

# The getopt Package

April 30, 2008

**Type** Package

**Title** C-like getopt behavior.

**Version** 1.14

**Date** 2008-04-29

**Author** Allen Day <allenday@ucla.edu>

**Maintainer** Allen Day <allenday@ucla.edu>

**Description** Use this with Rscript to write “#!” shebang scripts that accept short and long flags/options.

**License** GPL

## R topics documented:

getopt.package . . . . . 1

**Index** 5

---

getopt.package      *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.

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*.

## Usage

```
getopt ( spec=NULL, opt=commandArgs (TRUE), command=strsplit (commandArgs (FALSE) [4], "="
```

## Arguments

spec	<p>The getopt specification, or spec of what options are considered valid. The specification must be either a 4-5 column <a href="#">matrix</a>, or a <a href="#">character vector</a> coercible into a 4 column <a href="#">matrix</a> using <code>matrix(x,ncol=4,byrow=TRUE)</code> command. The <a href="#">matrix/vector</a> contains:</p> <p>Column 1: the <i>long flag</i> name. A multi-<a href="#">character</a> string.</p> <p>Column 2: <i>short flag</i> alias of Column 1. A single-<a href="#">character</a> string.</p> <p>Column 3: <i>Argument</i> mask of the <i>flag</i>. An <a href="#">integer</a>. 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 <a href="#">storage.mode</a>. A multi-<a href="#">character</a> string. This only considered for same-row Column 3 values of 1,2. Possible values: <a href="#">logical</a>, <a href="#">integer</a>, <a href="#">double</a>, <a href="#">complex</a>, <a href="#">character</a>.</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>.</p> <p>If R was invoked directly via the “R” command, this corresponds to all arguments passed to R after the “-args” 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 <a href="#">commandArgs</a> and <a href="#">Rscript</a> to learn more.</p>
command	<p>The string to use in the usage message as the name of the script. See argument <i>usage</i>.</p>
usage	<p>If TRUE, argument <i>opt</i> will be ignored and a usage statement (character string) will be generated and returned from <i>spec</i>.</p>
debug	<p>This is used internally to debug the <code>getopt()</code> function itself.</p>

## Details

Current issues:

1. No 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. No support for multi-valued flags, e.g. "-libpath=/usr/local/lib -libpath=/tmp/foo".
3. No support for lists, e.g. "-define os=linux -define os=redhat" would set resultoslinux=TRUE and resultosredhat=TRUE.
4. No support for incremental, argument-less flags, e.g. "/path/to/script -vvv" should set v=3.
5. Need more unit tests (see end of examples section).
6. Support partial-but-unique string match on options, e.g. "-verb" and "-verbose" both match long flag "-verbose".

## Author(s)

Allen Day

## See Also

[getopt](#)

## 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).
opt = getopt(c(
  'verbose', 'v', 2, "integer",
  'help'    , 'h', 0, "logical",
  'count'   , 'c', 1, "integer",
  'mean'    , 'm', 1, "double",
  'sd'      , 's', 1, "double"
));

#help was asked for.
if ( !is.null(opt$help) ) {
  #get the script name (only works when invoked with Rscript).
  self = commandArgs()[1];
  #print a friendly message and exit with a non-zero error code
  cat(paste("Usage: ",self," [-[vh]] [-[mean|m] <mean>] [-[sd|s] <sd>] [-[-count|c] <count>]"),
      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(paste(rnorm(opt$count,mean=opt$mean,sd=opt$sd),collapse="\n"));
cat("\n");

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

### END ###
#regression tests follow.  not part of the example.
spec = c(
  'verbose', 'v', 2, "integer",
  'help'    , 'h', 0, "logical",
  'dummy1'  , 'd', 0, "logical",
  'dummy2'  , 'e', 2, "logical",
  'count'   , 'c', 1, "integer",
  'mean'    , 'm', 1, "double",
  'sd'      , 's', 1, "double",
  'output'  , 'O', 1, "character"
);
opt = getopt(spec, c('-v', '-m', '3'));
opt = getopt(spec, c('-m', '3', '-v'));
opt = getopt(spec, c('-m', '3', '-v', '2', '-v'));
opt = getopt(spec, c('-O', '-', '-m', '3'));
opt = getopt(spec, c('-m', '3', '-O', '-'));
opt = getopt(spec, c('-de'));
opt = getopt(spec, c('-ed'));
opt = getopt(spec, c('-d'));
opt = getopt(spec, c('-e', '1'));
opt = getopt(spec, c('-de', '1'));
opt = getopt(spec, c('--verbose'));
opt = getopt(spec, c('--help'));
opt = getopt(spec, c('--verbose', '--help'));
opt = getopt(spec, c('--verbose', '--mean', '5'));
opt = getopt(spec, c('--mean=5'));
opt = getopt(spec, c('--verbose', '--mean=5'));
opt = getopt(spec, c('--verbose', '--mean=5', '--sd', '5'));
opt = getopt(spec, c('--mean=5', '--sd', '5', '--verbose'));
opt = getopt(spec, c('--mean=5', '--verbose', '--sd', '5'));

spec = c(
  'date'      , 'd', 1, "character",
  'help'      , 'h', 0, "logical",
  'getdata'   , 'g', 0, "logical",
  'market'    , 'm', 1, "character",
  'threshold' , 't', 1, "double"
);
opt = getopt(spec, c('--date', '20080421', '--market', 'YM', '--getdata'));
opt = getopt(spec, c('--date', '20080421', '--getdata', '--market', 'YM'));
opt = getopt(spec, c('--date', '20080421', '--getdata', '--market', 'YM'), usage=TRUE, debug=TRUE)

```

```
print(getopt(spec, c('--date', '20080421', '--getdata', '--market', 'YM'), usage=TRUE));
```

# Index

## \*Topic **data**

- `getopt.package`, [1](#)
  
- `character`, [1](#), [2](#)
- `commandArgs`, [2](#)
- `complex`, [2](#)
  
- `double`, [2](#)
  
- `getopt`, [3](#)
- `getopt (getopt.package)`, [1](#)
- `getopt.package`, [1](#)
  
- `integer`, [2](#)
  
- `list`, [1](#)
- `logical`, [2](#)
  
- `matrix`, [2](#)
  
- `names`, [1](#)
  
- `Rscript`, [1](#), [2](#)
  
- `storage.mode`, [2](#)
  
- `vector`, [1](#), [2](#)