option values. Default is derived from the long flag in `opt_str`.



`default` The default value `optparse` should use if it does not find the option on the command line.

`help` A character string describing the option to be used by `print_help` in generating a usage message. `%default` will be substituted by the value of `default`.

`metavar` A character string that stands in for the option argument when printing help text. Default is the value of `dest`.

`callback` A function that executes after the each option value is fully parsed

`callback_args` Additional arguments that pass to the callback function.

**See Also**

[make\\_option](#)

---

parse_args	<i>Parse command line options.</i>
------------	------------------------------------

---

**Description**

`parse_args` parses command line options using an `OptionParser` instance for guidance. `parse_args2` is a wrapper to `parse_args` setting the options `positional_arguments` and `convert_hyphens_to_underscores` to `TRUE`.

**Usage**

```
parse_args(
  object,
  args = commandArgs(trailingOnly = TRUE),
  print_help_and_exit = TRUE,
  positional_arguments = FALSE,
  convert_hyphens_to_underscores = FALSE
)
```

```
parse_args2(
  object,
  args = commandArgs(trailingOnly = TRUE),
  print_help_and_exit = TRUE
)
```

**Arguments**

<code>object</code>	An <code>OptionParser</code> instance.
<code>args</code>	A character vector containing command line options to be parsed. Default is everything after the Rscript program in the command line. If <code>positional_arguments</code> is not <code>FALSE</code> then <code>parse_args</code> will look for positional arguments at the end of this vector.

**print\_help\_and\_exit**

Whether `parse_args` should call `print_help` to print out a usage message and exit the program. Default is `TRUE`.

**positional\_arguments**

Number of *positional* arguments. A numeric denoting the exact number of supported arguments, or a numeric vector of length two denoting the minimum and maximum number of arguments (`Inf` for no limit). The value `TRUE` is equivalent to `c(0, Inf)`. The default `FALSE` is supported for backward compatibility only, as it alters the format of the return value.

**convert\_hyphens\_to\_underscores**

If the names in the returned list of options contains hyphens then convert them to underscores. The default `FALSE` is supported for backward compatibility reasons as it alters the format of the return value

**Value**

Returns a list with field options containing our option values as well as another field `args` which contains a vector of positional arguments. For backward compatibility, if and only if `positional_arguments` is `FALSE`, returns a list containing option values.

**Acknowledgement**

A big thanks to Steve Lianoglou for a bug report and patch; Juan Carlos Borrás for a bug report; Jim Nikelski for a bug report and patch; Ino de Bruijn and Benjamin Tyner for a bug report; Jonas Zimmermann for bug report; Miroslav Posta for bug reports; Stefan Seemayer for bug report and patch; Kirill MÅ½ller for patches; Steve Humburg for patch.

**Author(s)**

Trevor Davis.

**References**

Python's `optparse` library, which inspired this package, is described here: <https://docs.python.org/3/library/optparse.html>

**See Also**

[OptionParser print\\_help](#)

**Examples**

```
# example from vignette
option_list <- list(
  make_option(c("-v", "--verbose"), action = "store_true", default = TRUE,
    help = "Print extra output [default]"),
  make_option(c("-q", "--quietly"), action = "store_false",
    dest = "verbose", help = "Print little output"),
  make_option(c("-c", "--count"), type = "integer", default = 5,
    help = "Number of random normals to generate [default %default]",
    metavar = "number"),
```

```

make_option("--generator", default = "rnorm",
  help = "Function to generate random deviates [default \"%default\"]"),
make_option("--mean", default = 0,
  help = "Mean if generator == \"rnorm\" [default %default]"),
make_option("--sd", default = 1, metavar = "standard deviation",
  help = "Standard deviation if generator == \"rnorm\" [default %default]")
)
parse_args(OptionParser(option_list = option_list), args = c("--sd=3", "--quietly"))

# example from vignette using positional arguments
option_list2 <- list(
  make_option(c("-n", "--add-numbers"), action = "store_true", default = FALSE,
    help = "Print line number at the beginning of each line [default]")
)
parser <- OptionParser(usage = "%prog [options] file", option_list = option_list2)

parse_args(parser, args = c("--add-numbers", "example.txt"), positional_arguments = TRUE)

parse_args(parser, args = c("--add-numbers", "example.txt"), positional_arguments = TRUE,
  convert_hyphens_to_underscores = TRUE)

parse_args2(parser, args = c("--add-numbers", "example.txt"))

```

---

print\_help

*Printing an usage message from an OptionParser object*


---

## Description

print\_help print an usage message from an OptionParser object, usually called by parse\_args when it encounters a help option.

## Usage

```
print_help(object)
```

## Arguments

object            A OptionParser instance.

## Value

print\_help uses the cat function to print out a usage message. It returns invisible(NULL).

## Author(s)

Trevor Davis.

**References**

Python's optparse library, which inspired this package, is described here: <https://docs.python.org/3/library/optparse.html>

**See Also**

`{parse_args}` `{OptionParser}`

# Index

- \* **package**
  - optparse-package, 2
- add\_option, 7
- add\_option (make\_option), 4
- getopt, 3
- IndentedHelpFormatter, 3
- IndentedHelpFormatter(), 7, 8
- make\_option, 4, 7, 9
- OptionParser, 6, 6, 10
- OptionParser(), 3
- OptionParser-class, 7
- OptionParserOption, 8
- OptionParserOption
  - (OptionParserOption-class), 8
- OptionParserOption-class, 8
- optparse (optparse-package), 2
- optparse-package, 2
- parse\_args, 6, 7, 9
- parse\_args2 (parse\_args), 9
- print\_help, 10, 11
- TitledHelpFormatter
  - (IndentedHelpFormatter), 3
- TitledHelpFormatter(), 7, 8