Package 'logr'

October 4, 2021

Title Creates Log Files

Version 1.2.6

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

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 7.1.1

NeedsCompilation no

Author David Bosak [aut, cre]

Maintainer David Bosak <dbosak01@gmail.com>

Repository CRAN

Date/Publication 2021-10-04 12:50:02 UTC

R topics documented:

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`
- `log_print`
- `log_close`

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()
```

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 to open the log, and log_print 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. 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.

Usage

```
log_code()
```

Value

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

See Also

log_open to open the log, and log_close 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)
```

Arguments

file_name	The name of the log file. If no path is specified, the working directory will be used.
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.

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. Anything 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. Recommended 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.

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 logging 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 for printing to the log (and console), and log_close 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*

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)
```

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` to open the log, and `log_close` 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_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)

# Check status after log is opened
log_status()
# [1] "open"

# Close log
log_close()
```

Index