**Package ‘logr’**

March 8, 2022

**Title** Creates Log Files

**Version** 1.2.9

**Description** Contains functions to help create log files. The

package aims to overcome the difficulty of the base R sink() command. The log\_print() function will print to both the console and the file log,

without interfering in other write operations.

**License** CC0

**Encoding** UTF-8

**URL** [https://logr.r-sassy.org](https://logr.r-sassy.org/)

**BugReports** <https://github.com/dbosak01/logr/issues>

**Depends** R (>= 3.4.0)

**Suggests** knitr, rmarkdown, testthat, tidylog, dplyr, covr

**Imports** withr, utils, this.path **VignetteBuilder** knitr **RoxygenNote** 7.1.2 **NeedsCompilation** no **Author** David Bosak [aut, cre]

**Maintainer** David Bosak [<dbosak01@gmail.com>](mailto:dbosak01@gmail.com)

**Repository** CRAN

**Date/Publication** 2022-03-08 16:10:02 UTC

R **topics documented:**

logr [2](#_bookmark0)

log\_close [2](#_bookmark0)

log\_code [3](#_bookmark2)

log\_open [4](#_bookmark3)

log\_path [6](#_bookmark5)

log\_print [7](#_bookmark6)

log\_status [9](#_bookmark9)

Index [11](#_bookmark10)

1

2 *log\_close*

logr *Creates log files*

## Description

The **logr** package contains functions to easily create log files.

## Details

The **logr** package helps create log files for R scripts. The package provides easy logging, without the complexity of other logging systems. It is designed for analysts who simply want a written log of the their program execution. The package is designed as a wrapper to the base R sink() function.

## How to use

There are only three **logr** functions:

* [log\_open](#_bookmark4)
* [log\_print](#_bookmark7)
* [log\_close](#_bookmark1)

The log\_open() function initiates the log. The log\_print() function prints an object to the log. The log\_close() function closes the log. In normal situations, a user would place the call to log\_open at the top of the program, call log\_print() as needed in the program body, and call log\_close() once at the end of the program.

Logging may be controlled globally using the options "logr.on" and "logr.notes". Both options accept TRUE or FALSE values, and control log printing or log notes, respectively.

See function documentation for additional details.

log\_close *Close the log*

## Description

The log\_close function closes the log file.

## Usage

log\_close()

*log\_code* 3

## Details

The log\_close function terminates logging. As part of the termination process, the function prints any outstanding warnings to the log. Errors are printed at the point at which they occur. But warnings can be captured only at the end of the logging session. Therefore, any warning messages will only be printed at the bottom of the log.

The function also prints the log footer. The log footer contains a date-time stamp of when the log was closed.

## Value

None

## See Also

[log\_open](#_bookmark4) to open the log, and [log\_print](#_bookmark7) for printing to the log.

## Examples

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log

lf <- log\_open(tmp)

# Send message to log

log\_print("High Mileage Cars Subset")

# Perform operations

hmc <- subset(mtcars, mtcars$mpg > 20)

# Print data to log log\_print(hmc)

# Close log log\_close()

# View results writeLines(readLines(lf))

log\_code *Log the current program code*

## Description

A function to send the program/script code to the currently opened log. The log must be opened first with [log\_open](#_bookmark4). Code will be prefixed with a right arrow (">") to differentiate it from standard logging lines. The log\_code function may be called from anywhere within the program. Code will be inserted into the log at the point where it is called. The log\_code function will log the code as it is saved on disk. It will not capture any unsaved changes in the editor. If the current program file cannot be found, the function will return FALSE and no code will be written.

4 *log\_open*

## Usage

log\_code()

## Value

A TRUE or FALSE value to indicate success or failure of the function.

## See Also

[log\_open](#_bookmark4) to open the log, and [log\_close](#_bookmark1) to close the log.

## Examples

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log

lf <- log\_open(tmp)

# Write code to the log log\_code()

# Send message to log

log\_print("High Mileage Cars Subset")

# Perform operations

hmc <- subset(mtcars, mtcars$mpg > 20)

# Print data to log log\_print(hmc)

# Close log log\_close()

# View results writeLines(readLines(lf))

log\_open *Open a log*

## Description

A function to initialize the log file.

