Package 'configr'

October 13, 2026

Type Package

Title An Implementation of Parsing and Writing Configuration File (JSON/INI/YAML/TOML)

Version 0.3.6

Description Implements the JSON, INI, YAML and TOML parser for R setting and writing of configuration file. The functionality of this package is similar to that of package 'config'.

Depends R (>= 3.3.0)

URL https://github.com/Miachol/configr

BugReports https://github.com/Miachol/configr/issues

License MIT + file LICENSE

Encoding UTF-8

Imports jsonlite (>= 1.2), ini (>= 0.2), yaml (>= 2.1.3), RcppTOML (>= 0.1.3), stringr (>= 1.2.0), utils, glue

Suggests testthat, knitr, rmarkdown, prettydoc

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Jianfeng Li [aut, cre] (ORCID: <https://orcid.org/0000-0003-2349-208X>)

Maintainer Jianfeng Li <lee_jianfeng@sjtu.edu.cn>

Repository CRAN

Date/Publication 2026-10-13 04:40:02 UTC

Contents

config.help 2
config.list.merge 3
config.sections.del 3
configr 4
convert.config 5

eval.config 6
eval.config.merge 7
eval.config.sections 8
fetch.config 8
get.config.type 9
is.config.active 10
is.configfile.active 11
is.ini.file 11
is.json.file 12
is.toml.file 13
is.yaml.file 14
parse.extra 15
read.config 16
str2config 18
write.config 18

Index **20**

`config.help` *Function to access external helps about configurations format or other related information*

Description

Function to access external helps about configurations format or other related information

Usage

```
config.help(name = NULL, show_all_names = FALSE)
```

Arguments

`name`	Name or number of helps
`show_all_names`	Show all urls name

Examples

```
 config.help()
## Not run:
 config.help(1)
 config.help('ini_git_search')

## End(Not run)
```

`config.list.merge` *Merge list file (From config package), list.right will overwrite the element also existed in list.left*

Description

Merge list file (From config package), list.right will overwrite the element also existed in list.left

Usage

```
config.list.merge(list.left = list(), list.right = list())
```

Arguments

`list.left`	One list be merged left
`list.right`	One list be merged right

Value

A list

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
list.left <- list()
list.right <- eval.config(file = config.json)
config.list.merge(list.left, list.right)
list.left <- list(a=c(123,456))
list.right <- list(a=c(4,5,6))
config.list.merge(list.left, list.right)
```

`config.sections.del` *Delete sections in config, just do config[sections] <- NULL*

Description

Delete sections in config, just do config[sections] <- NULL

Usage

```
config.sections.del(config, sections)
```

Arguments

`config`	a list of config (eg. generated by read.config)
`sections`	Sections that need to be deleted

Value

A list of config

Examples

```
config.json <- system.file('extdata', 'config.json', package = 'configr')
config <- read.config(config.json, file.type = 'json')
config <- config.sections.del(config, 'default')
```

---

configr *configr package*

Description

Implements the YAML parser, JSON parser, INI parser and TOML parser for R setting and writing of configuration file.

See Also

Useful links:

https://github.com/Miachol/configr

Report bugs at https://github.com/Miachol/configr/issues

Examples

```
example.toml <- system.file('toml', 'example.toml', package='RcppTOML')
is.toml <- is.toml.file(example.toml)
file.type <- get.config.type(example.toml)
toml.list.raw <- read.config(example.toml)
owner.config <- eval.config(file = example.toml, config = 'owner')
owner.config.name <- eval.config(value = 'name', file = example.toml, config = 'owner')
toml.sections <- eval.config.sections(example.toml)
toml.merged.all <- eval.config.merge(example.toml)
toml.merged.selected <- eval.config.merge(example.toml, sections = c('database', 'owner'))

others <- list(others = list(lastupdate='2017-01-07'))
toml.list.update <- config.list.merge(toml.list.raw, others)
```

convert.config *Covert configuration file from JSON/INI/YAML/TOML to JSON/INI/YAML*

Description

Covert configuration file from JSON/INI/YAML/TOML to JSON/INI/YAML

Usage

