Package 'chronicler'

May 8, 2026

Title Add Logging to Functions

Version 0.3.0

Description Decorate functions to make them return enhanced output. The enhanced output consists in an object of type 'chronicle' containing the result of the function applied to its arguments, as well as a log detailing when the function was run, what were its inputs, what were the errors (if the function failed to run) and other useful information. Tools to handle decorated functions are included, such as a forward pipe operator that makes chaining decorated functions possible.

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports diffobj, dplyr, ggplot2, maybe, rlang, tibble, utils

Suggests clipr, knitr, lubridate, openxlsx, purrr, rmarkdown, testthat, tidyr

VignetteBuilder knitr

Encoding UTF-8

LazyData TRUE

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author Bruno Rodrigues [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-3211-3689>), Matouš Eibich [ctb]

Maintainer Bruno Rodrigues <bruno@brodrigues.co>

Repository CRAN

Date/Publication 2025-08-18 09:20:33 UTC

Contents

as.data.frame.chronicle . . . 2
as_chronicle . . . 3
avia . . . 4
bind_record . . . 4
check_diff . . . 5
check_g . . . 6
flatten_record . . . 6
fmap_record . . . 7
is_chronicle . . . 8
print.chronicle . . . 8
purely . . . 9
read_log . . . 10
record . . . 11
record_ggplot . . . 12
record_many . . . 12
unveil . . . 13
write_chronicle_df . . . 14
zap_log . . . 15
%>=% . . . 16

Index **17**

`as.data.frame.chronicle` *Coerce a chronicle object to a data.frame*

Description

This is an S3 method that allows for the easy coercion of a `chronicle` object into a standard `data.frame`. It automatically unwraps the object by calling `unveil(.c, "value")` and then attempts to convert the result into a data frame.

Usage

```
## S3 method for class 'chronicle'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	A `chronicle` object.
`row.names`	`NULL` or a character vector giving the row names for the data frame.
`optional`	logical. If `TRUE`, setting row names and converting column names is optional.
`...`	Additional arguments to be passed to or from other methods.

Details

The function will produce an error if the underlying `value` of the chronicle object cannot be coerced to a data frame by the base `as.data.frame()` method. The error message will be specific, indicating the class of the object that caused the failure.

Value

A `data.frame` if the `value` inside the chronicle object is a `data.frame` or can be successfully coerced into one.

Examples

```
library(dplyr)

# --- Successful Example ---

# Create a chronicle object whose value is a data frame
starwars_chronicle <- starwars %>%
  record(filter)(species == "Human") %>%
  bind_record(record(select), name, height, mass)

# Now, you can use as.data.frame() directly on the chronicle object
sw_df <- as.data.frame(starwars_chronicle)

class(sw_df)
head(sw_df)


# --- Error Example ---

# Create a chronicle object whose value is a number
numeric_chronicle <- record(sqrt)(100)

# This will fail with a specific error message because a number
# cannot be turned into a data frame.
try(as.data.frame(numeric_chronicle))
```

---

as_chronicle *Coerce an object to a chronicle object.*

Description

Coerce an object to a chronicle object.

Usage

```
as_chronicle(.x, .log_df = data.frame())
```

Arguments

.x	Any object.
.log_df	Used internally, the user does need to interact with it. Defaults to an empty data frame.

Value

Returns a chronicle object with the object as the $value.

Examples

```
as_chronicle(3)
```

---

avia — *avia Air passenger transport between the main airports of Luxembourg and their main partner airports*

Description

A non-tidy dataset from EUROSTAT which can be found here.

Usage

```
avia
```

Format

A data frame with 1,434 rows and 332 columns.

---

bind_record — *Evaluate a decorated function; used to chain multiple decorated functions.*

Description

Evaluate a decorated function; used to chain multiple decorated functions.

Usage

```
bind_record(.c, .f, ...)
```

Arguments

.c	A chronicle object.
.f	A chronicle function to apply to the returning value of .c.
...	Further parameters to pass to .f.

Value

A chronicle object.

Examples

```
r_sqrt <- record(sqrt)
r_exp <- record(exp)
3 |> r_sqrt() |> bind_record(r_exp)
```

`check_diff` *Check the output of the diff column*

Description

Check the output of the diff column

Usage

