

Package: ronfig (via r-universe)

October 4, 2026

Type Package

Title Load Configuration Values

Version 0.0.10

Description A simple approach to configuring R projects with different parameter values. Configurations are specified using a reduced subset of base R and parsed accordingly.

URL <https://timtaylor.codeberg.page/ronfig/>

BugReports <https://codeberg.org/TimTaylor/ronfig/issues>

License GPL-2 | GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports carrier, cli, utils

Suggests litedown, stats, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder litedown

Config/roxygen2/version 8.0.0

Repository <https://timtaylor.r-universe.dev>

Date/Publication 2026-05-07 12:18:50 UTC

RemoteUrl <https://codeberg.org/TimTaylor/ronfig>

RemoteRef HEAD

RemoteSha 74a96920f5362093fe96d25647244f671114763b

RemoteSubdir pkg

Contents

load_config	2
Index	5

load_config

*Load configuration***Description**

Load a user defined configuration from file. By default (i.e. when `as_is = FALSE`), `load_config()` requires inputs to be given as uniquely-named lists. It first parses the configuration file looking for a 'default' entry. With no additional arguments this will be returned as a list to the user. If the user specifies an additional list to consider (via the `config` argument) then this list is layered on top (using `utils::modifyList()`).

`list_config()` returns the names of (possible) configurations within the given file. It corresponds to `names(load_config(filename, as_is = TRUE))`.

Usage

```
load_config(
  filename,
  config,
  crates,
  ...,
  as_is = FALSE,
  default = "default",
  strict = TRUE,
  allow_null = TRUE
)
```

```
list_config(filename, strict = TRUE, allow_null = TRUE)
```

Arguments

<code>filename</code>	Configuration file to read from.
<code>config</code>	If <code>as_is = FALSE</code> , name of entry in configuration file to layer on top of 'default'. If <code>as_is = TRUE</code> then, if specified, this entry will be selected from the configuration and returned.
<code>crates</code>	A list of <code>carrier::crate</code> objects which are used to inject functions in to the environment where the configuration file will be evaluated.
<code>...</code>	Not currently used.
<code>as_is</code>	Should the configuration file be read in as is, without layering on top of the default config? Defaults to <code>FALSE</code> .
<code>default</code>	The default configuration to use. Not used if <code>as_is = FALSE</code> .
<code>strict</code>	If <code>TRUE</code> then configuration files can be specified using only a reduced subset of base R as well as a handful of other functions. See details for more information.

`allow_null` If TRUE then configuration files can contain NULL values. Otherwise, NULL entries will trigger an error. This works recursively across lists but will not check classed lists (i.e. only lists where `is.object(x) == FALSE` are checked).

Details

If `strict` is TRUE (default), then configuration files can be specified using a reduced subset of base R. By default this is restricted to the following operators and functions:

- `<-`, `=`, `+`, `-`, `*`, `:`
- `$`, `[`, `[[`
- `$<-`, `[<-`, `[[<-`
- `c()`
- `as.Date()`
- `array()`, `matrix()`
- `list()`, `data.frame()`
- `Sys.Date()`, `Sys.time()`
- `seq()`, `sequence()`, and `seq_len()`
- `file.path()`
- `modifyList()`

We also enable a convenience function, `cc`, which automatically quotes input to save typing.

If `strict` is FALSE then the entire base namespace, as well as `cc` and `modifyList` are made available.

Users can also inject their own functions in to the evaluation environment by supplying a list of [crates](#) as an additional argument.

Value

If `as_is` = FALSE (default), `load_config()` returns a list contain entries corresponding to the chosen config.

If `as_is` = TRUE and no config value specified, a list of all entries in the evaluated configuration file. If a config value is specified, this entry is pulled from the list and returned.

`list_config()` returns a character vector of (possible) configurations within the given file. It corresponds to `names(load_config(filename, as_is = TRUE))`.

Examples

```
# load the example configuration
file <- system.file("config.R", package = "ronfig")
cat(readChar(file, file.info(file)$size))

# default configuration
str(load_config(file))
```

```
# debug configuration
str(load_config(file, "debug"))

# forecast configuration
str(load_config(file, "forecast"))

# Injecting crated function
f <- tempfile()
cat("default <- list(a=mean(1:10))", file = f)

# will fail as mean() not available
tryCatch(
  with(load_config(f), a),
  error = function(cnd) cat(conditionMessage(cnd))
)

# will work if we inject crated mean
crate <- carrier::crate(function(x) mean(x))
with(load_config(f, crates = list(mean = crate)), a)

unlink(f)
```

Index

`carrier::crate`, [2](#)
`crates`, [3](#)

`list_config (load_config)`, [2](#)
`load_config`, [2](#)