Monitoring and Logging R Scripts in Production

https://daroczig.github.io/logger

Gergely Daroczi

Directions in Statistical Computing
September 19, 2019

About me (using R for production)

- 2006: calling R scripts from PHP (both reading from MySQL) to generate custom plots on surveys in a homepage for interactive use
- 2008: automated/batch R scripts to generate thousands of pages of crosstables, ANOVA and plots from SPSS with `pdflatex`
- 2011: web application combining Ruby on Rails, `pandoc` and RApache providing reports in plain English for survey analysis (NoSQL databases, vertical scaling, security, central error tracking etc)
- 2012: plain RApache web application for NLP and network analysis
- 2015: standardizing the data infrastructure of a fintech startup to use R both in bath jobs and stream processing (ETL, reporting, fraud detection, daily operations, customer communication etc)
- 2017: redesign, monitor and scale the DS infrastructure of an adtech startup for batch and live scoring

About me (using R for production)

Using in R in a non-interactive way:

- jobs are scheduled to run without manual intervention (eg CRON or upstream job trigger, API request)
- the output of the jobs are recorded and monitored (eg `error` handler for ErrBit, CloudWatch logs or Splunk etc)
- if an error occurs, usually there is no other way to figure out what happened then looking at the recorded job output
- need for a standard, eg containerized environment (eg R and package versions, OS packages, `.Rprofile` etc)
- security! (safeguarded production environment, SQL injection, AppArmor, encrypted credentials etc)

Motivation

Nooooooooooo

```
$ Rscript super_important_business_stuff.R
```

Motivation

Noooooooooo

```
$ Rscript super_important_business_stuff.R
```

```
Error in l[[x]] : subscript out of bounds
Calls: g -> f
Execution halted
```

Motivation

Nooooooooooo

```
$ Rscript super_important_business_stuff.R
```

```
Error in l[[x]] : subscript out of bounds
Calls: g -> f
Execution halted
```

```
$ Rscript super_important_business_stuff.R

Error in .subset2(x, i, exact = exact) : subscript out of bour
Execution halted
```

Motivation

Debugging

```r
for (i in 1:100) {
    ## do something slow
    print(i)
}
```

Motivation
Debugging

```r
for (i in 1:100) {
    ## do something slow
    print(i)
}
```

```r
N <- 42
for (i in 1:N) {
    ## do something slow
    print(paste(
        Sys.time(), '|',
        i, 'out of', N,
        '=', round(i / N * 100), '%'))
    flush.console()
}
```

Motivation

Debugging

```
[1] "2019-09-15 00:05:34 | 1 out of 42 = 2 %"
[1] "2019-09-15 00:05:35 | 2 out of 42 = 5 %"
[1] "2019-09-15 00:05:35 | 3 out of 42 = 7 %"
[1] "2019-09-15 00:05:36 | 4 out of 42 = 10 %"
[1] "2019-09-15 00:05:36 | 5 out of 42 = 12 %"
[1] "2019-09-15 00:05:37 | 6 out of 42 = 14 %"
[1] "2019-09-15 00:05:37 | 7 out of 42 = 17 %"
[1] "2019-09-15 00:05:38 | 8 out of 42 = 19 %"
[1] "2019-09-15 00:05:38 | 9 out of 42 = 21 %"
[1] "2019-09-15 00:05:39 | 10 out of 42 = 24 %"
[1] "2019-09-15 00:05:39 | 11 out of 42 = 26 %"
[1] "2019-09-15 00:05:40 | 12 out of 42 = 29 %"
[1] "2019-09-15 00:05:40 | 13 out of 42 = 31 %"
[1] "2019-09-15 00:05:41 | 14 out of 42 = 33 %"
[1] "2019-09-15 00:05:41 | 15 out of 42 = 36 %"
[1] "2019-09-15 00:05:42 | 16 out of 42 = 38 %"
[1] "2019-09-15 00:05:42 | 17 out of 42 = 40 %"
Error in .subset2(x, i, exact = exact) : subscript out of bounds
Execution halted
```

Motivation
Logging

```r
sink('/opt/foobar.log', append = TRUE, split = TRUE)
N <- 42
for (i in 1:N) {
    ## do something slow
    print(paste(Sys.time(), '|', i, 'out of', N, '=', round(i / N * 100), '%'))
}
```

Motivation

Logging

```r
sink('/opt/foobar.log', append = TRUE, split = TRUE)
N <- 42
for (i in 1:N) {
    ## do something slow
    print(paste(Sys.time(), '|', i, 'out of', N, '=', round(i / N * 100), '%'))
}
```

```r
logfile <- '/opt/foobar.log'
for (i in 1:N) {
    ## do something slow
    cat(
       paste(Sys.time(), '|', i, 'out of', N, '=', round(i / N * 100), '%'),
       file = logfile, append = TRUE)
}
```

Motivation

Logging

```r
sink('/opt/foobar.log', append = TRUE, split = TRUE)
N <- 42
for (i in 1:N) {
    ## do something slow
    print(paste(Sys.time(), '|', i, 'out of', N, '=', round(i / N * 100), '%'))
}
```

```r
logfile <- '/opt/foobar.log'
for (i in 1:N) {
    ## do something slow
    cat(
       paste(Sys.time(), '|', i, 'out of', N, '=', round(i / N * 100), '%'),
       file = logfile, append = TRUE)
}
```

```r
log <- function(message) {
    cat(paste(Sys.time(), '|', message),
        file = logfile, append = TRUE)
}
```