```
check_diff(.c, columns = c("ops_number", "function"))
```

Arguments

`.c`	A chronicle object.
`columns`	Columns to select for the output. Defaults to c("ops_number", "function").

Details

`diff` is an option argument to the `record()` function. When `diff` = "full", a diff of the input and output of the decorated function gets saved, and if `diff` = "summary" only a summary of the diff is saved.

Value

A data.frame with the selected columns and column "diff_obj".

Examples

```
r_subset <- record(subset, diff = "full")
result <- r_subset(mtcars, select = am)
check_diff(result) # <- this is the data frame listing the operations and the accompanying diffs
check_diff(result)$diff_obj # <- actually look at the diffs
```

check_g *Check the output of the .g function*

Description

Check the output of the .g function

Usage

```
check_g(.c, columns = c("ops_number", "function"))
```

Arguments

.c	A chronicle object.
columns	Columns to select for the output. Defaults to c("ops_number", "function").

Details

.g is an option argument to the record() function. Providing this optional function allows you, at each step of a pipeline, to monitor interesting characteristics of the value object. See the package's Readme file for an example with data frames.

Value

A data.frame with the selected columns and column "g".

Examples

```
r_subset <- record(subset, .g = dim)
result <- r_subset(mtcars, select = am)
check_g(result)
```

flatten_record *Flatten nested chronicle objects*

Description

Flatten nested chronicle objects

Usage

```
flatten_record(.c)
```

Arguments

.c	A nested chronicle object, where the $value element is itself a chronicle object

Value

Returns `.c` where value is the actual value, and logs are concatenated.

Examples

```
r_sqrt <- record(sqrt)
r_log <- record(log)
a <- as_chronicle(r_log(10))
a
flatten_record(a)
```

`fmap_record` *Evaluate a non-chronicle function on a chronicle object.*

Description

Evaluate a non-chronicle function on a chronicle object.

Usage

```
fmap_record(.c, .f, ...)
```

Arguments

`.c`	A chronicle object.
`.f`	A non-chronicle function.
`...`	Further parameters to pass to `.f`.

Value

Returns the result of `.f(.c$value)` as a new chronicle object.

Examples

```
as_chronicle(3) |> fmap_record(sqrt)
```

is_chronicle *Checks whether an object is of class "chronicle"*

Description

Checks whether an object is of class "chronicle"

Usage

```
is_chronicle(.x)
```

Arguments

.x	An object to test.

Value

TRUE if .x is of class "chronicle", FALSE if not.

print.chronicle *Print method for chronicle objects.*

Description

Print method for chronicle objects.

Usage

```
## S3 method for class 'chronicle'
print(x, ...)
```

Arguments

x	A chronicle object.
...	Unused.

Details

`chronicle` object are, at their core, lists with the following elements:

- "$value": a an object of type `maybe` containing the result of the computation (see the "Maybe monad" vignette for more details on `maybe`s).
- "$log_df": a `data.frame` object containing the printed object's log information.

`print.chronicle()` prints the object on screen and shows:

- the value using its `print()` method (for example, if the value is a data.frame, `print.data.frame()` will be used)
- a message indicating to the user how to recuperate the value inside the `chronicle` object and how to read the object's log

Value

No return value, called for side effects (printing the object on screen).

`purely` *Capture all errors, warnings and messages.*

Description

Capture all errors, warnings and messages.

Usage

```
purely(.f, strict = 2)
```

Arguments

`.f`	A function to decorate.
`strict`	Controls if the decorated function should catch only errors (1), errors and warnings (2, the default) or errors, warnings and messages (3).

Value

A function which returns a list. The first element of the list, `$value`, is the result of the original function `.f` applied to its inputs. The second element, `$log` is `NULL` in case everything goes well. In case of error/warning/message, `$value` is NA and `$log` holds the message. `purely()` is used by `record()` to allow the latter to handle errors.

Examples

```
purely(log)(10)
purely(log)(-10)
purely(log, strict = 1)(-10) # This produces a warning, so with strict = 1 nothing gets captured.
```

read_log *Read and display the log of a chronicle*

Description

`read_log()` provides different human-readable views of the log information stored in a `chronicle` object. It can show a pretty, narrative-style summary, a tabular summary suitable for inspection or debugging, or a compact error-focused report.

Usage

