

Package ‘rconfig’

February 11, 2023

Type Package

Title Manage R Configuration at the Command Line

Version 0.2.0

Date 2023-02-10

Description Configuration management using files (JSON, YAML, separated text), JSON strings, and command line arguments. Command line arguments can be used to override configuration. Period-separated command line flags are parsed as hierarchical lists. Environment variables, R global variables, and configuration values can be substituted.

License MIT + file LICENSE

LazyLoad yes

Imports yaml, jsonlite

RoxygenNote 7.2.1

Encoding UTF-8

BugReports <https://github.com/analythium/rconfig/issues>

URL <https://github.com/analythium/rconfig>

Language en-US

NeedsCompilation no

Author Peter Solymos [aut, cre] (<<https://orcid.org/0000-0001-7337-1740>>),
Analythium Solutions Inc. [cph, fnd]

Maintainer Peter Solymos <peter@analythium.io>

Repository CRAN

Date/Publication 2023-02-11 08:20:02 UTC

R topics documented:

| | |
|-------------------|----------|
| rconfig | 2 |
| Index | 6 |

rconfig

Manage R Configuration at the Command Line

Description

Manage R configuration using files (JSON, YAML, separated text) JSON strings and command line arguments. Command line arguments can be used to provide commands and to override configuration. Period-separated command line flags are parsed as hierarchical lists.

Usage

```
rconfig(
  file = NULL,
  list = NULL,
  eval = NULL,
  flatten = NULL,
  debug = NULL,
  sep = NULL,
  sub = NULL,
  ...
)

value(x, ...)

## Default S3 method:
value(x, default = NULL, coerce = TRUE, ...)

command(x, ...)

## Default S3 method:
command(x, ...)
```

Arguments

| | |
|---------|--|
| file | Configuration file name or URL (NULL to not use this configuration file to override the default behavior). Can be a vector, in which case each element will be treated as a configuration file, and these will be parsed and applied in the order they appear. |
| list | A list to override other configs (NULL to not use this list to override the default behavior). This argument is treated as a single configuration (as opposed to file). List names need to be unique. |
| eval | Logical, evaluate !expr R expressions. |
| flatten | Logical, should config contain nested lists or should results be flat, i.e. a\$b\$c to flattened into the key a.b.c; like <code>unlist()</code> but returning a list and preserving the value types. |

| | |
|---------|--|
| debug | Logical, when debug mode is on the configuration source information are attached as the "trace" attribute. |
| sep | Character, separator for text files. |
| sub | Logical, substitute environment variables (see Details). |
| ... | Other arguments passed to file parsers: <code>yaml::yaml.load_file()</code> for YAML, <code>jsonlite::fromJSON()</code> for JSON, and <code>utils::read.table()</code> for text files. |
| x | A list, e.g. the <code>rconfig()</code> output. |
| default | A default value to be used when a configuration entry is not set. |
| coerce | Logical, should values of <code>x</code> coerced to the same type as <code>storage.mode(default)</code> . |

Details

Merges configuration after parsing files, JSON strings, and command line arguments. Note that `rconfig` only considers trailing command line arguments from Rscript. `rconfig` differentiates verb/noun syntax, where verbs are sub-commands following the R script file name and preceding the command line flags (starting with `-` or `--`). Configurations are merged in the following order (key-values from last element override previous values for the same key):

1. `R_RCONFIG_FILE` value or "rconfig.yml" from working directory
2. JSON strings (following `-j` and `--json` flags) and files (following `-f` and `--file` flags) provided as command line arguments are parsed and applied in the order they appear (key-value pairs are separated by space, only atomic values considered, i.e. file name or string) for each flag, but multiple file/JSON flags are accepted in sequence
3. the remaining other command line arguments, that can be sub-commands or command line flags (starting with `-` or `--`), period-separated command line flags are parsed as hierarchical lists (key-value pairs are separated by space, flags must begin with `--`, values are treated as vectors when contain spaces, i.e. `--key 1 2 3`)
4. configuration from the `file` argument (one or multiple files, parsed and applied in the order they appear)
5. configuration from the `list` argument