Motivation

Logging parallel processes

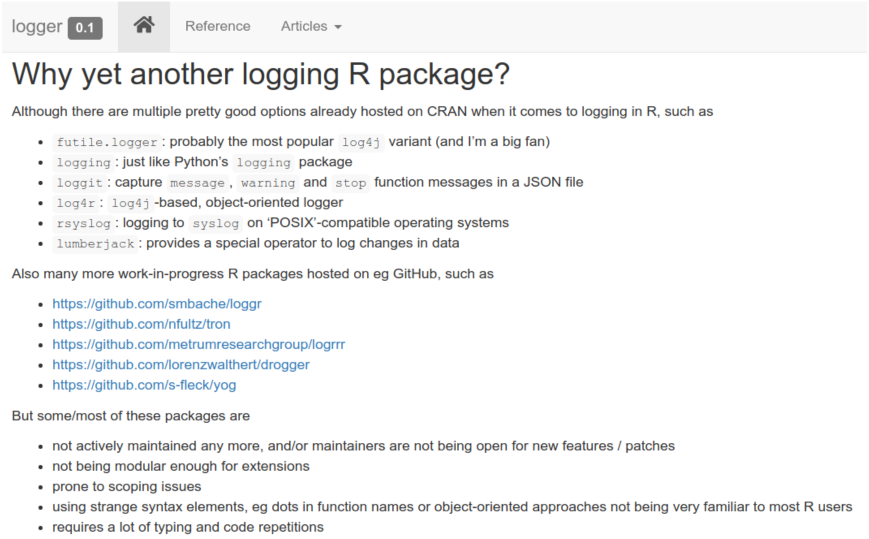
```
mclapply(1:N, function(n) {
    ## do something slow
    log(paste(i, 'out of', N, '=', round(i / N * 100), '%'))
}
[1] "2019-09-15 00:05:34 | 1 out of 42 = 2 %"
[1] "2019-09-15 00:05:35 | 2 out of 42 = 5 %"
[1] "2019-09-15 00:05:39 | 10 out of 42 = 24 %"
[1] "2019-09-15 00:05:35 | 3 out of 42 = 7 %"
[1] "2019-09-15 00:05:39 | 11 out of 42 = 26 %"
[1] "2019-09-15 00:05:36 | 4 out of 42 = 10 %"
[1] "2019-09-15 00:05:40 | 12 out of 42 = 29 %"[1] "2019-09-15 00:05:36 | 5 out of
[1] "2019-09-15 00:05:37 | 6 out of 42 = 14 %"
[1] "2019-09-15 00:05:37 | 7 out of 42 = 17 %"
[1] "2019-09-15 00:05:38 | 8 out of 42 = 19 %"
[1] "2019-09-15 00:05:38 | 9 out of 42 = 21 %"
Error in .subset2(x, i, exact = exact) : subscript out of bounds
Execution halted
```

Logging packages on CRAN

```
> library(data.table)
> packages <- data.table(available.packages())
> ## avoid analog, logit, (archeo|bio|genea|hydro|topo|...)logy
> packages[grepl('(?<!ana)log(?![it|y])', Package, perl = TRUE), Package]

 [1] "adjustedcranlogs"     "bayesloglin"          "blogdown"
 [4] "CommunityCorrelogram" "cranlogs"             "efflog"
 [7] "eMLEloglin"           "futile.logger"        "gemlog"
[10] "gglogo"               "ggseqlogo"            "homologene"
[13] "lifelogr"             "log4r"                "logbin"
[16] "logconcens"           "logcondens"           "logcondens.mode"
[19] "logcondiscr"          "logger"               "logging"
[22] "loggit"               "loggle"               "logKDE"
[25] "loglognorm"           "logmult"              "lognorm"
[28] "logNormReg"           "logOfGamma"           "logspline"
[31] "lolog"                "luzlogr"              "md.log"
[34] "mdir.logrank"         "mpmcorrelogram"       "PhylogeneticEM"
[37] "phylogram"            "plogr"                "poilog"
[40] "rChoiceDialogs"       "reactlog"             "rmetalog"
[43] "robustloggamma"       "rsyslog"              "shinylogs"
[46] "ssrm.logmer"          "svDialogs"            "svDialogstcltk"
[49] "tabulog"              "tidylog"              "wavScalogram"
```

Logging packages on CRAN



Why yet another logging R package?