```
read_log(.c, style = c("pretty", "table", "errors-only"))
```

Arguments

`.c`	A chronicle object.
`style`	A string indicating the display style. One of: • `"pretty"`: a short, human-friendly log with OK/NOK status, function names, timestamps, and runtimes. • `"table"`: a tabular summary of the log as a data frame, including function names, status, runtime, and messages. • `"errors-only"`: a minimal report. If all steps succeed, only a single success message is shown. If any step fails, only the failures are listed.

Value

- If `style = "pretty"`: a character vector of sentences.
- If `style = "table"`: a data frame summarising the log (with an attribute `"total_runtime_secs"` storing the total runtime in seconds).
- If `style = "errors-only"`: a character string if all succeeded, or a character vector listing only the failed steps.

Examples

```
r_select <- record(dplyr::select)
r_group_by <- record(dplyr::group_by)
r_summarise <- record(dplyr::summarise)

output <- dplyr::starwars |>
  r_select(height, mass, species, sex) |>
  bind_record(r_group_by, species, sex) |>
  bind_record(r_summarise, mass = mean(mass, na.rm = TRUE))

read_log(output, style = "pretty")
read_log(output, style = "table")
read_log(output, style = "errors-only")
```

`record` *Decorates a function to output objects of type* `chronicle`.

Description

Decorates a function to output objects of type `chronicle`.

Usage

```
record(.f, .g = (function(x) NA), strict = 2, diff = "none")
```

Arguments

`.f`	A function to decorate.
`.g`	Optional. A function to apply to the intermediary results for monitoring purposes. Defaults to returning NA.
`strict`	Controls if the decorated function should catch only errors (1), errors and warnings (2, the default) or errors, warnings and messages (3).
`diff`	Whether to show the diff between the input and the output ("full"), just a summary of the diff ("summary"), or none ("none", the default)

Details

To chain multiple decorated function, use `bind_record()` or `%>=%`. If the `diff` parameter is set to "full", `diffobj::diffObj()` (or `diffobj::summary(diffobj::diffObj()`, if diff is set to "summary") gets used to provide the diff between the input and the output. This diff can be found in the `log_df` element of the result, and can be viewed using `check_diff()`.

Value

A function which returns objects of type `chronicle`. `chronicle` objects carry several elements: a `value` which is the result of the function evaluated on its inputs and a second object called `log_df`. `log_df` contains logging information, which can be read using `read_log()`. `log_df` is a data frame with columns: outcome, function, arguments, message, start_time, end_time, run_time, g and diff_obj.

Examples

```
record(sqrt)(10)
record(sqrt)(x = 10)
```

record_ggplot *Record a ggplot expression*

Description

record_ggplot captures a complete {ggplot2} expression, evaluates it, and creates a chronicle object. It uses a robust tryCatch and withCallingHandlers pattern to reliably capture errors, warnings, and messages.

To trigger all conditions, including rendering-time warnings and messages, it forces a full render of the plot. This is achieved safely by opening a null graphics device (pdf(NULL)), scheduling its closure with on.exit(dev.off()), and then printing the plot. This guarantees that the temporary device is always closed, even if an error occurs, preventing any side effects on the user's active graphics session.

Usage

```
record_ggplot(ggplot_expression, strict = 2)
```

Arguments

ggplot_expression	The entire {ggplot2} expression to be recorded.
strict	An optional integer argument controlling what is treated as a failure: • 1: Catches only errors. • 2: Catches errors and warnings (the default). • 3: Catches errors, warnings, and messages.

Value

A chronicle object. When printed, it will display the plot if successful or an error plot if it failed.

record_many *Decorate a list of functions*

Description

Decorate a list of functions

Usage

```
record_many(list_funcs, .g = (function(x) NA), strict = 2, diff = "none")
```

Arguments

list_funcs	A list of function names, as strings.
.g	Optional. Defaults to a function which returns NA.
strict	Controls if the decorated function should catch only errors (1), errors and warnings (2, the default) or errors, warnings and messages (3).
diff	Whether to show the diff between the input and the output ("full"), just a summary of the diff ("summary"), or none ("none", the default)

Details

Functions must be entered as strings of the form "function" or "package::function". The code gets generated and copied into the clipboard. The code can then be pasted into the text editor. On GNU/Linux systems, you might get the following error message on first use: "Error in : Clipboard on X11 requires that the DISPLAY envvar be configured". This is an error message from `clipr::write_clip()`, used by `record_many()` to put the generated code into the system's clipboard. To solve this issue, run `echo $DISPLAY` in the system's shell. This command should return a string like ":0". Take note of this string. In your .Rprofile, put the following command: Sys.setenv(DISPLAY = ":0") and restart the R session. `record_many()` should now work.

Value

Puts a string into the systems clipboard.

Examples