The following environment variables and options can be set to modify the default behavior:

- `R_RCONFIG_FILE`: location of the default configuration file, it is assumed to be `rconfig.yml` in the current working directory. The file name can be an URL or it can be missing.
- `R_RCONFIG_EVAL`: coerced to logical, indicating whether R expressions starting with `!expr` should be evaluated in the namespace environment for the base package (overrides the value of `getOption("rconfig.eval")`). When not set the value assumed is `TRUE`.
- `R_RCONFIG_SUB`: coerced to logical, indicating whether environment variables should be substituted (overrides the value of `getOption("rconfig.sub")`). When not set the value assumed is `TRUE`.
- `R_RCONFIG_FLATTEN`: coerced to logical, flatten nested lists, i.e. `abc` becomes the key `a.b.c` (overrides the value of `getOption("rconfig.flatten")`). When not set the value assumed is `FALSE`.
- `R_RCONFIG_DEBUG`: coerced to logical, to turn on debug mode (overrides the value of `getOption("rconfig.debug")`). When not set the value assumed is `FALSE`.

- `R_RCONFIG_SEP`: separator for text file parser, (overrides the value of `getOption("rconfig.sep")`). When not set the value assumed is `"="`.

When the configuration is a file (file name can also be a URL), it can be nested structure in JSON or YAML format. Other text files are parsed using the separator (`R_RCONFIG_SEP` or `getOption("rconfig.sep")`) and period-separated keys are parsed as hierarchical lists (i.e. `a.b.c=12` is treated as `abc = 12`) by default.

When the configuration is a file or a JSON string, values starting with `!expr` will be evaluated depending on the settings `R_RCONFIG_EVAL` and `getOption("rconfig.eval")`. E.g. `cores: !expr getOption("mc.cores")` etc.

The rconfig package interprets 3 kinds of substitution patterns:

- environment variables (`${VALUE}`): these variables are already present when the configuration is read from the calling environment or from `.Renviron` file in the project specific or home folder, set variables can be null or not-null
- R global variables (`@{VALUE}`): the rconfig package looks for variables in the global environment at the time of configuration evaluation, however, expressions are not evaluated (unlike the `!expr` option for values)
- configuration values (`#{VALUE}`): the configuration level variables are evaluated last, thus these values can refer to existing keys that are already substituted

For additional details see the package website at <https://github.com/analythium/rconfig>.

Value

The configuration value (a named list, or an empty list). When debug mode is on, the `"trace"` attribute traces the merged configurations. The `value()` method returns the value of a given argument or the default value when it is not found (i.e. `NULL`). The `command()` method returns a character vector with command line sub-commands (can be of length 0).

See Also

[utils::modifyList\(\)](#)

Examples

```
cfile <- function(file) {
  system.file("examples", file, package = "rconfig")
}

rconfig::rconfig()

rconfig::rconfig(
  file = cfile("rconfig.yml"))

rconfig::rconfig(
  file = c(cfile("rconfig.json"),
          cfile("rconfig-prod.txt")),
  list = list(user = list(name = "Jack")))
```

```
rconfig::rconfig(  
  file = c(cfile("rconfig.json"),  
           cfile("rconfig-prod.txt")),  
  list = list(user = list(name = "Jack")),  
  flatten = TRUE)  
  
CONFIG <- rconfig::rconfig(  
  file = cfile("rconfig.yml"))  
value(CONFIG$cores, 2L) # set to 1L  
value(CONFIG$test, FALSE) # unset
```

Index

* **models**

 rconfig, 2

* **regression**

 rconfig, 2

command (rconfig), 2

jsonlite::fromJSON(), 3

rconfig, 2

unlist(), 2

utils::modifyList(), 4

utils::read.table(), 3

value (rconfig), 2

yaml::yaml.load_file(), 3