Package ‘GetoptLong’

June 10, 2020

Type Package
Title Parsing Command-Line Arguments and Simple Variable Interpolation
Version 1.0.0
Date 2020-06-02
Author Zuguang Gu
Maintainer Zuguang Gu <z.gu@dkfz.de>
Depends R (>= 3.6.0)
Imports rjson, GlobalOptions (>= 0.1.0), methods, crayon
Suggests testthat (>= 1.0.0), knitr, markdown
VignetteBuilder knitr
Description This is a command-line argument parser which wraps the powerful Perl module Getopt::Long and with some adaptations for easier use in R. It also provides a simple way for variable interpolation in R.
URL https://github.com/jokergoo/GetoptLong
SystemRequirements Perl, Getopt::Long
License MIT + file LICENSE
Repository CRAN
NeedsCompilation no
Date/Publication 2020-06-10 20:00:06 UTC

R topics documented:

  GetOptions ................................. 2
  GetoptLong .................................. 2
  GetoptLong.options ....................... 4
  get_scriptdir ............................. 5
  get_scriptname ......................... 5
  qq ........................................ 6
  qq.options ............................... 7
  qqcat .................................... 8
  source_script .......................... 9
GetOptions

Wrapper of the Perl module Getopt::Long in R

Description

Wrapper of the Perl module Getopt::Long in R

Usage

GetOptions(..., envir = parent.frame())

Arguments

... 

Pass to GetoptLong.

envir 

User’s environment where GetoptLong looks for default values and exports variables.

Details

This function is the same as GetoptLong. It is just to make it consistent as the GetOptions() subroutine in Getopt::Long module in Perl.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

# There is no example
NULL

GetoptLong

Wrapper of the Perl module Getopt::Long in R

Description

Wrapper of the Perl module Getopt::Long in R

Usage

GetoptLong(..., help_head = NULL, help_foot = NULL, envir = parent.frame(), argv_str = NULL, template_control = list(), help_style = GetoptLong.options$help_style)
Arguments

- `...` Specification of options. The value can be a two-column matrix, a vector with even number of elements or a text template. See the vignette for detailed explanation.
- `help_head` Head of the help message when invoking Rscript foo.R --help.
- `help_foot` Foot of the help message when invoking Rscript foo.R --help.
- `envir` User's environment where `GetoptLong` looks for default values and exports variables.
- `argv_str` A string that contains command-line arguments. It is only for testing purpose.
- `template_control` A list of parameters for controlling when the specification is a template.
- `help_style` The style of the help messages. Value should be either "one-column" or "two-column".

Details

Following shows a simple example. Put following code at the beginning of your script (e.g. foo.R):

```r
library(GetoptLong)
cutoff = 0.05
GetoptLong(
    "number=i", "Number of items.",
    "cutoff=f", "Cutoff for filtering results.",
    "verbose", "Print message."
)
```

Then you can call the script from command line either by:

```bash
Rscript foo.R --number 4 --cutoff 0.01 --verbose
Rscript foo.R --number 4 --cutoff=0.01 --verbose
Rscript foo.R -n 4 -c 0.01 -v
Rscript foo.R -n 4 --verbose
```

In this example, `number` is a mandatory option and it should only be in integer mode. `cutoff` is optional and it already has a default value 0.05. `verbose` is a logical option. If parsing is successful, two variables `number` and `verbose` will be imported into the working environment with the specified values. Value for `cutoff` will be updated if it is specified in command-line.

For advanced use of this function, please go to the vignette.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```r
# There is no example
NULL
```
GetoptLong.options

Global options for GetoptLong()

Description

Global options for GetoptLong()

Usage

GetoptLong.options(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)

Arguments

... Options, see 'Details' section.
RESET Whether to reset options to their default values.
READ.ONLY Whether to only return read-only options.
LOCAL Whether to switch local mode.
ADD Whether to add new options.

Details

Supported global options are following:

config Configuration of Getopt::Long, check http://perldoc.perl.org/Getopt/Long.html#Configuring-Getopt%3a%3aLong.
template_tag The tag for identifying specifications in the template. The format should be in left_tag CODE right_tag.
help_style The style of the help message.

GetoptLong.options(...) should be put before calling GetoptLong function.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

# There is no example
NULL
### get_scriptdir

**Directory of current script**

**Description**

Directory of current script

**Usage**

```
get_scriptdir()
```

**Value**

If the R script is not run from the command-line, it returns `NULL`.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

### get_scriptname

**File name of current script**

**Description**

File name of current script

**Usage**

```
get_scriptname()
```

**Value**

If the R script is not run from the command-line, it returns `NULL`.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```
Simple variable interpolation in texts

Description

Simple variable interpolation in texts

Usage