## Usage

log\_open(file\_name = "", logdir = TRUE, show\_notes = TRUE, autolog = NULL)

*log\_open* 5

## Arguments

file\_name The name of the log file. If no path is specified, the working directory will be used. As of v1.2.7, the name and path of the program or script will be used as a default if the file\_name parameter is not supplied.

logdir Send the log to a log directory named "log". If the log directory does not exist, the function will create it. Valid values are TRUE and FALSE. The default is TRUE.

show\_notes If true, will write notes to the log. Valid values are TRUE and FALSE. Default is TRUE.

autolog Whether to turn on autolog functionality. Autolog automatically logs functions from the dplyr, tidyr, and sassy family of packages. To enable autolog, either set this parameter to TRUE or set the "logr.autolog" option to TRUE. A FALSE value on this parameter will override the global option. The global option will override a NULL on this parameter. Default is that autolog is disabled.

## Details

The log\_open function initializes and opens the log file. This function must be called first, before any logging can occur. The function determines the log path, attaches event handlers, clears existing log files, and initiates a new log.

The file\_name parameter may be a full path, a relative path, or a file name. An relative path or file name will be assumed to be relative to the current working directory. If the file\_name does not have a ’.log’ extension, the log\_open function will add it.

As of v1.2.7, if the file\_name parameter is not supplied, the function will use the program/script name as the default log file name, and the program/script path as the default path.

If requested in the logdir parameter, the log\_open function will write to a ’log’ subdirectory of the path specified in the file\_name. If the ’log’ subdirectory does not exist, the function will create it.

The log file will be initialized with a header that shows the log file name, the current working directory, the current user, and a timestamp of when the log\_open function was called.

All errors, the last warning, and any log\_print output will be written to the log. The log file will exist in the location specified in the file\_name parameter, and will normally have a ’.log’ extension.

If errors or warnings are generated, a second file will be written that contains only error and warning messages. This second file will have a ’.msg’ extension and will exist in the specified log directory. If the log is clean, the msg file will not be created. The purpose of the msg file is to give the user a visual indicator from the file system that an error or warning occurred. This indicator msg file is useful when running programs in batch.

To use **logr**, call log\_open, and then make calls to log\_print as needed to print variables or data frames to the log. The log\_print function can be used in place of a standard print function. Any- thing printed with log\_print will be printed to the log, and to the console if working interactively.

This package provides the functionality of sink, but in much more user-friendly way. Recom- mended usage is to call log\_open at the top of the script, call log\_print as needed to log interim state, and call log\_close at the bottom of the script.

6 *log\_path*

Logging may be controlled globally using the "logr.on" option. This option accepts a TRUE or FALSE value. If the option is set to FALSE, **logr** will print to the console, but not to the log. Example: options("logr.on" = TRUE)

Notes may be controlled globally using the "logr.notes" option. This option also accepts a TRUE or FALSE value, and determines whether or not to print notes in the log. The global option will override the show\_notes parameter on the log\_open function. Example: options("logr.notes"

= FALSE)

Version v1.2.0 of the **logr** package introduced **autolog**. The autolog feature provides automatic log- ging for **dplyr**, **tidyr**, and the **sassy** family of packages. To use autolog, set the autolog parameter to TRUE, or set the global option logr.autolog to TRUE. To maintain backward compatibility with prior versions, autolog is disabled by default.

## Value

The path of the log.

## See Also

[log\_print](#_bookmark7) for printing to the log (and console), and [log\_close](#_bookmark1) to close the log.

## Examples

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log

lf <- log\_open(tmp)

# Send message to log

log\_print("High Mileage Cars Subset")

# Perform operations

hmc <- subset(mtcars, mtcars$mpg > 20)

# Print data to log log\_print(hmc)

# Close log log\_close()

# View results writeLines(readLines(lf))

log\_path *Get the path of the current log*

*log\_print* 7

## Description

The log\_path function gets the path to the currently opened log. This function may be useful when you want to manipulate the log in some way, and need the path. The function takes no parameters.