Although there are multiple pretty good options already hosted on CRAN when it comes to logging in R, such as

- `futile.logger`: probably the most popular `log4j` variant (and I'm a big fan)
- `logging`: just like Python's `logging` package
- `loggit`: capture `message`, `warning` and `stop` function messages in a JSON file
- `log4r`: `log4j`-based, object-oriented logger
- `rsyslog`: logging to `syslog` on 'POSIX'-compatible operating systems
- `lumberjack`: provides a special operator to log changes in data

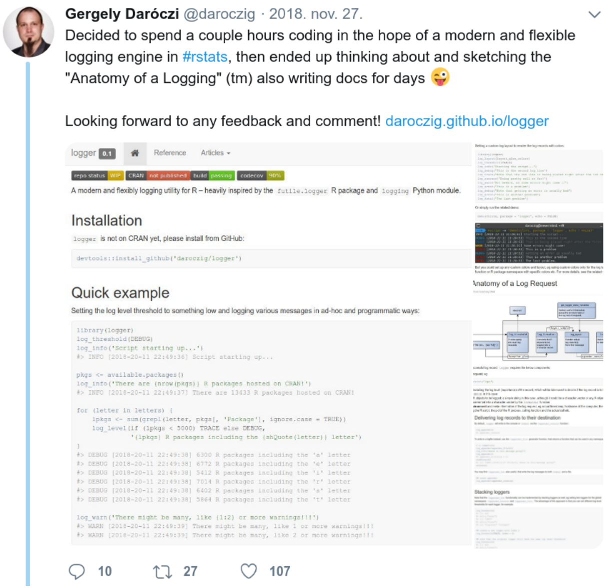
Also many more work-in-progress R packages hosted on eg GitHub, such as

- https://github.com/smbache/loggr
- https://github.com/nfultz/tron
- https://github.com/metrumresearchgroup/logrrr
- https://github.com/lorenzwalthert/drogger
- https://github.com/s-fleck/yog

But some/most of these packages are

- not actively maintained any more, and/or maintainers are not being open for new features / patches
- not being modular enough for extensions
- prone to scoping issues
- using strange syntax elements, eg dots in function names or object-oriented approaches not being very familiar to most R users
- requires a lot of typing and code repetitions

Logging packages on CRAN



Gergely Daróczi @daroczig · 2018. nov. 27.

Decided to spend a couple hours coding in the hope of a modern and flexible logging engine in #rstats, then ended up thinking about and sketching the "Anatomy of a Logging" (tm) also writing docs for days 😜

Looking forward to any feedback and comment! daroczig.github.io/logger

A modern and flexibly logging utility for R – heavily inspired by the `futile.logger` R package and `logging` Python module.

Installation

`logger` is not on CRAN yet, please install from GitHub:

```
devtools::install_github('daroczig/logger')
```

Quick example

Setting the log level threshold to something low and logging various messages in ad-hoc and programmatic ways:

```
library(logger)
log_threshold(DEBUG)
log_info('Script starting up...')
#> INFO [2018-20-11 22:49:36] Script starting up...

pkgs <- available.packages()
log_info('There are {nrow(pkgs)} R packages hosted on CRAN!')
#> INFO [2018-20-11 22:49:37] There are 13433 R packages hosted on CRAN!

for (letter in letters) {
    lpkgs <- sum(grepl(letter, pkgs[, 'Package'], ignore.case = TRUE))
    log_level(if (lpkgs < 5000) TRACE else DEBUG,
              '{lpkgs} R packages including the {shQuote(letter)} letter')
}
#> DEBUG [2018-20-11 22:49:38] 6300 R packages including the 'a' letter
#> DEBUG [2018-20-11 22:49:38] 6772 R packages including the 'e' letter
#> DEBUG [2018-20-11 22:49:38] 5412 R packages including the 'i' letter
#> DEBUG [2018-20-11 22:49:38] 7014 R packages including the 'r' letter
#> DEBUG [2018-20-11 22:49:38] 6402 R packages including the 's' letter
#> DEBUG [2018-20-11 22:49:38] 5864 R packages including the 't' letter

log_warn('There might be many, like {1:2} or more warnings!!!')
#> WARN [2018-20-11 22:49:39] There might be many, like 1 or more warnings!!!
#> WARN [2018-20-11 22:49:39] There might be many, like 2 or more warnings!!!
```

Anatomy of a Log Request

Delivering log records to their destination

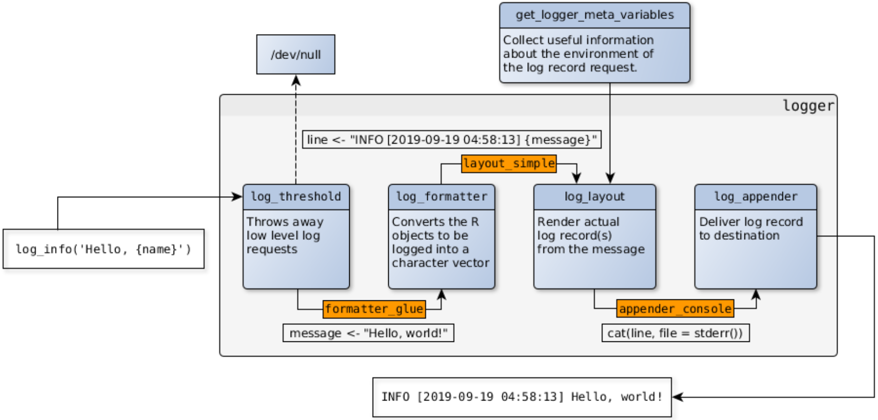
Stacking loggers

The Anatomy of a Log Record

```
> log_info('Hello, {name}!')
INFO [2019-09-19 04:58:13] Hello, world!
```

The Anatomy of a Log Record

```
> log_info('Hello, {name}!')
INFO [2019-09-19 04:58:13] Hello, world!
```



get_logger_meta_variables
Collect useful information about the environment of the log record request.
/dev/null
logger
line <- "INFO [2019-09-19 04:58:13] {message}"
layout_simple
log_threshold
Throws away low level log requests
log_formatter
Converts the R objects to be logged into a character vector
log_layout
Render actual log record(s) from the message
log_appender
Deliver log record to destination
log_info('Hello, {name}')
formatter_glue
message <- "Hello, world!"
appender_console
cat(line, file = stderr())
INFO [2019-09-19 04:58:13] Hello, world!