```
## Not run:
list_funcs <- list("exp", "dplyr::select", "exp")
record_many(list_funcs)

## End(Not run)
```

---

unveil *Retrieve an element from a chronicle object.*

Description

Retrieve an element from a chronicle object.

Usage

```
unveil(.c, .e = "value")
```

Arguments

.c	A chronicle object.
.e	Element of interest to retrieve, one of "value" (default) or "log_df".

Value

The `value` or `log_df` element of the chronicle object .c.

Examples

```
r_sqrt <- record(sqrt)
r_exp <- record(exp)
3 |> r_sqrt() %>=% r_exp() |> unveil("value")
```

`write_chronicle_df` *Write a chronicler Data Frame object to a file*

Description

Saves the contents of a `chronicler` object to a CSV or Excel file, including both the dataset and the log of operations. The data is stored in the `value` component of the chronicler object, and the log is included as metadata in the output.

Usage

```
write_chronicle_df(.c, path, row.names = FALSE, sep = ",", ...)
```

Arguments

`.c`	A `chronicler` object.
`path`	A single character string specifying the output file path. The file extension must be either `.csv` or `.xlsx`.
`row.names`	Logical, whether to include row names when writing to CSV. Defaults to `FALSE`.
`sep`	Character. Field separator for CSV output. Defaults to `","`.
`...`	Additional arguments passed to `write.table` when writing CSV.

Details

When writing a CSV file, the first few lines contain the log of operations performed on the data. Users should skip these lines when reading the data back in. When writing an Excel file, two sheets are created: `value` containing the dataset, and `log` containing the log of operations as a data frame for better readability.

Value

Invisibly returns `NULL`. The function is called for its side effect of writing files.

Examples

```
## Not run:
# Assume `c` is a chronicler object
write_chronicle_df(c, path = "output.csv")
write_chronicle_df(c, path = "output.xlsx")

## End(Not run)
```

---

zap_log *Zap the log of a chronicle object*

Description

This function replaces the entire existing log of a `chronicle` object with a single, new entry. This new entry simply records that the log was "zapped" and the time at which it occurred.

This is useful for simplifying a `chronicle` object before saving or sharing, or to mark a definitive checkpoint in a workflow, effectively discarding the previous history. The underlying `value` of the object remains completely unchanged.

Usage

```
zap_log(.c)
```

Arguments

`.c`	A `chronicle` object.

Value

A new `chronicle` object with the same `value` as the input, but with its `log_df` replaced by a single entry.

Examples

```
library(dplyr)

# 1. Create a chronicle object with a multi-step log
r_select <- record(select)
r_filter <- record(filter)

original_chronicle <- starwars %>%
  r_select(name, height, mass, species) %>%
  bind_record(r_filter, species == "Human")

# 2. View the original, detailed log
cat("--- Original Log ---\n")
read_log(original_chronicle)
```

```
# 3. Zap the log
zapped_chronicle <- zap_log(original_chronicle)

# 4. View the new, simplified log
cat("\n--- Zapped Log ---\n")
read_log(zapped_chronicle)

# 5. The underlying data value is unaffected
identical(
  unveil(original_chronicle, "value"),
  unveil(zapped_chronicle, "value")
)
```

`%>=%` *Pipe a chronicle object to a decorated function.*

Description

Pipe a chronicle object to a decorated function.

Usage

```
.c %>=% .f
```

Arguments

`.c`	A value returned by record.
`.f`	A chronicle function to apply to the returning value of .c.

Value

A chronicle object.

Examples

```
r_sqrt <- record(sqrt)
r_exp <- record(exp)
3 |> r_sqrt() %>=% r_exp()
```

Index

∗ **datasets**
    avia, 4
%>=%, 16

as.data.frame.chronicle, 2
as_chronicle, 3
avia, 4

bind_record, 4

check_diff, 5
check_g, 6

flatten_record, 6
fmap_record, 7

is_chronicle, 8

print.chronicle, 8
purely, 9

read_log, 10
record, 11
record_ggplot, 12
record_many, 12

unveil, 13

write.table, *14*
write_chronicle_df, 14

zap_log, 15