## Usage

log\_path()

## Value

The full path to the currently opened log, or NULL if no log is open.

## Examples

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log log\_open(tmp)

# Get path

lf <- log\_path()

# Close log log\_close()

lf

log\_print *Print an object to the log*

## Description

The log\_print function prints an object to the currently opened log.

## Usage

log\_print(x, ..., console = TRUE, blank\_after = TRUE, msg = FALSE, hide\_notes = FALSE) put(x, ..., console = TRUE, blank\_after = TRUE, msg = FALSE, hide\_notes = FALSE) sep(x, console = TRUE)

log\_hook(x)

8 *log\_print*

## Arguments

x The object to print.

... Any parameters to pass to the print function.

console Whether or not to print to the console. Valid values are TRUE and FALSE. Default is TRUE.

blank\_after Whether or not to print a blank line following the printed object. The blank line helps readability of the log. Valid values are TRUE and FALSE. Default is TRUE.

msg Whether to print the object to the msg log. This parameter is intended to be used internally. Value values are TRUE and FALSE. The default value is FALSE.

hide\_notes If notes are on, this parameter gives you the option of not printing notes for a particular log entry. Default is FALSE, meaning notes will be displayed. Used internally.

## Details

The log is initialized with log\_open. Once the log is open, objects like variables and data frames can be printed to the log to monitor execution of your script. If working interactively, the function will print both to the log and to the console. The log\_print function is useful when writing and debugging batch scripts, and in situations where some record of a scripts’ execution is required.

If requested in the log\_open function, log\_print will print a note after each call. The note will contain a date-time stamp and elapsed time since the last call to log\_print. When printing a data frame, the log\_print function will also print the number and rows and column in the data frame. These counts may also be useful in debugging.

Notes may be turned off either by setting the show\_notes parameter on log\_open to FALSE, or by setting the global option "logr.notes" to FALSE.

The put function is a shorthand alias for log\_print. You can use put anywhere you would use

log\_print. The functionality is identical.

The sep function is also a shorthand alias for log\_print, except it will print a separator before and after the printed text. This function is intended for documentation purposes, and you can use it to help organize your log into sections.

The log\_hook function is for other packages that wish to integrate with **logr**. The function prints to the log only if autolog is enabled. It will not print to the console.

## Value

The object, invisibly

## See Also

[log\_open](#_bookmark4) to open the log, and [log\_close](#_bookmark1) to close the log.

*log\_status* 9

## Examples

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log

lf <- log\_open(tmp)

# Send message to log

log\_print("High Mileage Cars Subset")

# Perform operations

hmc <- subset(mtcars, mtcars$mpg > 20)

# Print data to log log\_print(hmc)

# Close log log\_close()

# View results writeLines(readLines(lf))

log\_status *Get the status of the log*

## Description

The log\_status function gets the status of the log. Possible status values are ’on’, ’off’, ’open’, or ’closed’. The function takes no parameters.

## Usage

log\_status()

## Value

The status of the log as a character string.

## Examples

# Check status before the log is opened log\_status()

# [1] "closed"

# Create temp file location

tmp <- file.path(tempdir(), "test.log")

# Open log

lf <- log\_open(tmp)

10 *log\_status*

# Check status after log is opened log\_status()

# [1] "open"

# Close log log\_close()

# Index

log\_close, [*2*](#_bookmark0), [2](#_bookmark0), [*4*](#_bookmark3), [*6*](#_bookmark5), [*8*](#_bookmark8)

log\_code, [3](#_bookmark2)

log\_hook *(*log\_print*)*, [7](#_bookmark6)

log\_open, [*2*](#_bookmark0)*–*[*4*](#_bookmark3), [4](#_bookmark3), [*8*](#_bookmark8)

log\_path, [6](#_bookmark5)

log\_print, [*2*](#_bookmark0)*,* [*3*](#_bookmark2), [*6*](#_bookmark5), [7](#_bookmark6)

log\_status, [9](#_bookmark9)

logr, [2](#_bookmark0)

put *(*log\_print*)*, [7](#_bookmark6)

sep *(*log\_print*)*, [7](#_bookmark6)

11