```
convert.config(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  out.file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  convert.to = "ini",
  ...
)
```

Arguments

file	File name of configuration file to read from. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
out.file	Output path of configuration file. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist)
convert.to	JSON/INI/YAML
...	Arguments for read.config and write.config

Value

Logical indicating whether convert success

See Also

fromJSON JSON file will read by this

read.ini INI file will read by this

yaml.load_file YAML file will read by this

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config <- convert.config(file=config.json, out.file = sprintf('%s/config.ini', tempdir()))
```

`eval.config` *Read from the currently active configuration (JSON/INI/YAML/TOML be supported), 'retreiving either a single named value or all values as a config obj which have 'config', 'configtype', 'file' 'property.*

Description

Read from the currently active configuration (JSON/INI/YAML/TOML be supported), 'retreiving either a single named value or all values as a config obj which have 'config', 'configtype', 'file' 'property.

Usage

```
eval.config(
  value = NULL,
  config = Sys.getenv("R_CONFIG_ACTIVE", "default"),
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  ...
)
```

Arguments

`value`	Name of value (NULL to read all values)
`config`	Name of configuration to read from. Default is the value of 'the R_CONFIG_ACTIVE environment variable (Set to 'default' if the variable does not exist).
`file`	File name of configuration file to read from. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
`...`	Arguments for `read.config`

Value

Either a single value or all values as a list or logical FALSE indicating that is not standard JSON/INI/YAML/TOML format file

See Also

`read.config` read config by this

`eval.config.merge` which can merge multiple parameter sets by sections

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config <- eval.config(file=config.json)
config.extra.parsed.1 <- eval.config(file = config.json, extra.list = list(debug = 'TRUE'))
other.config <- system.file('extdata', 'config.other.yaml', package='configr')
config.extra.parsed.2 <- eval.config(file = config.json, extra.list = list(debug = 'TRUE'),
other.config = other.config)
```

`eval.config.merge` *Merge config parameter sets by sections.*

Description

Merge config parameter sets by sections.

Usage

```
eval.config.merge(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  sections = NULL,
  ...
)
```

Arguments

`file`	File name of configuration file to read from. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
`sections`	Need be merged parameter sets, eg. sections=c('default', 'version'), will default to all of config sections
`...`	Arguments for `get.config.type`, `eval.config.sections`, `eval.config`

Value

A list or logical FALSE indicating that is not standard JSON/INI/YAML/TOML format file

See Also

`eval.config.sections` which only get all of the mainly parameter sets name in config file, `read.config` which only read from a config as a list, `eval.config` which only read one sections as config obj or a value from config file.

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
eval.config.merge(config.json)
eval.config.merge(config.ini)
eval.config.merge(config.yaml)
eval.config.merge(config.toml)
```

eval.config.sections *Get config file parameter sections*

Description

Get config file parameter sections

Usage

```
eval.config.sections(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  ...
)
```

Arguments

file	File name of configuration file to read from. Default is the value of the 'R_CONFIGFILE_ACTIVE' environment variable (Set to 'config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
...	Arguments for read.config

Value

a character vector including the sections infomation of configure file or logical FALSE indicating that is not standard JSON/INI/YAML/TOML format file

See Also

eval.config.merge use this function to get all of sections of config file.

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
eval.config.sections(config.json)
```

fetch.config *Fetch configuration file and generate a merged list*

Description

Fetch configuration file and generate a merged list

Usage

```
fetch.config(
  links,
  return.files = FALSE,
  destdir = normalizePath("./"),
  keep.basename = TRUE,
  ...
)
```

Arguments

links	Url or files path that need to be merged, e.g. /tmp/config, http://, ftp://.
return.files	Only save the links configuration files to destdir and not to read and merge the configuration files, default is FALSE
destdir	Fetch configuration files and copy to this directory, default is ./
keep.basename	Whather use the links basename as the saved name or use paste0(tempfile(), '_configr')
...	Extra parameters pass to read.config

Value

A list or a vector

Examples

