Package 'lumberjack'

May 9, 2020

Maintainer Mark van der Loo <mark.vanderloo@gmail.com>

License EUPL

Title Track Changes in Data

LazyData no

Type Package

LazyLoad yes

Description A framework that allows for easy logging of changes in data. Main features: start tracking changes by adding a single line of code to an existing script. Track changes in multiple datasets, using multiple loggers. Add custom-built loggers or use loggers offered by other packages.

Version 1.2.0

URL <https://github.com/markvanderloo/lumberjack>

BugReports <https://github.com/markvanderloo/lumberjack/issues>

Imports utils, R6

Depends R (>= 3.4.0)

Suggests tinytest

RoxygenNote 7.1.0

Encoding UTF-8

NeedsCompilation no

Author Mark van der Loo [aut, cre] (<https://orcid.org/0000-0002-9807-4686>)

Repository CRAN

Date/Publication 2020-05-08 22:40:03 UTC

R topics documented:

1

---

cellwise   *The cellwise logger.*

Description

The cellwise logger registers the row, column, old, and new value of cells that changed, along with
a step number, timestamp, source reference, and the expression used to alter a dataset.

Usage

```
cellwise(key, verbose=TRUE, tempfile=file.path(tempdir(),"cellwise.csv"))
```

Arguments

key   [character|integer] index to column that uniquely identifies a row.

verbose   [logical] toggle verbosity.

tempfile   [character] filename for temporary log storage.

Format

An R6 class object.

Creating a logger

```
cellwise$new(key,verbose=TRUE,file=tempfile())
```

key   [character|integer] index to column that uniquely identifies a row.
verbose   [logical] toggle verbosity.
tempfile   [character] filename for temporary log storage.

Dump options

```
$dump(file=NULL)
```

file   [character] location to write final output to.

The default location is "cellwise.csv" in an interactive session, and "DATA_cellwise.csv" in a script that executed via run_file. Here, DATA is the variable name of the data being tracked or the label provided with start_log.

Getting data from the logger

$logdata() Returns a data frame with the current log.

Details

At initialization, the cellwise logger opens a connection to a temporary file. All logging info is appended to that connection. When dump_log is called, the temporary file is closed, copied to the output file, and reopened for writing. The connection is closed automatically when the logger is destroyed, for example when calling stop_log().

See Also

Other loggers: expression_logger, filedump, simple

Examples

```
logfile <- tempfile(fileext=".csv")

# convert height from inch to cm and log changes.
# we need to set a unique key.
women$sleutel <- 1:nrow(women)
out <- women %L>%
  start_log(log=cellwise$new(key="sleutel")) %L>%
  {.$height <- .$height*2.54; .} %L>%
  dump_log(file=logfile, stop=TRUE)

read.csv(logfile) %L>% head()

# work with an externally defined logger.
iris$id <- seq_len(nrow(iris))
logger <- cellwise$new(key="id")
iris %L>%
  start_log(logger) %L>%
  head() %L>%
  stop_log(dump=FALSE)
logger$logdata()
```

---

dump_log   *Dump logging data*

Description

Calls the $dump(...) method of logger(s) tracking an R object.

Usage

```
dump_log(data, logger = NULL, stop = TRUE, ...)
```

Arguments

data	An R object tracked by one or more loggers.
logger	[character] vector. Class names of loggers to dump (e.g. "simple"). When loggers=NULL, all loggers are dumped for this object.
stop	[logical] stop logging after the dump? Removes the logger(s) tracking the object.
...	Arguments passed to the dump method of the logger.

Value

data, invisibly.

See Also

Other control: %>>%(), get_log(), run_file(), start_log(), stop_log()

Examples

```
logfile <- tempfile(fileext=".csv")
women %L>%
  start_log(logger=simple$new()) %L>%
  transform(height_cm = height*2.52) %L>%
  dump_log(file=logfile)
logdata <- read.csv(logfile)
head(logdata)
```

---

expression_logger   *The expression logger.*

Description

Records the result of one or more user-defined expressions that perform calculations on the object being tracked.

Format

An R6 class object.

Creating a logger

```
expression_logger$new(...,verbose=TRUE)
```

            `...`   A comma-separated list of `name = expression` pairs.

    `verbose`   `[logical]` toggle verbosity.

Each expression will be evaluated in the context of the object tracked with this logger. An expression is expected to have a single `numeric` or `character` output.

