Package 'GetoptLong'

September 26, 2016

Type Package

Title Parsing Command-Line Arguments and Variable Interpolation

Version 0.1.5

Date 2016-8-22

Author Zuguang Gu

Maintainer Zuguang Gu <z.gu@dkfz.de>

Depends R (>= 3.2.0)

Suggests testthat (>= 1.0.0), knitr, markdown

VignetteBuilder knitr

Imports rjson, GlobalOptions (>= 0.0.10), methods

Description This is yet another command-line argument parser which wraps the powerful Perl module Getopt::Long and with some adaptation 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 GPL (>= 2)

Repository CRAN

Date/Publication 2016-09-26 23:38:10

NeedsCompilation no

R topics documented:

GetOptions *Wrapper of the Perl module* Getopt::Long *in R*

Description

Wrapper of the Perl module Getopt::Long in R

Usage

```
GetOptions(...)
```

Arguments

...	pass to GetoptLong

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 = TRUE, version = TRUE, envir = parent.frame(), argv_str = NULL,
    head = NULL, foot = NULL)
```

Arguments

...	specification of options. The value should be a vector having even number of elements.
help	whether to add help option
version	whether to add version option
envir	user's enrivonment where GetoptLong will look for default values and export variables
argv_str	command-line arguments, only for testing purpose
head	head of the message when invoking Rscript foo.R --help
foot	foot of the message when invoking Rscript foo.R --help

Details

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

```
library(GetoptLong)
cutoff = 0.05
GetoptLong(
    "number=i", "Number of items, integer, mandatory option",
    "cutoff=f", "cutoff to filter results, optional, default (0.05)",
    "verbose",  "print messages"
)
```

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

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

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# 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)
```

Arguments

...	options, see 'details' section
RESET	Whether to reset options to their default values
READ.ONLY	only return read-only options?
LOCAL	switch local mode

Details

Supported options are following:

startingMsg message that will be printed before the helping message when running `Rscript foo.R --help`. Ignored if `head` is set in `GetoptLong`

endingMsg message that will be printed after the helping message when running `Rscript foo.R --help`. Ignored if `foot` is set in `GetoptLong`

config configuration of `Getopt::Long`, check http://perldoc.perl.org/Getopt/Long.html#Configuring-Getopt%3a%3aLong

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

get_scriptname *Full path of current script*

Description

Full path of current script

Usage

```
get_scriptname()
```

Details

...

Value

If the R script is not run under command-line, it return NULL.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

qq *Simple variable interpolation in texts*

Description

Simple variable interpolation in texts

Usage

```
qq(text, envir = parent.frame(), code.pattern = NULL, collapse = TRUE)
```

Arguments

`text`	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}`.
`collapse`	If variables return vector of length larger than one, whether collapse into one string or return a vector

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 explaination of this function, please refer to vignette.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

a = 1:2
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)
```

Arguments

...	options, see 'details' section
RESET	Whether to reset options to their default values
READ.ONLY	only return read-only options?
LOCAL	switch local mode

Details

Supported options are following:

cat_prefix prefix of the string which is printed by qqcat

cat_verbose whether to print text by qqcat

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)
```

---

qqcat *Print a string which has been intepolated with variables*

Description

Print a string which has been intepolated with variables

Usage

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

Arguments

text	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

Details

This function is a shortcut of

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

Additionally, you can add global prefix:

```
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.

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

Please refer to qq to find more details.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
a = 1
b = "text"
qqcat("a = @{a}, b = '@{b}'\n")
qqcat("a = `a`, b = '`b`'\n", code.pattern = "`CODE`")

qq.options("cat_prefix" = function(x) format(Sys.time(), "[%Y-%m-%d %H:%M:%S] "))
qqcat("a = @{a}, b = '@{b}'\n")
Sys.sleep(2)
qqcat("a = @{a}, b = '@{b}'\n")
qq.options(RESET = TRUE)
```

Index