

Package ‘getopt’

October 1, 2023

Encoding UTF-8

Type Package

Title C-Like 'getopt' Behavior

Version 1.20.4

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

Imports stats

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

Description Package designed to be used with Rscript to write '#!' shebang scripts that accept short and long flags/options. Many users will prefer using instead the packages `optparse` or `argparse` which add extra features like automatically generated help option and usage, support for default values, positional argument support, etc.

License GPL (>= 2)

Suggests testthat

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Trevor L Davis [aut, cre] (<<https://orcid.org/0000-0001-6341-4639>>),
Allen Day [aut] (Original package author),
Roman Zenka [ctb]

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

Repository CRAN

Date/Publication 2023-10-01 06:30:01 UTC

R topics documented:

getopt	2
get_Rscript_filename	5
sort_list	5

Index	6
--------------	----------

 getopt

C-like getopt behavior

Description

getopt 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 R directly, but is probably less useful in this context.

Usage

```
getopt(
  spec = NULL,
  opt = NULL,
  command = get_Rscript_filename(),
  usage = FALSE,
  debug = FALSE
)
```

Arguments

spec	<p>The getopt specification, or spec of what options are considered valid. The specification must be either a 4-5 column matrix, or a character vector coercible into a 4 column matrix using <code>matrix(x, ncol=4, byrow=TRUE)</code> command. The matrix/vector contains:</p> <p>Column 1: the <i>long flag</i> name. A multi-character string.</p> <p>Column 2: <i>short flag</i> alias of Column 1. A single-character string.</p> <p>Column 3: <i>Argument</i> mask of the <i>flag</i>. An integer. Possible values: 0=no argument, 1=required argument, 2=optional argument.</p> <p>Column 4: Data type to which the <i>flag</i>'s argument shall be cast using <code>storage.mode()</code>. A multi-character string. This only considered for same-row Column 3 values of 1,2. Possible values: logical, integer, double, complex, character. If numeric is encountered then it will be converted to double.</p> <p>Column 5 (optional): A brief description of the purpose of the option.</p> <p>The terms <i>option</i>, <i>flag</i>, <i>long flag</i>, <i>short flag</i>, and <i>argument</i> have very specific meanings in the context of this document. Read the "Description" section for definitions.</p>
opt	<p>This defaults to the return value of <code>commandArgs(TRUE)</code> unless <code>argv</code> is in the global environment in which case it uses that instead (this is for compatibility with <code>littler</code>).</p> <p>If R was invoked directly via the R command, this corresponds to all arguments passed to R after the <code>--args</code> flag.</p> <p>If R was invoked via the Rscript command, this corresponds to all arguments after the name of the R script file.</p> <p>Read about <code>commandArgs()</code> and <code>Rscript</code> to learn more.</p>

command	The string to use in the usage message as the name of the script. See argument <i>usage</i> .
usage	If TRUE, argument <i>opt</i> will be ignored and a usage statement (character string) will be generated and returned from <i>spec</i> .
debug	This is used internally to debug the <code>getopt()</code> function itself.

Details

`getopt()` returns a list data structure containing names of the flags that were present in the character vector passed in under the *opt* argument. Each value of the list is coerced to the data type specified according to the value of the *spec* argument. See below for details.

Notes on naming convention:

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*.

Many users wonder whether they should use the `getopt` package, `optparse` package, or `argparse` package. Here is some of the major differences:

Features available in `getopt` unavailable in `optparse`

1. As well as allowing one to specify options that take either no argument or a required argument like `optparse`, `getopt` also allows one to specify option with an optional argument.

Some features implemented in `optparse` package unavailable in `getopt`

1. Limited support for capturing positional arguments after the optional arguments when `positional_arguments` set to TRUE in `optparse::parse_args()`
2. Automatic generation of an help option and printing of help text when encounters an `-h`
3. Option to specify default arguments for options as well the variable name to store option values

There is also new package `argparse` introduced in 2012 which contains all the features of both `getopt` and `optparse` but which has a dependency on Python 2.7 or 3.2+.

Some Features unlikely to be implemented in `getopt`:

1. Support for multiple, identical flags, e.g. for `-m 3 -v 5 -v`, the trailing `-v` overrides the preceding `-v 5`, result is `v=TRUE` (or equivalent typecast).
2. Support for multi-valued flags, e.g. `--libpath=/usr/local/lib --libpath=/tmp/foo`.

3. Support for lists, e.g. `--define os=linux --define os=redhat` would set `resultoslinux=TRUE` and `resultosredhat=TRUE`.
4. Support for incremental, argument-less flags, e.g. `/path/to/script -vvv` should set `v=3`.
5. Support partial-but-unique string match on options, e.g. `--verb` and `--verbose` both match long flag `--verbose`.
6. No support for mixing in positional arguments or extra arguments that don't match any options. For example, you can't do `my.R --arg1 1 foo bar baz` and recover `foo`, `bar`, `baz` as a list. Likewise for `my.R foo --arg1 1 bar baz`.

Author(s)

Allen Day

Examples

```
#!/path/to/Rscript
library('getopt')
# get options, using the spec as defined by the enclosed list.
# we read the options from the default: commandArgs(TRUE).
spec = matrix(c(
  'verbose', 'v', 2, "integer",
  'help'    , 'h', 0, "logical",
  'count'   , 'c', 1, "integer",
  'mean'    , 'm', 1, "double",
  'sd'      , 's', 1, "double"
), byrow=TRUE, ncol=4)
opt = getopt(spec)

# if help was asked for print a friendly message
# and exit with a non-zero error code
if (!is.null(opt$help)) {
  cat(getopt(spec, usage = TRUE))
  q(status = 1)
}

# set some reasonable defaults for the options that are needed,
# but were not specified.
if (is.null(opt$mean)) opt$mean <- 0
if (is.null(opt$sd))  opt$sd  <- 1
if (is.null(opt$count)) opt$count <- 10
if (is.null(opt$verbose)) opt$verbose <- FALSE

# print some progress messages to stderr, if requested.
if (opt$verbose) write("writing...", stderr())

# do some operation based on user input.
cat(rnorm(opt$count, mean = opt$mean, sd = opt$sd), sep = "\n")

# signal success and exit.
# q(status=0)
```

get_Rscript_filename *Returns file name being interpreted by Rscript*

Description

get_Rscript_filename() returns the file name that Rscript is interpreting.

Usage

```
get_Rscript_filename()
```

Value

A string with the filename of the calling script. If not found (i.e. you are in a interactive session) returns NA_character_.

sort_list *Recursively sorts a list*

Description

sort_list() returns a recursively sorted list

Usage

```
sort_list(unsorted_list)
```

Arguments

unsorted_list A list.

Value

A sorted list.

Examples

```
l <- list(b = 2, a = 1)
sort_list(l)
```

Index

* data

 getopt, 2

commandArgs(), 2

get_Rscript_filename, 5

getopt, 2

getopt(), 3

getopt-package (getopt), 2

Rscript, 2

sort_list, 5

storage.mode(), 2