Dump options

```
$dump(file=NULL)
```

        `file`   `[character]` location to write final output to.

The default location is `"expression.csv"` in an interactive session, and `"DATA_expression.csv"` in a script that executed via [run_file](#). Here, `DATA` is the variable name of the data being tracked or the `label` provided with [start_log](#).

See Also

Other loggers: [cellwise](#), [filedump](#), [simple](#)

Examples

```
logfile <- file.path(tempfile(fileext=".csv"))
e <- expression_logger$new(mean=mean(height), sd=sd(height))

out <- women %L>%
  start_log(e) %L>%
  within(height <- height * 2) %L>%
  within(height <- height * 3) %L>%
  dump_log(file=logfile)

read.csv(logfile)
```

---

filedump   *The file dumping logger.*

Description

The file dumping logger dumps the most recent version of a dataset to csv in a directory of choice.

Format

An R6 class object.

Creating a logger

```
filedump$new(dir=file.path(tempdir(),"filedump"),filename="%sstep%03d.csv",verbose=TRUE)
```

    dir   [character] Directory location to write the file dumps.
    filename   [character] Template, used to create file names. to create a file name.
    verbose   [logical] toggle verbosity.

File locations are created with `file.path(dir,file)`, where `file` is generated as `sprintf(filename,DATA,STEP)`. In interactive sessions `DATA=""`. In sessions where a script is executed using `run_file`, DATA is the name of the R object being tracked or the `label` provided with `start_log`. STEP is a counter that increases at each dump.

Dump options

```
$dump(...)
```

                        ...   Currently unused.

Retrieve log data

`$logdata()` returns a list of data frames, sorted in the order returned by `base::dir()`

Details

If `dir` does not exist it is created.

See Also

Other loggers: `cellwise`, `expression_logger`, `simple`

Examples

```
logger <- filedump$new()

out <- women %L>%
  start_log(logger) %L>%
  within(height <- height * 2) %L>%
  within(height <- height * 3) %L>%
  dump_log()
dir(file.path(tempdir(),"filedump"))
```

---

get_log   *Get log object from a data item*

Description

Get log object from a data item

Usage

```
get_log(data, logger = NULL)
```

Arguments

data   An R object.

logger   [character] scalar. Logger to return. Can be NULL when a single logger is
                attached.

Value

A logging object, or NULL if none exists.

See Also

Other control: %>>%(), dump_log(), run_file(), start_log(), stop_log()

---

lumberjack   *Track changes in data*

Description

This package allows you to track changes in R objects by defining one or more loggers for each
object. There are a number of built-in loggers and users (or package authors) can create their own
loggers. To get started please have a look at the using lumberjack vignette.

Author(s)

Mark van der Loo

---

run_file   *Run a file while tracking changes in data*

Description

Run all code in a file. Changes in data that are tracked, (e.g. with [start_log](data)) will be followed by the assigned loggers.

Usage

```
run_file(file, auto_dump = TRUE, envir = NULL)

source_file(file, auto_dump = TRUE)
```

Arguments

file	[character] file to run.
auto_dump	[logical] Toggle automatically dump all remaining logs after executing file.
envir	[environment] to run the code in. By default a new environment will be created with .GlobalEnv as parent.

Value

The environment where the code was executed, invisibly.

Details

run_file runs code in a separate environment, and returns the environment with all the variables created by the code. source_file acts like [source](#) and runs all the code in the current global workspace (.GlobalEnv).

See Also

Other control: [%>>%](), [dump_log](), [get_log](), [start_log](), [stop_log]()

Examples

```
# using 'dontrun'
## Not run:
# create an R file, with logging.
script <- "
library(lumberjack)
data(women)
start_log(women, logger=simple$new())
women$height <- women$height*2.54
women$weight <- women$weight*0.453592
dump_log()
"
```

```
write(script, file="myscript.R")
# run the script
lumberjack::run_file("myscript.R")
# read the logfile
read.csv("women_simple.csv")

## End(Not run)
```

---

simple   *The simple logger*

Description

Record for each expression a POSIXct timestamp and a logical indicating whether the tracked object has changed.

Format

An R6 class object.

Creating a logger

simple$new(verbose=TRUE)

verbose   toggle verbosity

Dump options

$dump(file=NULL,...)

file   filename or connection to write output to.
...   extra options passed to write.csv, except row.names, which is set to FALSE.

