Package 'params'

July 23, 2025

Type Package

Title Simplify Parameters

Description An interface to simplify organizing parameters used in a package, using external configuration files. This attempts to provide a cleaner alternative to options().

Version 0.7.7

Depends R (>= 3.0.2), whisker

Imports utils, RcppTOML, glue, readr (>= 1.4.0), purrr

Suggests openxlsx, readxl, testthat, knitr, utf8

License GPL-2

URL https://github.com/sahilseth/params

BugReports https://github.com/sahilseth/params/issues

RoxygenNote 7.3.2

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Sahil Seth [aut, cre],
Yihui Xie [ctb] (kable from knitr R/table.R)

Maintainer Sahil Seth <me@sahilseth.com>

Repository CRAN

Date/Publication 2025-02-07 00:40:07 UTC

Contents

`check_args` *Assert none of the arguments of a function are null.*

Description

Checks all the arguments in the parent function and makes sure that none of them are NULL

Usage

```
check_args(ignore, select)
```

Arguments

`ignore`	optionally ignore a few variables for checking [character vector].
`select`	optionally only check a few variables of the function [character vector].

Examples

```
myfunc <- function(verbose = get_opts("verbose"), b = get_opts("b")){
  check_args()
}

set_opts(verbose = 1)
## this will throw an error, suggesting b is not defined properly
try(myfunc())
```

`fix_column_names` *fix_column_names*

Description

removes special chars from names

Usage

```
fix_column_names(x, char = "_")
```

Arguments

`x`	a character vector
`char`	replace special characters with this symbol

`fix_names` *fix_names*

Description

Replace special characters in column names of a data.frame

Usage

```
fix_names(x, char = "_")
```

Arguments

`x`	a vector of column names
`char`	substitute special char with this.

See Also

make.names

`kable` *Create tables in LaTeX, HTML, Markdown and reStructuredText*

Description

This is a very simple table generator. It is simple by design. It is not intended to replace any other R packages for making tables. This is a trimmed down version of the original kable function in knitr package. Please refer to knitr's kable function for details.

Usage

```
kable(
  x,
  format,
  digits = getOption("digits"),
  row.names = NA,
  col.names = colnames(x),
  align,
  caption = NULL,
  escape = TRUE,
  ...
)
```

Arguments

x	an R object (typically a matrix or data frame)
format	a character string; possible values are latex, html, markdown, pandoc, and rst; this will be automatically determined if the function is called within **knitr**; it can also be set in the global option knitr.table.format
digits	the maximum number of digits for numeric columns (passed to round()); it can also be a vector of length ncol(x) to set the number of digits for individual columns
row.names	a logical value indicating whether to include row names; by default, row names are included if rownames(x) is neither NULL nor identical to 1:nrow(x)
col.names	a character vector of column names to be used in the table
align	the alignment of columns: a character vector consisting of 'l' (left), 'c' (center) and/or 'r' (right); by default, numeric columns are right-aligned, and other columns are left-aligned; if align = NULL, the default alignment is used
caption	the table caption
escape	escape special characters when producing HTML or LaTeX tables
...	other arguments (see examples)

Author(s)

Yihui Xie http://yihui.org

---

load_opts *Setting/loading and extracting various options into the environment*

Description

- set_opts(): set options into a custom envir
- get_opts(): extract options
- load_opts(): Read a tab delimited file using read_sheet or toml file and load them as options using set_opts
- new_opts(): create a options manager to be included in a pacakge
- print.opts(): print pkg options as a pretty table

Usage

```
load_opts(x, check = TRUE, envir = opts, verbose = TRUE, .parse = TRUE, ...)

load_toml(toml, .remove_period = T, envir = envir, verbose = T)

new_opts(envir = new.env())

get_opts(x, envir = opts, .use.names = FALSE)
```

```
set_opts(..., .dots, .parse = TRUE, envir = opts)

## S3 method for class 'opts'
print(x, ...)
```

Arguments

x	• get_opts: a character vector of names of options to extract. • load_opts: path to a configuration file
check	load_opts(): in case of a configuration file, whether to check if files defined in parameters exists. [TRUE]
envir	environ used to store objects. Default is a environ object called opts [params:::opts]
verbose	load_opts(): Logical variable indicate level of verboseness [TRUE]
.parse	set_opts(), load_opts(): logical, whether to auto-complete {{myvar}} using previously defined options. [TRUE]
...	set_opts(): a named set of variable/value pairs separated by comma
toml	load_toml(): instead of a tsv, use toml to load options
.remove_period	load_opts(): remove \. period from option names (and replace with _)
.use.names	get_opts(): The resulting vector should be have names (esp. if length(x) is 1). If length(x)>1, this returns a list.
.dots	set_opts(): A named list, as a alternative to ...