The Anatomy of a Log Record

A more detailed example

```r
library(logger)
log_threshold(TRACE)
log_formatter(formatter_glue)
log_layout(layout_simple)
log_appender(appender_console)
log_info('Hello, {name}!')
```

The Anatomy of a Log Record

Log threshold

```
> INFO
[1] 400
attr(,"level")
[1] "INFO"
attr(,"class")
[1] "loglevel" "integer"

> TRACE
[1] 600
attr(,"level")
[1] "TRACE"
attr(,"class")
[1] "loglevel" "integer"

> INFO <= TRACE
[1] TRUE
```

The Anatomy of a Log Record

Log threshold

```
> INFO
[1] 400
attr(,"level")
[1] "INFO"
attr(,"class")
[1] "loglevel" "integer"

> TRACE
[1] 600
attr(,"level")
[1] "TRACE"
attr(,"class")
[1] "loglevel" "integer"

> INFO <= TRACE
[1] TRUE
```

```
> name <- 'world'
> log_threshold(TRACE)
> log_info('Hello, {name}!')
INFO [2019-09-18 00:05:32] Hello, world!

> log_threshold(ERROR)
> log_info('Hello, {name}!')
```

The Anatomy of a Log Record

Log message formatter

```
> formatter_glue('Hello, {name}!')
[1] "Hello, world!"
```

The Anatomy of a Log Record

Log message formatter

```
> formatter_glue('Hello, {name}!')
[1] "Hello, world!"
```

```
> formatter_sprintf('Hello, %s!', name)
[1] "Hello, world!"
```

The Anatomy of a Log Record

Log message formatter

```
> formatter_glue('Hello, {name}!')
[1] "Hello, world!"
```

```
> formatter_sprintf('Hello, %s!', name)
[1] "Hello, world!"
```

- `formatter_paste`
- `formatter_sprintf`
- `formatter_glue`
- `formatter_glue_or_sprintf`
- `formatter_logging`

The Anatomy of a Log Record

Log record layout

```
> layout_simple(level = INFO, msg = 'Hello, world!')
[1] "INFO [2019-09-18 00:16:34] Hello, world"
```

The Anatomy of a Log Record

Log record layout

```
> layout_simple(level = INFO, msg = 'Hello, world!')
[1] "INFO [2019-09-18 00:16:34] Hello, world"
```

```
> example_layout <- layout_glue_generator(
>     format = '{node}/{pid}/{ns}/{ans}/{topenv}/{fn} {time} {level}: {msg}')
> example_layout(INFO, 'Hello, world!')
nevermind/3601/NA/global/R_GlobalEnv/NULL 2019-09-18 00:18:11 INFO: Hello, world!
```

The Anatomy of a Log Record

Log record layout

```
> layout_simple(level = INFO, msg = 'Hello, world!')
[1] "INFO [2019-09-18 00:16:34] Hello, world"
```

```
> example_layout <- layout_glue_generator(
>     format = '{node}/{pid}/{ns}/{ans}/{topenv}/{fn} {time} {level}: {msg}')
> example_layout(INFO, 'Hello, world!')
nevermind/3601/NA/global/R_GlobalEnv/NULL 2019-09-18 00:18:11 INFO: Hello, world!
```

```
> logger.tester::logger_info_tester_function('Hello, world!')
nevermind/3601/logger.tester/global/logger.tester/logger.tester::logger_info_tester_
```

The Anatomy of a Log Record

Log record layout

```
> layout_simple(level = INFO, msg = 'Hello, world!')
[1] "INFO [2019-09-18 00:16:34] Hello, world"
```

```
> example_layout <- layout_glue_generator(
>     format = '{node}/{pid}/{ns}/{ans}/{topenv}/{fn} {time} {level}: {msg}')
> example_layout(INFO, 'Hello, world!')
nevermind/3601/NA/global/R_GlobalEnv/NULL 2019-09-18 00:18:11 INFO: Hello, world!
```

```
> logger.tester::logger_info_tester_function('Hello, world!')
nevermind/3601/logger.tester/global/logger.tester/logger.tester::logger_info_tester_
```

```
> layout_json()(level = INFO, msg = 'Hello, world!')
{"time":"2019-09-18 00:19:34","level":"INFO","ns":null,"ans":"global",
 "topenv":"R_GlobalEnv","fn":"cat","node":"nevermind","arch":"x86_64",
 "os_name":"Linux","os_release":"4.15.0-20-generic",
 "os_version":"#21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018",
 "pid":3601,"user":"daroczig","msg":"Hello, world!"}
```