The default location is "simple.csv" in an interactive session, and "DATA_simple.csv" in a script that executed via run_file. Here, DATA is the variable name of the data being tracked or the label provided with start_log.

Get data

$logdata() Returns a data frame with the current log.

See Also

Other loggers: cellwise, expression_logger, filedump

Examples

```
logfile <- tempfile(fileext=".csv")
out <- women %L>%
  start_log(log=simple$new(verbose=FALSE)) %L>%
  identity() %L>%
  head() %L>%
  dump_log(file=logfile, stop=TRUE)


read.csv(logfile,stringsAsFactors=FALSE)
```

---

start_log   *Start tracking an R object*

Description

Start tracking an R object

Usage

```
start_log(data, logger = simple$new(), label = NULL)
```

Arguments

data	An R object.
logger	A logging object (typically an environment wrapped in an S3 class)
label	[character] scalar. A label to attach to the logger (for loggers supporting it).

Details

All loggers that come with **lumberjack** support labeling. The label is used by dump methods to create a unique file name for each object/logger combination.

If label is not supplied, start_log attempts to create a label from the name of the data variable. This probably fails when data is not a variable but an expression (like read.csv...). A label is also not created when data is passed via the lumberjack not-a-pipe operator. In that case the label is (silently) not set. In cases where multiple datasets are logged with the same type of logger, this could lead to overwriting of dump files, unless file is explicitly defined when calling dump_log.

See Also

Other control: %>>%(), dump_log(), get_log(), run_file(), stop_log()

Examples

```
logfile <- tempfile(fileext=".csv")
women %L>%
  start_log(logger=simple$new()) %L>%
  transform(height_cm = height*2.52) %L>%
  dump_log(file=logfile)
logdata <- read.csv(logfile)
head(logdata)
```

---

stop_log   *Stop logging*

Description

Calls the logger's `$stop()` method if it exists, and removes the logger as attribute from `data`.

Usage

```
stop_log(data, logger = NULL, dump = TRUE, ...)
```

Arguments

data	An R object.
logger	[character] vector. Class names of loggers to dump (e.g. "simple"). When loggers=NULL, all loggers are stopped and removed for this data.
dump	['logical'] Toggle dump log file.
...	Passed to the logger's dump method, if it exists.

Value

The data, invisibly.

See Also

Other control: %>>%(), dump_log(), get_log(), run_file(), start_log()

Examples

```
logfile <- tempfile(fileext=".csv")
women %L>%
  start_log(logger=simple$new()) %L>%
  transform(height_cm = height*2.52) %L>%
  dump_log(file=logfile)
logdata <- read.csv(logfile)
head(logdata)
```

%>>%   *The lumberjack operator*

Description

The not-a-pipe operator that tracks changes in data.

Usage

```
lhs %>>% rhs

lhs %L>% rhs
```

Arguments

lhs	Input value
rhs	Function call or 'dotted' expression (see below). as value

Piping

The operators %L>% and %>>% are synonyms. The %L>% is the default since version 0.3.0 to avoid confusion with the %>>% operator of the pipeR package but %>>% still works.

The lumberjack operator behaves as a simplified version of the magrittr pipe operator. The basic behavior of lhs %>>% rhs is the following:

- If the rhs uses dot-variables (.), these are interpreted as the left-hand side, except in formulas where dots already have a special meaning.
- If the rhs is a function call, with no dot-variables used, the lhs is used as its first argument.

The most notable differences with 'magrittr' are the following.

- it does not allow you to define functions in the magrittr style, like a <-. %>% sin(.)
- there is no assignment-pipe like %<>%.
- you cannot do things like x %>% sin (without the brackets).

Logging

If the left-hand-side is tagged for logging, the lumberjack will update the log by calling the logger's $add() method, with arguments meta, input, output. Here, meta is a list with information on the operations performed, and input and output are the left-hand-side and the result, respectively.

See Also

Other control: dump_log(), get_log(), run_file(), start_log(), stop_log()

Examples

```
# pass arguments to a function
1:3 %L>% mean()

# pass arguments using "."
TRUE %L>% mean(c(1,NA,3), na.rm = .)

# pass arguments to an expression, using "."
1:3 %L>% { 3 * .}

# in a more complicated expression, return "." explicitly
women %L>% { .$height <- 2*.$height; . }
```

Index