```
links <- c(paste0('https://raw.githubusercontent.com/JhuangLab',
'/BioInstaller/master/inst/extdata/config/db/db_annovar.toml'),
paste0('https://raw.githubusercontent.com/JhuangLab/BioInstaller',
'/master/inst/extdata/config/db/db_main.toml'),
system.file('extdata', 'config.toml', package = 'configr'))
x <- fetch.config(links)
```

---

get.config.type *Get config file type retreiving json/ini/yaml or FALSE*

Description

Get config file type retreiving json/ini/yaml or FALSE

Usage

```
get.config.type(file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"), ...)
```

Arguments

file	File name of configuration file to test. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
...	Arguments for is.json.file, is.ini.file, is.yaml.file, is.toml.file

Value

Character json/ini/yaml/toml or Logical FALSE indicating that is not standard JSON/INI/YAML/TOML format file

See Also

is.json.file, is.ini.file, is.yaml.file, is.toml.file

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
get.config.type(file=config.json)
get.config.type(file=config.ini)
get.config.type(file=config.yaml)
get.config.type(file=config.toml)
```

---

is.config.active *Test active configuration*

Description

Check whether a configuration group is currently active

Usage

```
is.config.active(config)
```

Arguments

config	Configuration name

Value

Logical indicating whether the specified configuration is active

Examples

```
is.config.active('default')
```

is.configfile.active *Test active configuration file*

Description

Check whether a configuration file is currently active

Usage

```
is.configfile.active(config.file)
```

Arguments

config.file	Configuration filename

Value

Logical indicating whether the specified configuration file is active

Examples

```
is.configfile.active('config.cfg')
```

is.ini.file *Function to check whether file is INI format*

Description

Function to check whether file is INI format

Usage

```
is.ini.file(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  ini.file.debug = FALSE,
  ...
)
```

Arguments

file	File name of configuration file to test. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
ini.file.debug	If TRUE, it will show error infomation when read INI config, default is FALSE
...	Arguments for read.ini

Value

Logical indicating whether the specified configuration file is INI format

See Also

`is.json.file`, `is.yaml.file`, `is.toml.file`

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
print(is.ini.file(config.ini))
print(is.ini.file(config.json))
print(is.ini.file(config.yaml))
print(is.ini.file(config.toml))
```

---

is.json.file *Function to check whether file is JSON format*

Description

Function to check whether file is JSON format

Usage

```
is.json.file(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  json.file.debug = FALSE,
  ...
)
```

Arguments

`file`	File name of configuration file to test. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
`json.file.debug`	If TRUE, it will show error infomation when read JSON config, default is FALSE
`...`	Arguments for `readLines` and `fromJSON`

Value

Logical indicating whether the specified configuration file is JSON format

See Also

is.ini.file, is.yaml.file, is.toml.file

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
print(is.json.file(config.json))
print(is.json.file(config.ini))
print(is.json.file(config.yaml))
print(is.json.file(config.toml))
```

---

is.toml.file *Function to check whether file is TOML format*

Description

Function to check whether file is TOML format

Usage

```
is.toml.file(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  toml.file.debug = FALSE,
  ...
)
```

Arguments

file	File name of configuration file to test. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
toml.file.debug	If TRUE, it will show error infomation when read TOML config, default is FALSE
...	Arguments for parseTOML

Value

Logical indicating whether the specified configuration file is TOML format

See Also

is.json.file is.ini.file, is.yaml.file

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
print(is.toml.file(config.json))
print(is.toml.file(config.ini))
print(is.toml.file(config.yaml))
print(is.toml.file(config.toml))
```

---

is.yaml.file *Function to check whether file is YAML format*

Description

Function to check whether file is YAML format

Usage