The Anatomy of a Log Record

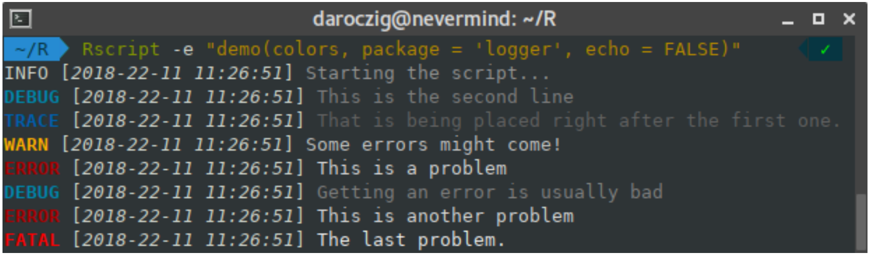
Log record layout

```r
log_layout(layout_glue_colors)
log_info('Starting the script...')
log_debug('This is the second log line')
log_trace('Note that the 2nd line is being placed right after the 1st one.')
log_success('Doing pretty well so far!')
log_warn('But beware, as some errors might come :/')
log_error('This is a problem')
log_debug('Note that getting an error is usually bad')
log_error('This is another problem')
log_fatal('The last problem')
```

The Anatomy of a Log Record

Log record layout

```r
log_layout(layout_glue_colors)
log_info('Starting the script...')
log_debug('This is the second log line')
log_trace('Note that the 2nd line is being placed right after the 1st one.')
log_success('Doing pretty well so far!')
log_warn('But beware, as some errors might come :/')
log_error('This is a problem')
log_debug('Note that getting an error is usually bad')
log_error('This is another problem')
log_fatal('The last problem')
```



```
~/R  Rscript -e "demo(colors, package = 'logger', echo = FALSE)"
INFO [2018-22-11 11:26:51] Starting the script...
DEBUG [2018-22-11 11:26:51] This is the second line
TRACE [2018-22-11 11:26:51] That is being placed right after the first one.
WARN [2018-22-11 11:26:51] Some errors might come!
ERROR [2018-22-11 11:26:51] This is a problem
DEBUG [2018-22-11 11:26:51] Getting an error is usually bad
ERROR [2018-22-11 11:26:51] This is another problem
FATAL [2018-22-11 11:26:51] The last problem.
```

The Anatomy of a Log Record

Log record destination

- `appender_console` / `appender_stderr`
- `appender_stdout`
- `appender_file` (basic log rotating coming soon)
- `appender_tee`

The Anatomy of a Log Record

Log record destination

- `appender_console` / `appender_stderr`
- `appender_stdout`
- `appender_file` (basic log rotating coming soon)
- `appender_tee`

- `appender_slack`
- `appender_telegram`
- `appender_pushbullet`

The Anatomy of a Log Record

Log record destination

- `appender_console` / `appender_stderr`
- `appender_stdout`
- `appender_file` (basic log rotating coming soon)
- `appender_tee`

- `appender_slack`
- `appender_telegram`
- `appender_pushbullet`

- `appender_syslog`
- `appender_kinesis`
- `appender_insert` (DB insert via `dbr` coming soon)

The Anatomy of a Log Record

Log record destination

- `appender_console` / `appender_stderr`
- `appender_stdout`
- `appender_file` (basic log rotating coming soon)
- `appender_tee`

- `appender_slack`
- `appender_telegram`
- `appender_pushbullet`

- `appender_syslog`
- `appender_kinesis`
- `appender_insert` (DB insert via `dbr` coming soon)

- `appender_async`

The Anatomy of a Log Record

Async destination: motivations

```
> appender_file_slow <- function(file) {
+     force(file)
+     function(lines) {
+         Sys.sleep(1)
+         cat(lines, sep = '\n', file = file, append = TRUE)
+     }
+ }
> log_appender(appender_file_slow(tempfile()))
> system.time(for (i in 1:25) log_info(i))
```

The Anatomy of a Log Record

Async destination: motivations

```
> appender_file_slow <- function(file) {
+     force(file)
+     function(lines) {
+         Sys.sleep(1)
+         cat(lines, sep = '\n', file = file, append = TRUE)
+     }
+ }
> log_appender(appender_file_slow(tempfile()))
> system.time(for (i in 1:25) log_info(i))
```

```
   user  system elapsed
  0.057   0.002  25.083
```

The Anatomy of a Log Record

Async destination: how?

- create a local, disk-based storage for the message queue via `txtq`
- start a background process for the async execution of the message queue with `callr`
- loads minimum required packages in the background process
- connects to the message queue from the background process
- pass actual `appender` function to the background process (serialized to disk)
- pass parameters of the async appender to the background process (eg batch size)
- start infinite loop processing log records
- check if background process still works ...

