Package 'loggit'

July 22, 2025

Title Modern Logging for the R Ecosystem

Description An effortless 'ndjson' (newline-delimited 'JSON') logger, with two primary log-writing interfaces. It provides a set of wrappings for base R's message(), warning(), and stop() functions that maintain identical functionality, but also log the handler message to an 'ndjson' log file. 'loggit' also exports its internal 'loggit()' function for powerful and configurable custom logging. No change in existing code is necessary to use this package, and should only require additions to fully leverage the power of the logging system. 'loggit' also provides a log reader for reading an 'ndjson' log file into a data frame, log rotation, and live echo of the 'ndjson' log messages to terminal 'stdout' for log capture by external systems (like containers). 'loggit' is ideal for Shiny apps, data pipelines, modeling work flows, and more. Please see the vignettes for detailed example use cases.

Version 2.1.1

Date 2021-02-27

License MIT + file LICENSE

Depends R (>= 3.4.0)

Suggests knitr (>= 1.19), rmarkdown (>= 1.8), testthat (>= 2.0.0)

URL https://github.com/ryapric/loggit

BugReports https://github.com/ryapric/loggit/issues

RoxygenNote 7.1.1

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Ryan Price [cre, aut]

Maintainer Ryan Price <ryapric@gmail.com>

Repository CRAN

Date/Publication 2021-02-28 05:30:02 UTC

Contents

`get_logfile` *Get Log File*

Description

Return the log file that loggit will write to.

Usage

```
get_logfile()
```

Examples

```
get_logfile()
```

`get_timestamp_format` *Get Timestamp Format*

Description

Get timestamp format for use in output logs.

Usage

```
get_timestamp_format()
```

Examples

```
get_timestamp_format()
```

handlers *loggit's Exception Handlers*

Description

These exception handlers are identical to base R's message, warning, and stop, but with included logging of the exception messages via `loggit()`.

Usage

```
message(..., domain = NULL, appendLF = TRUE, .loggit = TRUE, echo = TRUE)

warning(
  ...,
  call. = TRUE,
  immediate. = FALSE,
  noBreaks. = FALSE,
  domain = NULL,
  .loggit = TRUE,
  echo = TRUE
)

stop(..., call. = TRUE, domain = NULL, .loggit = TRUE, echo = TRUE)
```

Arguments

`...`	zero or more objects which can be coerced to character (and which are pasted together with no separator) or (for `message` only) a single condition object.
`domain`	see `gettext`. If `NA`, messages will not be translated, see also the note in `stop`.
`appendLF`	logical: should messages given as a character string have a newline appended?
`.loggit`	Should loggit function execute? Defaults to `TRUE`.
`echo`	Should loggit's log entry be echoed to the console, as well? Defaults to `TRUE`.
`call.`	logical, indicating if the call should become part of the warning message.
`immediate.`	logical, indicating if the call should be output immediately, even if `getOption("warn") <= 0`.
`noBreaks.`	logical, indicating as far as possible the message should be output as a single line when `options(warn = 1)`.

Examples

```
if (2 < 1) message("Don't say such silly things!")

if (2 < 1) warning("You may want to review that math, and so this is your warning")

if (2 < 1) stop("This is a completely false condition, which throws an error")
```

`loggit` *Log entries to file*

Description

This function executes immediately before the function definitions for the base handler functions (message, warning, and stop, and logs their timestamped output (a bit more verbosely) to a log file. The log file is an ndjson file, which is a portable, JSON-based format that is easily parsed by many line-processing systems.

Usage

```
loggit(log_lvl, log_msg, ..., echo = TRUE, custom_log_lvl = FALSE, sanitizer)
```

Arguments

`log_lvl`	Level of log output. In actual practice, one of "DEBUG", "INFO", "WARN", and "ERROR" are common, but any string may be supplied if `custom_log_lvl` is TRUE. Will be coerced to class `character`.
`log_msg`	Main log message. Will be coerced to class `character`.
`...`	A named `list` or named `vector` (each element of length one) of other custom fields you wish to log. You do not need to explicitly provide these fields as a formal list or vector, as shown in the example; R handles the coercion.
`echo`	Should the log file entry be printed to the console as well? Defaults to `TRUE`, and will print out the `ndjson` line to be logged. This argument is passed as `FALSE` when called from `loggit`'s handlers, since they still call base R's handlers at the end of execution, all of which print to the console as well.
`custom_log_lvl`	Allow log levels other than "DEBUG", "INFO", "WARN", and "ERROR"? Defaults to `FALSE`, to prevent possible typos by the developer, and to limit the variation in structured log contents. Overall, setting this to 'TRUE" is not recommended, but is an option for consistency with other frameworks the user may work with.
`sanitizer`	Sanitizer function to run over elements in log data. The default sanitizer, if not specified, is `default_ndjson_sanitizer()`. See the sanitizers documentation for information on how to write your own (un)sanitizer functions.

Examples

```
loggit("INFO", "This is a message", but_maybe = "you want more fields?",
sure = "why not?", like = 2, or = 10, what = "ever")
```

read_logs *Return log file as an R data frame*

Description

This function returns a `data.frame` containing all the logs in the provided `ndjson` log file. If no explicit log file is provided, calling this function will return a data frame of the log file currently pointed to by the `loggit` functions.

