

Package ‘versioning’

May 8, 2026

Title Settings and File I/O using a Configuration YAML File

Version 0.2.0

Description R data pipelines commonly require reading and writing data to versioned directories. Each directory might correspond to one step of a multi-step process, where that version corresponds to particular settings for that step and a chain of previous steps that each have their own versions. This package creates a configuration object that makes it easy to read and write versioned data, based on YAML configuration files loaded and saved to each versioned folder.

Depends R (>= 4.1.0)

Imports assertthat, glue, R6, yaml

Suggests data.table, foreign, haven, readxl, sf, terra, knitr, rmarkdown, testthat (>= 3.0.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Nathaniel Henry [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-8150-4988>>)

Maintainer Nathaniel Henry <nat@henryspatialanalysis.com>

Repository CRAN

Date/Publication 2025-03-10 23:50:01 UTC

Contents

autoread	2
autowrite	2
Config	3
get_file_reading_functions	6
get_file_writing_functions	7

pull_from_list	7
qstop	8
require_namespace_or_stop	8

Index	9
--------------	----------

autoread	<i>Auto-read from file</i>
----------	----------------------------

Description

Automatically read a file based on extension

Usage

```
autoread(file, ...)
```

Arguments

file	Full path to be read
...	Other arguments to be passed to the particular loading function

Value

The object loaded by the file

See Also

[get_file_reading_functions\(\)](#) [autowrite\(\)](#)

autowrite	<i>Auto-write to file</i>
-----------	---------------------------

Description

Automatically write an object to a file based on extension

Usage

```
autowrite(x, file, ...)
```

Arguments

x	Object to be saved
file	Full path to save the object to
...	Other arguments to be passed to the particular saving function

Value

Invisibly passes TRUE if the file saves successfully

See Also

[get_file_writing_functions\(\)](#) [autoread\(\)](#)

Config

R6 Class representing a configuration object

Description

R6 Class representing a configuration object

R6 Class representing a configuration object

Details

The special sublist `directories` is structured to contain three items for each directory name:

- `versioned`: a T/F value specifying whether the directory is versioned
- `path`: the full path to the top level of that directory.
- `files`: A named list referencing file paths within that directory.

If the directory is versioned, a version must be set in the `versions` sublist of the `config` list. `versions` is itself a named list where each key corresponds to a versioned folder in `directories` and the value gives the particular folder version (for example, a timestamp) that corresponds to the particular run.

Public fields

`config_list` The list representation of the Config object

Methods**Public methods:**

- [Config\\$new\(\)](#)
- [Config\\$print\(\)](#)
- [Config\\$get\(\)](#)
- [Config\\$get_dir_path\(\)](#)
- [Config\\$get_file_path\(\)](#)
- [Config\\$read\(\)](#)
- [Config\\$write\(\)](#)
- [Config\\$write_self\(\)](#)
- [Config\\$clone\(\)](#)

Method new(): Create a new Config object

Usage:

```
Config$new(config_list, versions = NULL)
```

Arguments:

`config_list` either a list or a filepath to a YAML file containing that list
`versions` (default NULL) A named list containing versions for versioned directories. If passed, used to define or update items in `config_list$versions`.

Method print(): Print the list representation of the Config object

Usage:

```
Config$print()
```

Method get(): Get a subset of the `config_list`

Usage:

```
Config$get(...)
```

Arguments:

... Nested indices (character or numeric) down the config list

Details: If no parameters are passed, returns the entire `config_list`

Returns: A subset of the list. If the item is NULL or missing, returns an error

Method get_dir_path(): Construct a directory path from the config object

Usage:

```
Config$get_dir_path(
  dir_name,
  custom_version = NULL,
  fail_if_does_not_exist = FALSE
)
```

Arguments:

`dir_name` Directory name

`custom_version` (character, default NULL) A custom version that will be applied to this folder, rather than pulling from `config_list$versions[[dir]]`. Only applies to versioned folders.

`fail_if_does_not_exist` (logical, default FALSE) should this method return an error if the directory in question does not already exist?

Details: Works differently for versioned and non-versioned directories. See the class description for more information.

Returns: The full path to the directory

Method get_file_path(): Construct a file path from the config object

Usage:

```
Config$get_file_path(
  dir_name,
  file_name,
  custom_version = NULL,
  fail_if_does_not_exist = FALSE
)
```

Arguments:

`dir_name` Directory name

`file_name` File name within that directory

`custom_version` (character, default NULL) A custom version that will be applied to this folder, rather than pulling from `config_list$versions[[dir]]`. Only applies to versioned folders.