```
is.yaml.file(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  yaml.file.debug = FALSE,
  ...
)
```

Arguments

file	File name of configuration file to test. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
yaml.file.debug	If TRUE, it will show error infomation when read YAML config, default is FALSE
...	Arguments for is.json.file, readLines and yaml.load

Value

Logical indicating whether the specified configuration file is YAML format

See Also

is.json.file, is.ini.file, is.toml.file

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.ini <- system.file('extdata', 'config.ini', package='configr')
config.yaml <- system.file('extdata', 'config.yaml', package='configr')
config.toml <- system.file('extdata', 'config.toml', package='configr')
print(is.yaml.file(config.yaml))
print(is.yaml.file(config.json))
print(is.yaml.file(config.ini))
print(is.yaml.file(config.toml))
```

`parse.extra` *Parse the configuration* `{{var}}` *format, and replace it by extra.list values*

Description

Parse the configuration `{{var}}` format, and replace it by extra.list values

Usage

```
parse.extra(
  config,
  extra.list = list(),
  other.config = "",
  rcmd.parse = FALSE,
  bash.parse = FALSE,
  glue.parse = FALSE,
  glue.flag = "!!glue",
  global.vars.field = "global_vars"
)
```

Arguments

`config`	A list that were generated by read.config/eval.config/eval.config.merge
`extra.list`	A list that can replace the configuration file `{{debug}}` by list(debug = TRUE), and `{{debug}}` will be setted to TRUE
`other.config`	Path of another configuration file that can replace the configuration file `{{key:value}}`
`rcmd.parse`	Logical whether parse '@>@str_replace('abc', 'b', 'c')@<@' in config to 'acc'
`bash.parse`	Logical whether parse '#>#echo $HOME#<#' in config to your HOME PATH
`glue.parse`	Logical whether parse `'!!glue{1:5}'` in config to ['1','2','3','4','5']; ['nochange', `'!!glue{1:5}'`, 'nochange'] => ['nochange', '1', '2', '3', '4', '5', 'nochange']
`glue.flag`	A character flag indicating wheater run glue() function to parse (Default is !!glue)
`global.vars.field`	All vars defined in global.vars.field will as the extra.list params [gloval_var]

Value

A list

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config.other <- system.file('extdata', 'config.other.yaml', package='configr')
config <- read.config(config.json)
parse.extra(config, list(debug = 'TRUE'))
parse.extra(config, list(debug = 'TRUE'), other.config = config.other)
parse.extra(config, list(debug = 'TRUE'), other.config = config.other,
rcmd.parse = TRUE)
parse.extra(config, list(debug = 'TRUE'), other.config = config.other,
rcmd.parse = TRUE, bash.parse = TRUE)

raw <- c('a', '!!glue{1:5}', 'c')
expect.parsed.1 <- c('a', '1', '2', '3', '4', '5', 'c')
list.raw <- list(glue = raw, nochange = 1:10)
parsed <- parse.extra(list.raw, glue.parse = TRUE)

raw <- c('!!glue_numeric{1:5}')
expect.parsed.1 <- c(1, 2, 3, 4, 5)
list.raw <- list(glue = raw, nochange = 1:10)
parsed <- parse.extra(list.raw, glue.parse = TRUE)
```

`read.config` *Read from the file (JSON/INI/YAML/TOML be supported), retreiving all values as a list.*

Description

Read from the file (JSON/INI/YAML/TOML be supported), retreiving all values as a list.

Usage