The Anatomy of a Log Record

Async destination: preparations

```
> appender_file_slow <- function(file) {
+     force(file)
+     function(lines) {
+         Sys.sleep(1)
+         cat(lines, sep = '\n', file = file, append = TRUE)
+     }
+ }
```

```
> ## log what's happening in the background
> log_threshold(TRACE, namespace = 'async_logger')
> log_appender(appender_console, namespace = 'async_logger')

> ## start async appender
> t <- tempfile()
> log_info('Logging in the background to {t}')
TRACE [2019-09-18 02:57:11] Logging in the background to /tmp/RtmpLW4bY4/file63ff7f
> my_appender <- appender_async(appender_file_slow(file = t))
TRACE [2019-09-18 02:57:11] Async writer storage: /tmp/RtmpLW4bY4/file63ff6bf714c2
TRACE [2019-09-18 02:57:11] Async writer PID: 29378
TRACE [2019-09-18 02:57:11] Async appender cached at: /tmp/RtmpLW4bY4/file63ff7a2eb
```

The Anatomy of a Log Record

Async destination: usage

```
> ## use async appander
> log_appender(my_appender)
> log_info('Was this slow?')
> system.time(for (i in 1:25) log_info(i))
   user  system elapsed
   0.02    0.00    0.02

> Sys.sleep(1)
> readLines(t)
[1] "INFO [2019-09-18 02:57:12] Was this slow?"
> Sys.sleep(5)
> readLines(t)
[1] "INFO [2019-09-18 02:57:12] Was this slow?"
[2] "INFO [2019-09-18 02:57:12] 1"
[3] "INFO [2019-09-18 02:57:12] 2"
[4] "INFO [2019-09-18 02:57:12] 3"
[5] "INFO [2019-09-18 02:57:12] 4"
[6] "INFO [2019-09-18 02:57:12] 5"
```

The Anatomy of a Log Record

Async destination: debugging with ‘txtq‘ and ‘callr‘

```
> attr(my_appender, 'async_writer_queue')$count()
[1] 0
> attr(my_appender, 'async_writer_queue')$log()
             title                                 message
1  1568768232.15263 INFO [2019-09-18 02:57:12] Was this slow? 2019-09-18 02:57:12.1
2  1568768232.22928              INFO [2019-09-18 02:57:12] 1 2019-09-18 02:57:12.2
3  1568768232.23001              INFO [2019-09-18 02:57:12] 2 2019-09-18 02:57:12.2
4   1568768232.2307              INFO [2019-09-18 02:57:12] 3 2019-09-18 02:57:12.2
5  1568768232.23142              INFO [2019-09-18 02:57:12] 4 2019-09-18 02:57:12.2
...

> attr(my_appender, 'async_writer_process')$get_pid()
[1] 29378
> attr(my_appender, 'async_writer_process')$get_state()
[1] "busy"
> attr(my_appender, 'async_writer_process')$poll_process(1)
[1] "timeout"
> attr(my_appender, 'async_writer_process')$read()
NULL

> attr(my_appender, 'async_writer_process')$is_alive()
[1] TRUE
```

A *logger* definition

- log threshold
- log message formatter
- log record layout
- log record destination(s)

A *logger* definition

- log threshold(s)
- log message formatter(s)
- log record layout(s)
- log record destination(s)

logger namespaces

What goes where

```
> log_appender(appender_stderr)
> log_threshold(INFO)

> my_appender <- appender_async(appender_slack(channel = '#foobar', token = '...'))
> log_appender(my_appender, namespace = 'slack')
> log_threshold(WARN, namespace = 'slack')
```

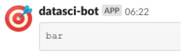
logger namespaces

What goes where

```
> log_appender(appender_stderr)
> log_threshold(INFO)

> my_appender <- appender_async(appender_slack(channel = '#foobar', token = '...'))
> log_appender(my_appender, namespace = 'slack')
> log_threshold(WARN, namespace = 'slack')
```

```
> log_info('foo')
INFO [2019-09-19 06:15:22] foo
> log_error('bar', namespace = 'slack')
```



datasci-bot APP 06:22

```
bar
```

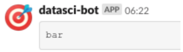
logger namespaces

What goes where

```
> log_appender(appender_stderr)
> log_threshold(INFO)

> my_appender <- appender_async(appender_slack(channel = '#foobar', token = '...'))
> log_appender(my_appender, namespace = 'slack')
> log_threshold(WARN, namespace = 'slack')
```

```
> log_info('foo')
INFO [2019-09-19 06:15:22] foo
> log_error('bar', namespace = 'slack')
```



datasci-bot APP 06:22

```
bar
```

- R packages using `logger` automatically gets their own namespace, so eg `dbr` using `logger` can be silenced by

```
log_threshold(FATAL, namespace = 'dbr')
```

logger namespaces

Stacking loggers

```
> log_appender(appender_stderr)
> log_threshold(INFO)

> log_appender(appender_file(file = '/var/log/myapp.log'), index = 2)
> log_threshold(TRACE, index = 2)

> my_appender <- appender_async(appender_slack(channel = '#foobar', token = '...'))
> log_appender(my_appender, index = 3)
> log_threshold(ERROR, index = 3)
```

logger namespaces

Stacking loggers

```
> log_appender(appender_stderr)
> log_threshold(INFO)

> log_appender(appender_file(file = '/var/log/myapp.log'), index = 2)
> log_threshold(TRACE, index = 2)

> my_appender <- appender_async(appender_slack(channel = '#foobar', token = '...'))
> log_appender(my_appender, index = 3)
> log_threshold(ERROR, index = 3)
```

```
> log_info('foo')
INFO [2019-09-19 06:15:22] foo
> log_error('bar')
ERROR [2019-09-19 06:15:22] bar

> readLines('/var/log/yapp.log')
[1] "INFO [2019-09-19 06:15:22] foo"  "ERROR [2019-09-19 06:15:22] bar"
```