Details

Integrating params in a package:

create a options manager:

`opts_mypkg = new_opts()`

The object `opts_mypkg` is a list of a few functions, which set, fetch and load options (using a isolated environment). Here are a few examples:

Set some options:

`opts_mypkg$set(version = '0.1', name = 'mypkg')`

Fetch ALL options:

`opts_mypkg$get()` OR `opts_mypkg$get("version")` to fetch a specific option.

Loading configuration files:

`load_opts()` OR `opts_pkg$load()`:

There are cases when options and params are actually paths to scripts or other apps or folders etc. In such cases it might be useful to quickly check if these paths exists on the system. As such, load_opts() automatically checks params ending with `path|dir|exe` (if check=TRUE).

For example, values for variables like `mypath`, `my_path`, `tool_exe`, etc would be check if they exists and a warning would be shown if they don't exist.

Below is a list example options, retrieved via

`get_opts()`:

```
|name          |value            |
|default_regex |(.*)             |
|my_conf_path  |~/flowr/conf     |
|my_dir        |path/to/a/folder |
|my_path       |~/flowr          |
|my_tool_exe   |/usr/bin/ls      |
```

See Also

read_sheet

read_sheet

read_sheet load_opts

Examples

```
## Set options
opts = set_opts(flow_run_path = "~/mypath")
#OR
opts = set_opts(.dots = list(flow_run_path = "~/mypath"))

## printing options, this is internally called by get_opts()
print(opts)

## Fetch options
get_opts()
get_opts("flow_run_path")

## Load options from a file
fl = system.file("conf/params.conf", package = "params")
load_opts(fl)


## Create a options manager:
opts_mypkg = new_opts()
## this provides three functions
opts_mypkg$set(version = '0.1', name = 'mypkg')
opts_mypkg$load(fl)
opts_mypkg$get()

## Additionally, one has the options of using braces ({{}})
## do define nested options:

set_opts(first = "John", last = "Doe", full = "{{first}} {{last}}")
```

---

opts *default opts*

Description

default opts

Usage

```
opts
```

Format

An object of class `environment` of length 6.

`parse_opts` *Parse options to expand* `{{variable}}` *into their respective values*

Description

This function is internally called by set_opts and load_opts

Usage

```
parse_opts(lst, envir)
```

Arguments

`lst`	a list of configuration options to parse
`envir`	environ used to store objects. Default is a environ object called opts [params:::opts]

`read_sheet` *Read/Write sheets (automatically detect the file type and work accordingly)*

Description

Read/Write sheets (automatically detect the file type and work accordingly)

write_sheet requires version 0.3.1.

- **tsv, txt, conf, def**: assumed to be tab-delimited
- **csv**: assumed to be comma delimited
- **xlsx**: microsoft excel, uses openxlsx to read the sheet. Also, it removes extra columns which often creep into excel files.

Usage

```
read_sheet(
  x,
  id_column,
  start_row = 1,
  sheet = 1,
  ext,
  header = TRUE,
  verbose = FALSE,
  ...
)

write_sheet(x, file, ext, type = "", ...)
```

Arguments

x	read: path to a file, to be read. write: a data.frame
id_column	all rows which have this column as blank are skipped. See details.
start_row	supplied to read.xlsx
sheet	supplied to read.xlsx, index or name of the sheet to be read from excel file. See read.xlsx
ext	determined using file extension. Specifying will override
header	first line is header? See read.table
verbose	verbosity level.
...	passed onto read.xlsx of openxlsx, read.table or read.csv2 depending on the file type.
file	write: output file name.
type	in case of writing an xlsx file, should the data.frame to written as excel 'table'. ['table']

Details

Note: for excel sheets:

- If `id_column` is missing, default if first column
- If `sheet` is missing, it automatically reads the first sheet

Some important default values for tsv and csv files:

`stringsAsFactors = FALSE, comment.char = '#', strip.white=TRUE, blank.lines.skip=TRUE`

Examples

```
## read a excel sheet
sheet = read_sheet(system.file("extdata/example.xlsx", package = "params"))

## read a comma separated sheet
```

```
csv = read_sheet(system.file("extdata/example.csv", package = "params"))

## read a tab separate sheet
tsv = read_sheet(system.file("extdata/example.tsv", package = "params"))


# write sheets -------

## Not run:
# throws a R CMD check note - don't run
## write a comma separated sheet
write_sheet(sheet, "example.csv")

## write a tab separated sheet
write_sheet(sheet, "example.tsv")

## write an excel separated sheet
write_sheet(sheet, "example.xlsx")

## End(Not run)
```

Index