```
read.config(
  file = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  extra.list = list(),
  other.config = "",
  rcmd.parse = FALSE,
  bash.parse = FALSE,
  glue.parse = FALSE,
  glue.flag = "!!glue",
  global.vars.field = "global_vars",
  file.type = NULL,
  ...
)
```

Arguments

`file`	File name of configuration file to read from. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist and JSON/INI/YAML/TOML format only)
`extra.list`	A list that can replace the configuration file `{{debug}}` by list(debug = 'TRUE'), and `{{debug}}` will be setted to 'TRUE'
`other.config`	Path of another configuration file that can replace the configuration file `{{key:value}}`
`rcmd.parse`	Logical whether parse '@>@str_replace('abc', 'b', 'c')@<@' that existed in config to 'acc'
`bash.parse`	Logical whether parse '#>#echo $HOME#<#' in config to your HOME PATH
`glue.parse`	Logical whether parse `'!!glue{1:5}'` in config to ['1','2','3','4','5']; ['nochange', `'!!glue{1:5}'`, 'nochange'] => ['nochange', '1', '2', '3', '4', '5', 'nochange']
`glue.flag`	A character flage indicating wheater run glue() function to parse (Default is !!glue)
`global.vars.field`	All vars defined in global.vars.field will as the extra.list params [gloval_var]
`file.type`	Default is no need to specify the variable, file.type will be automatically identify by `get.config.type`. If the value be specified, the step of filetype identification will be skipped.
`...`	Arguments for `get.config.type`, `fromJSON`, `read.ini`, `yaml.load_file`, `parseTOML`, `readLines`

Value

All values as a list or logical FALSE indicating that is not standard JSON/INI/YAML/TOML format file

See Also

`fromJSON` JSON file will read by this

`read.ini` INI file will read by this

`yaml.load_file` YAML file will read by this

`parseTOML` TOML file will read by this

Examples

```
config.json <- system.file('extdata', 'config.json', package='configr')
config <- read.config(file=config.json)
config.extra.parsed.1 <- read.config(config.json, list(debug = 'TRUE'))
other.config <- system.file('extdata', 'config.other.yaml', package='configr')
config.extra.parsed.2 <- read.config(config.json, list(debug = 'TRUE'), other.config)
```

str2config *Parse configuration string to R list object.*

Description

Parse configuration string to R list object.

Usage

```
str2config(text, ...)
```

Arguments

text	JSON, YAML, INI or TOML format string.
...	Arguments pass to read.config

Value

List

Examples

```
json_string <- '{"city" : "Crich"}\n'
yaml_string <- 'foo: 123\n'
json_config <- str2config(json_string)
yaml_config <- str2config(yaml_string)
```

write.config *Write config in a file (JSON/YAML/INI)*

Description

Write config in a file (JSON/YAML/INI)

Usage

```
write.config(
  config.dat,
  file.path = Sys.getenv("R_CONFIGFILE_ACTIVE", "config.cfg"),
  write.type = "ini",
  sections = NULL,
  ...
)
```

Arguments

config.dat	a list of config (eg. generated by read.config)
file.path	File path of configuration to write. Defaults to the value of the 'R_CONFIGFILE_ACTIVE' environment variable ('config.cfg' if the variable does not exist)
write.type	json/ini/yaml
sections	Sections that need be write in file, default is NULL and use all of sections
...	Arguments for write.ini, prettify, toJSON, as.yaml and cat encoding if not specifield

Value

Logical indicating whether the specified configuration file be writed successful

See Also

toJSON convert a list to JSON string, prettify convert a JSON string to user friendly view, write.ini write a list in a INI format file, as.yaml convert a list to YAML format

Examples

```
list.test <- list(a=c(123,456))
out.yaml <- sprintf('%s/test.yaml', tempdir())
write.config(list.test, out.yaml, write.type = 'yaml')
```

Index

as.yaml, *19*

cat, *19*
config.help, 2
config.list.merge, 3
config.sections.del, 3
configr, 4
convert.config, 5

eval.config, 6, *7*
eval.config.merge, *6*, 7, *8*
eval.config.sections, *7*, 8

fetch.config, 8
fromJSON, *5*, *12*, *17*

get.config.type, *7*, 9, *17*

is.config.active, 10
is.configfile.active, 11
is.ini.file, *10*, 11, *13, 14*
is.json.file, *10*, *12*, 12, *13, 14*
is.toml.file, *10*, *12, 13*, 13, *14*
is.yaml.file, *10*, *12, 13*, 14

parse.extra, 15
parseTOML, *13*, *17*
prettify, *19*

read.config, *5–8*, 16, *18*
read.ini, *5*, *11*, *17*
readLines, *12*, *14*, *17*

str2config, 18

toJSON, *19*

write.config, *5*, 18
write.ini, *19*

yaml.load, *14*
yaml.load_file, *5*, *17*