Package ‘getopt’

February 19, 2015

Type Package

Title C-like getopt behavior.

Version 1.20.0

Author Allen Day.
Contributions from Trevor L Davis.

Maintainer Trevor L Davis <trevor.l.davis@stanford.edu>

URL https://github.com/trevorld/getopt

BugReports https://github.com/trevorld/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

Collate ‘getopt.R’ 'utils.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-08-30 10:36:29

R topics documented:

  getopt ......................................................... 2
  get_Rscript_filename ................................. 4
  sort_list ................................................. 5

Index 6
getopt

**C-like getopt behavior**

Description

g 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

g getopt(spec = NULL, opt = commandArgs(TRUE),
    command = get_Rscript_filename(), usage = FALSE,
    debug = FALSE)

Arguments

**spec**

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 matrix(x,ncol=4,byrow=TRUE) command. The matrix/vector contains:

Column 1: the *long flag* name. A multi-character string.

Column 2: *short flag* alias of Column 1. A single-character string.

Column 3: *Argument* mask of the *flag*. An *integer*. Possible values: 0=no argument, 1=required argument, 2=optional argument.

Column 4: Data type to which the *flag*’s argument shall be cast using *storage.mode*. 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.

Column 5 (optional): A brief description of the purpose of the option.

The terms *option, flag, long flag, short flag, and argument* have very specific meanings in the context of this document. Read the “Description” section for definitions.

**opt**

This defaults to the return value of commandArgs(TRUE).

If R was invoked directly via the “R” command, this corresponds to all arguments passed to R after the “--args” flag.

If R was invoked via the “Rscript” command, this corresponds to all arguments after the name of the R script file.

Read about commandArgs and Rscript to learn more.

**command**

The string to use in the usage message as the name of the script. See argument usage.

**usage**

If TRUE, argument opt will be ignored and a usage statement (character string) will be generated and returned from spec.

**debug**

This is used internally to debug the getopt() function itself.
getopt

Details

g 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 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+ and has not been used in production since 2008 or 2009 like the getopt and optparse packages.

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 result$s$os$linux=TRUE and result$s$os$redhat=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".
get_Rscript_filename

Returns file name of calling Rscript

Author(s)

Allen Day

See Also

g 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).
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(specL 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(paste(rnorm(opt$countLmean=opt$meanLsd=opt$sd), collapse="\n"));
  cat("\n");

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

get_Rscript_filename
sort_list

Description
get_Rscript_filename returns the file name of calling Rscript

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.

sort_list Recursively sorts a list

Description
sort_list returns a sorted list

Usage
sort_list(unsorted_list)

Arguments
unsorted_list A list.

Value
A sorted list.
Index

*Topic data
  getopt, 2

character, 2, 3
commandArgs, 2
complex, 2
double, 2
get_Rscript_filename, 4
gopt, 2, 4
gopt-package (getopt), 2

integer, 2
list, 3
logical, 2

matrix, 2
names, 3
numeric, 2

Rscript, 2

sort_list, 5
storage.mode, 2

vector, 2, 3