`fail_if_does_not_exist` (logical, default FALSE) should this method return an error if the directory in question does not already exist?

Details: Looks for the file path under: `config_list$directories[[dir_name]]$files[[file_name]]`

Returns: The full path to the file Read a file based on the config

Method `read()`:*Usage:*

`Config$read(dir_name, file_name, ..., custom_version = NULL)`

Arguments:

`dir_name` Directory name

`file_name` File name within that directory

`...` Optional file reading arguments to pass to `autoread()`

`custom_version` (character, default NULL) A custom version that will be applied to this folder, rather than pulling from `config_list$versions[[dir]]`. Only applies to versioned folders. If passed, this argument must always be explicitly named.

Returns: The object loaded by `autoread()` Write an object to file based on the config

Method `write()`:*Usage:*

`Config$write(x, dir_name, file_name, ..., custom_version = NULL)`

Arguments:

`x` Object to write

`dir_name` Directory name

`file_name` File name within that directory

`...` Optional file writing arguments to pass to `autowrite()`

`custom_version` (character, default NULL) A custom version that will be applied to this folder, rather than pulling from `config_list$versions[[dir]]`. Only applies to versioned folders. If passed, this argument must always be explicitly named.

Returns: Invisibly passes TRUE if successful Convenience function: write the config list to a folder as 'config.yaml'

Method `write_self()`:*Usage:*

`Config$write_self(dir_name, ..., custom_version = NULL)`

Arguments:

`dir_name` Directory name

... Optional file writing arguments to pass to `autowrite()`
`custom_version` (character, default NULL) A custom version that will be applied to this folder, rather than pulling from `config_list$versions[[dir]]`. Only applies to versioned folders. If passed, this argument must always be explicitly named.

Returns: Invisibly passes TRUE if successful

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Config$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[pull_from_list\(\)](#)

`get_file_reading_functions`

Get the list of file reading functions

Description

Constructs a list of all file-reading functions based on extension

Usage

```
get_file_reading_functions()
```

Value

Named list where the names are file extensions, and the values are functions that read a file. All functions have ... arguments that can be used to extend the basic function.

See Also

[autoread\(\)](#) [get_file_writing_functions\(\)](#)

`get_file_writing_functions`*Get the list of file writing functions*

Description

Constructs a list of all file-reading functions based on extension

Usage

```
get_file_writing_functions()
```

Value

Named list where the names are file extensions, and the values are functions that read a file. All functions have ... arguments that can be used to extend the basic function.

See Also

[autoread\(\)](#) [get_file_reading_functions\(\)](#)

`pull_from_list`*Safely pull an item from a list*

Description

Indexing function for a list

Usage

```
pull_from_list(x, ..., fail_if_null = TRUE)
```

Arguments

<code>x</code>	List to pull items from
<code>...</code>	List indices to pull. Can be either numeric or (preferably) a character.
<code>fail_if_null</code>	(logical, default TRUE). Returns an informative error message if the list index is NULL. This function must always be named.

Details

Use the ... arguments to index the list. Not passing any ... arguments will return the entire list. The indexing will fail if either of two conditions are met:

1. The index (which can be numeric or a key) does not exist in the list
2. If the index exists but the value of the item is NULL, and `fail_if_null` is TRUE

qstop

Stop without listing the containing function call

Description

Stop without listing the containing function call

Usage

qstop(...)

Arguments

... Parameters passed to stop()

Value

Concisely stops program execution

require_namespace_or_stop

Require that a namespace be loaded, or stop execution

Description

Require that a namespace be loaded, or stop execution

Usage

require_namespace_or_stop(pkg)

Arguments

pkg (character(1)) Package to be loaded

Value

Silently loads namespace, or stops execution if package cannot be loaded

Index

autoread, [2](#)
autoread(), [3](#), [5–7](#)
autowrite, [2](#)
autowrite(), [2](#), [5](#), [6](#)

Config, [3](#)

get_file_reading_functions, [6](#)
get_file_reading_functions(), [2](#), [7](#)
get_file_writing_functions, [7](#)
get_file_writing_functions(), [3](#), [6](#)

pull_from_list, [7](#)
pull_from_list(), [6](#)

qstop, [8](#)

require_namespace_or_stop, [8](#)