```r
qq(..., envir = parent.frame(), code.pattern = NULL, collapse = TRUE, sep = " ")
```

Arguments

- `...`: Text string in which variables are marked with certain rules
- `envir`: Environment where to look for variables. By default it is the environment where `qq` is envoked. It can also be a list in which element names are the variable names to be interpolated.
- `code.pattern`: Pattern of marks for the variables. By default it is `@\{CODE\}` which means you can write your variable as `@{variable}`. This value can be a vector that all patterns are searched.
- `collapse`: If variables return vector of length larger than one, whether collapse into one string or return a vector
- `sep`: Separator character when there are multiple templates.

Details

I like variable interpolation in Perl. But in R, if you want to concatenate plain text and variables, you need to use functions such as `paste`. However, if there are so many variables, quotes, braces in the string you want to construct, it would be painful.

This function allows you to construct strings as in Perl style. Variables are marked in the text with certain rule. `qq` will look up these variables in user’s environment and replace the variable marks with their real values.

For more explanation of this function, please refer to vignette.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```r
a = 1
b = "text"
qq("a = @{a}, b = '"b'")
qq("a = @{a}"b = '"b'"", sep = ", ")
```

a = 1:2
qq.options

qq("a = @\{a\}, b = '@\{b\}'")
qq("a = @\{a\}, b = '@\{b\}'", collapse = FALSE)

a = 1
qq("a = 'a\', b = 'b\', code.pattern = '\"CODE\"'")

qq.options

Global options for qq() related functions

Description

Global options for qq() related functions

Usage

qq.options(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)

Arguments

... Options, see 'Details' section.
RESET Whether to reset options to their default values.
READ.ONLY Whether to only return read-only options.
LOCAL Whether to switch local mode.
ADD Whether to add new options.

Details

Supported options are following:

cat_prefix prefix of the string which is printed by qqcat
cat_verbose whether to print text by qqcat
cat_strwrap whether call strwrap to wrap the string
code.pattern code pattern for variable interpolation

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

a = 1
qq.options(cat_prefix = "[INFO ]")
qqcat("a = @\{a\}\n")
qq.options(cat_verbose = FALSE)
qqcat("a = @\{a\}\n")
qq.options(RESET = TRUE)
qq.options(code.pattern = "\`CODE\")
qqcat("a = \`a\n")
qq.options(RESET = TRUE)
Print a string which has been interpolated with variables

Description

Print a string which has been interpolated with variables

Usage

```r
qqcat(..., envir = parent.frame(), code.pattern = NULL, file = "", sep = " ", fill = FALSE, labels = NULL, append = FALSE, cat_prefix = NULL, strwrap = qq.options("cat_strwrap"), strwrap_param = list(), sep2 = "")
```

Arguments

- `...`: text string in which variables are marked with certain rules
- `envir`: environment where to look for those variables
- `code.pattern`: pattern of marks for the variables
- `file`: pass to `cat`
- `sep`: pass to `cat`
- `fill`: pass to `cat`
- `labels`: pass to `cat`
- `append`: pass to `cat`
- `cat_prefix`: prefix string. It is prior than `qq.options(cat_prefix)`.
- `strwrap`: whether call `strwrap` to wrap the string
- `strwrap_param`: parameters sent to `strwrap`, must be a list
- `sep2`: Separation character when there are multiple templates.

Details

This function is a shortcut of

```r
cat(qq(text, envir, code.pattern), ...)
```

Additionally, you can add global prefix:

```r
qq.options("cat_prefix" = "[INFO] ")
qq.options("cat_prefix" = function(x) format(Sys.time(), "[%Y-%m-%d %H:%M:%S] "))
qq.options("cat_prefix" = NULL)
```

You can also add local prefix by specifying `cat_prefix` in `qqcat`.

```r
qqcat(text, cat_prefix = "[INFO] ")
```

Please refer to `qq` to find more details.
## source_script

Source the R script with command-line arguments

### Description

Source the R script with command-line arguments

### Usage

```r
source_script(file, ..., argv_str = NULL)
```

### Arguments

- `file`: The R script
- `...`: Pass to `source).
- `argv_str`: The command-line arguments.

### Examples

```r
# There is no example
NULL
```
Index

cat, 8
get_scriptdir, 5
get_scriptname, 5
GetOptions, 2
GetoptLong, 2, 3, 4
GetoptLong.options, 4
paste, 6
qq, 6, 6, 8
qq.options, 7
qqcat, 7, 8, 8
source, 9
source_script, 9
strwrap, 7, 8