logger helper functions

```
> f <- sqrt
> g <- mean
> x <- 1:31
> log_eval(f(g(x)), level = INFO)
INFO [2019-09-19 04:38:17] 'f(g(x))' => '4'
[1] 4
```

logger helper functions

```
> f <- sqrt
> g <- mean
> x <- 1:31
> log_eval(f(g(x)), level = INFO)
INFO [2019-09-19 04:38:17] 'f(g(x))' => '4'
[1] 4
```

```
> log_failure('foobar')
[1] "foobar"
> log_failure(foobar)
ERROR [2019-09-19 04:39:27] object 'foobar' not found
Error in doTryCatch(return(expr), name, parentenv, handler) :
  object 'foobar' not found
```

logger helper functions

```
> f <- sqrt
> g <- mean
> x <- 1:31
> log_eval(f(g(x)), level = INFO)
INFO [2019-09-19 04:38:17] 'f(g(x))' => '4'
[1] 4
```

```
> log_failure('foobar')
[1] "foobar"
> log_failure(foobar)
ERROR [2019-09-19 04:39:27] object 'foobar' not found
Error in doTryCatch(return(expr), name, parentenv, handler) :
  object 'foobar' not found
```

```
> log_tictoc('warming up')
INFO [2019-09-19 04:38:56] global timer tic 0 secs -- warming up
> Sys.sleep(0.1)
> log_tictoc('running')
INFO [2019-09-19 04:38:57] global timer toc 1.27 secs -- running
> Sys.sleep(0.1)
> log_tictoc('running')
INFO [2019-09-19 04:38:59] global timer toc 1.36 secs -- running
> Sys.sleep(runif(1))
> log_tictoc('and running')
```

logger helper functions

```
> log_messages()
> message('hi there')
hi there
INFO [2019-09-19 05:41:29] hi there

> log_warnings()
> for (i in 1:5) {
+     Sys.sleep(runif(1))
+     suppressWarnings(warning(i))
+ }
WARN [2019-09-19 05:41:32] 1
WARN [2019-09-19 05:41:33] 2
WARN [2019-09-19 05:41:33] 3
WARN [2019-09-19 05:41:34] 4
WARN [2019-09-19 05:41:34] 5

> log_errors()
> stop('foobar')
ERROR [2019-09-19 05:41:37] foobar
Error: foobar
```

logger helper functions

```r
library(shiny)
ui <- bootstrapPage(
    numericInput('mean', 'mean', 0),
    numericInput('sd', 'sd', 1),
    textInput('title', 'title', 'title'),
    plotOutput('plot')
)
server <- function(input, output) {
    logger::log_shiny_input_changes(input)
    output$plot <- renderPlot({
        hist(rnorm(1e3, input$mean, input$sd), main = input$title)
    })
}
shinyApp(ui = ui, server = server)
```

logger helper functions

```r
library(shiny)
ui <- bootstrapPage(
    numericInput('mean', 'mean', 0),
    numericInput('sd', 'sd', 1),
    textInput('title', 'title', 'title'),
    plotOutput('plot')
)
server <- function(input, output) {
    logger::log_shiny_input_changes(input)
    output$plot <- renderPlot({
        hist(rnorm(1e3, input$mean, input$sd), main = input$title)
    })
}
shinyApp(ui = ui, server = server)
```

```
Listening on http://127.0.0.1:8080
INFO [2019-07-11 16:59:17] Default Shiny inputs initialized: {"mean":0,"title":"title","sd":1}
INFO [2019-07-11 16:59:26] Shiny input change detected on mean: 0 -> 1
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 1 -> 2
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 2 -> 3
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 3 -> 4
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 4 -> 5
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 5 -> 6
INFO [2019-07-11 16:59:27] Shiny input change detected on mean: 6 -> 7
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 1 -> 2
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 2 -> 3
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 3 -> 4
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 4 -> 5
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 5 -> 6
INFO [2019-07-11 16:59:29] Shiny input change detected on sd: 6 -> 7
INFO [2019-07-11 16:59:34] Shiny input change detected on title: title -> sfdsadsads
```

Using *logger* in R packages

A 'boto3' wrapper

```
> remotes::install_github('daroczig/botor')
> library(botor)
> my_mtcars <- s3_read('s3://botor/example-data/mtcars.csv', read.csv)
DEBUG [2019-09-19 04:46:57] Downloaded 1303 bytes from s3://botor/example-data/mtca
and saved at '/tmp/RtmpLW4bY4/file63ff42ed2fe1'

> log_threshold(TRACE, namespace = 'botor')
> my_mtcars <- s3_read('s3://botor/example-data/mtcars.csv.gz',
+                      read.csv, extract = 'gzip')
TRACE [2019-09-19 04:48:02] Downloading s3://botor/example-data/mtcars.csv.gz to
'/tmp/RtmpLW4bY4/file63ff17e137e9' ...
DEBUG [2019-09-19 04:48:03] Downloaded 567 bytes from s3://botor/example-data/mtcars
and saved at '/tmp/RtmpLW4bY4/file63ff17e137e9'
TRACE [2019-09-19 04:48:03] Decompressed /tmp/RtmpLW4bY4/file63ff17e137e9 via gzip
from 567 to 1303 bytes
TRACE [2019-09-19 04:48:03] Deleted /tmp/RtmpLW4bY4/file63ff17e137e9

> log_threshold(ERROR, namespace = 'botor')
> my_mtcars <- s3_read('s3://botor/example-data/mtcars.csv.gz',
+                      read.csv, extract = 'gzip')
```