Usage

```
read_logs(logfile, unsanitizer)
```

Arguments

`logfile`	Path to log file. Will default to currently-set log file.
`unsanitizer`	Unsanitizer function to use. For more info on sanitizers, please see the sanitizers section of the package documentation.

Value

A `data.frame`.

Examples

```
set_logfile(file.path(tempdir(), "loggit.log"), confirm = FALSE)
message("Test log message")
read_logs()
```

read_ndjson *Read ndJSON-formatted log file*

Description

Read ndJSON-formatted log file

Usage

```
read_ndjson(logfile, unsanitizer)
```

Arguments

`logfile`	Log file to read from, and convert to a `data.frame`.
`unsanitizer`	Unsanitizer function passed in from `read_logs()`.

Value

A `data.frame`

`rotate_logs` *Rotate log file*

Description

Truncates the log file to the line count provided as `rotate_lines`.

Usage

```
rotate_logs(rotate_lines = 1e+05, logfile)
```

Arguments

`rotate_lines`	The number of log entries to keep in the logfile. Defaults to 100,000.
`logfile`	Log file to truncate. Defaults to the currently-configured log file.

Details

`loggit` makes no assumptions nor enforcement of calling this function; that is to say, the onus of log rotation is up to the developer. You

Examples

```
# Truncate "default" log file to 100 lines
set_logfile()
for (i in 1:150) {loggit("INFO", i, echo = FALSE)}
rotate_logs(100)

# Truncate a different log file to 250 lines
another_log <- file.path(tempdir(), "another.log")
set_logfile(another_log)
for (i in 1:300) {loggit("INFO", i, echo = FALSE)}
set_logfile() # clears pointer to other log file
rotate_logs(250, another_log)
```

`sanitizers` *Default sanitization for ndJSON.*

Description

This is the default ndJSON sanitizer function for log data being read into the R session by `read_logs()`. This type of function is needed because since `loggit` reimplements its own string-based JSON parser, and not a fancy one built from an AST or something, it's very easy to have bad patterns break your logs. You may also specify your own sanitizer function to pass to `loggit()`, which takes a single string and returns an (optionally-transformed) string, where each string is an individual element of the log data.

Usage

```
default_ndjson_sanitizer(string, sanitize = TRUE)

default_ndjson_unsanitizer(string)
```

Arguments

`string`	Each element of the log data to operate on. Note that this is *each element*, not each line in the logs. For example, each entry in the `log_msg` field across all logs will be sanitized/unsanitized individually. This is important because if writing your own sanitizer function, it must **take and return a single string** as its argument.
`sanitize`	Whether the operation will sanitize, or unsanitize the log data. Defaults to `TRUE`, for sanitization on write.

Details

The default string patterns and their replacements are currently mapped as follows:

Character	Replacement in log file
{	__LEFTBRACE__
}	__RIGHTBRACE__
"	__DBLQUOTE__
,	__COMMA__
\r	__CR__
\n	__LF__

Value

A single string.

set_logfile *Set Log File*

Description

Set the log file that loggit will write to. No logs outside of a temporary directory will be written until this is set explicitly, as per CRAN policy. Therefore, the default behavior is to create a file named `loggit.log` in your system's temporary directory.

Usage

```
set_logfile(logfile = NULL, confirm = TRUE)
```

Arguments

`logfile`	Full or relative path to log file. If not provided, will write to `<tmpdir>/loggit.log`.
`confirm`	Print confirmation of log file setting? Defaults to `TRUE`.

Details

A suggested use of this function would be to call it early, to log to the current working directory, as follows: `set_logfile(paste0(getwd(), "/loggit.log"))`. If you are using `loggit` in your package, you can wrap this function in `.onLoad()` so that the logfile is set when your package loads.

Examples

```
set_logfile(file.path(tempdir(), "loggit.log"))
```

set_timestamp_format *Set Timestamp Format*

Description

Set timestamp format for use in output logs. This function performs no time format validations, but will echo out the current time in the provided format for manual validation.

Usage

```
set_timestamp_format(ts_format = "%Y-%m-%dT%H:%M:%S%z", confirm = TRUE)
```

Arguments

`ts_format`	ISO date format. Defaults to ISO-8601 (e.g. "2020-01-01T00:00:00+0000").
`confirm`	Print confirmation message of timestamp format? Defaults to `TRUE`.

Details

This function provides no means of setting a timezone, and instead relies on the host system's time configuration to provide this. This is to enforce consistency across software running on the host.

Examples

```
set_timestamp_format("%Y-%m-%d %H:%M:%S")
```

`write_ndjson` *Write ndJSON-formatted log file*

Description

Write ndJSON-formatted log file

Usage

```
write_ndjson(log_df, logfile, echo = TRUE, overwrite = FALSE)
```

Arguments

`log_df`	Data frame of log data. Rows are converted to `ndjson` entries, with the columns as the fields.
`logfile`	Log file to write to. Defaults to currently-configured log file.
`echo`	Echo the `ndjson` entry to the R console? Defaults to `TRUE`.
`overwrite`	Overwrite previous log file data? Defaults to `FALSE`, and so will append new log entries to the log file.

Index