Using *logger* in R packages

A convenient (and secure) DB connection manager

```
sqlite:
  drv: !expr RSQLite::SQLite()
  dbname: !expr tempfile()
```

Using *logger* in R packages

A convenient (and secure) DB connection manager

```
sqlite:
  drv: !expr RSQLite::SQLite()
  dbname: !expr tempfile()
```

```
> library(dbr)
> str(db_query('SELECT 42', 'sqlite'))

INFO [2018-07-11 17:07:12] Connecting to sqlite
INFO [2018-07-11 17:07:12] Executing:**********
INFO [2018-07-11 17:07:12] SELECT 42
INFO [2018-07-11 17:07:12] ********************
INFO [2018-07-11 17:07:12] Finished in 0.0007429 secs returning 1 rows
INFO [2018-07-11 17:07:12] Closing connection to sqlite

'data.frame':   1 obs. of  1 variable:
 $ 42: int 42
 - attr(*, "when")= POSIXct, format: "2018-07-11 17:07:12"
 - attr(*, "db")= chr "sqlite"
 - attr(*, "time_to_exec")=Class 'difftime'  atomic [1:1] 0.000743
  .. ..- attr(*, "units")= chr "secs"
 - attr(*, "statement")= chr "SELECT 42"
```

Using *logger* in R packages

A convenient (and secure) DB connection manager

```
default:
  shinydemo:
    drv: !expr RMySQL::MySQL()
    host: shiny-demo.csa7qlmguqrf.us-east-1.rds.amazonaws.com
    username: guest
    password: guest
    dbname: shinydemo
```

Using *logger* in R packages

A convenient (and secure) DB connection manager

```
default:
  shinydemo:
    drv: !expr RMySQL::MySQL()
    host: shiny-demo.csa7qlmguqrf.us-east-1.rds.amazonaws.com
    username: guest
    password: guest
    dbname: shinydemo
```

```
shinydemo:
  drv: !expr RMySQL::MySQL()
  host: !kms |
    AQICAHiMkU2ZNbL+kRcQoM3wGpuLb8HbIKjM9VcEGt72rZV2SAEXX7aTXvtsf91BzgoiiIDh
    AAAAlDCBkQYJKoZIhvcNAQcGoIGDMIGAAgEAMHsGCSqGSIb3DQEHATAeBglghkgBZQMEAS4w
    EQQMgVoMPjgAi+S7i7cvAgEQgE5X4dnyt/Tl0+PiX/yjzdC2wYl+tWzvHnApAhIahQroK+VJ
    8OQEQse/s/VE6n2gHPuXe4c/9lK9Od6e1aR8+YZCflyOA5F2sWFz6+hU5XI=
  username: !kms |
    AQICAHiMkU2ZNbL+kRcQoM3wGpuLb8HbIKjM9VcEGt72rZV2SAE6IQVMFPyj9JBP7cEgf9oT
    AAAAYzBhBgkqhkiG9w0BBwagVDBSAgEAME0GCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQM
    Q8zMzSSMTX0UzT0dAgEQgCBlwaYQyO29zKbtIBuQtSHBWxqgyu49/lUQKZn8CCwmyQ==
  password: !kms |
    AQICAHiMkU2ZNbL+kRcQoM3wGpuLb8HbIKjM9VcEGt72rZV2SAE6IQVMFPyj9JBP7cEgf9oT
    AAAAYzBhBgkqhkiG9w0BBwagVDBSAgEAME0GCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQM
    Q8zMzSSMTX0UzT0dAgEQgCBlwaYQyO29zKbtIBuQtSHBWxqgyu49/lUQKZn8CCwmyQ==
```

Using *logger* in R packages

A convenient (and secure) DB connection manager

```
> db_query(
+     sql = "SELECT Continent, COUNT(DISTINCT(Region)) FROM Country GROUP BY Contin
+     db = 'shinydemo')
INFO [2019-09-19 05:02:30] Looking up config for shinydemo

INFO [2019-09-19 05:02:30] Decrypting string via KMS ...
INFO [2019-09-19 05:02:30] Decrypting string via KMS ...
INFO [2019-09-19 05:02:31] Decrypting string via KMS ...

INFO [2019-09-19 05:02:31] Connecting to shinydemo
INFO [2019-09-19 05:02:32] Executing:**********
INFO [2019-09-19 05:02:32] SELECT Continent, COUNT(DISTINCT(Region)) FROM Country G
INFO [2019-09-19 05:02:32] ********************
INFO [2019-09-19 05:02:32] Finished in 0.2213 secs returning 7 rows
INFO [2019-09-19 05:02:32] Closing connection to shinydemo
       Continent COUNT(DISTINCT(Region))
1:          Asia                       4
2:        Europe                       6
3: North America                       3
4:        Africa                       5
5:       Oceania                       5
6:    Antarctica                       1
7: South America                       1
```

References

- daroczig/logger
- daroczig/botor
- daroczig/dbr