

Logging and Analyzing Events in Complex Shiny Apps

with **shinyEventLogger**

Kamil Wais

UseR! - Toulouse - 2019



1 / 32

Why to log?

2 / 32

Why to log?

- logging in development:
 - shows which events are firing (or not)
 - shows in what sequence events are firing
 - shows why events fired & within what context
 - helps with **debugging & development**
- logging in production:
 - shows how users use the app
 - helps to see usage patterns
 - helps with app **evaluation & optimization**

3 / 32

What do we need?

4 / 32

What do we need?

- logging tool **dedicated to** complex **shiny apps**
- ability to see most recent events in **real-time**
- quick & easy **access** to eventlogs
- **permanent storage** for eventlogs
- tools to **analyse** the eventlogs

lifecycle experimental

shinyEventLogger

R Package for Logging Events in Complex Shiny Apps

5 / 32

What type of events to log?

6 / 32

Logging generic events

```
library(shiny)
library(shinyEventLogger)
set_logging()
shinyApp(
  ui = fluidPage(log_init()),
  server = function(input, output) {
    set_logging_session()
    log_event("Event body", name = "Event name")
    log_event("Event without explicit name")
    x <- "123"
    log_event("Event", x)
  }
)
```

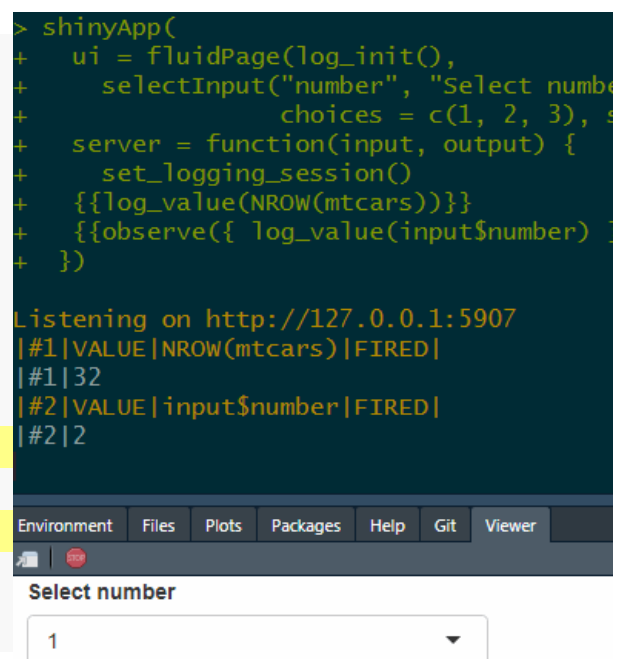
```
|#1|EVENT|Event name|FIRED|
|#1|Event body
|#2|EVENT|Event without explicit name|FIRED|
|#3|EVENT|Event 123|FIRED|
```

7 / 32

Logging values

```
set_logging()
shinyApp(
  ui = fluidPage(log_init(),
    selectInput("number",
      "Select number",
      choices = c(1, 2, 3),
      selected = 2)),
  server = function(input, output) {
    set_logging_session()
    log_value(NROW(mtcars))
    observe({
      log_value(input$number)
    })
  })
})
```

```
|#1|VALUE|NROW(mtcars)|FIRED|
|#1|32
|#2|VALUE|input$number|FIRED|
|#2|2
```

The screenshot shows a Shiny application window. The top part is the console, which displays the R code for the app and its output. The code includes a select input and two logging calls. The output shows the app listening on port 5907 and logging the number of rows in mtcars (32) and the selected number (2). The bottom part of the window is the user interface, which features a dropdown menu titled "Select number" with the value "1" selected. The window has tabs for Environment, Files, Plots, Packages, Help, Git, and Viewer.

```
> shinyApp(
+   ui = fluidPage(log_init(),
+     selectInput("number", "Select number",
+       choices = c(1, 2, 3),
+       selected = 2)),
+   server = function(input, output) {
+     set_logging_session()
+     {{log_value(NROW(mtcars))}}
+     {{observe({ log_value(input$number) })}}
+   })
Listening on http://127.0.0.1:5907
|#1|VALUE|NROW(mtcars)|FIRED|
|#1|32
|#2|VALUE|input$number|FIRED|
|#2|2
```

Environment Files Plots Packages Help Git Viewer

Select number

1

8 / 32

Logging function outputs

```
set_logging()
shinyApp(
  ui = fluidPage(log_init()),
  server = function(input, output) {
    set_logging_session()
    log_output(head(mtcars, 2))
    log_output(str(mtcars[, 1:3]))
  })
```

```
|#1|OUTPUT|head(mtcars, 2)|FIRED|
|#1|          mpg cyl disp  hp drat   wt  qsec vs am gear carb
|#1|Mazda RX4         21   6  160 110   3.9 2.620 16.46  0  1    4    4
|#1|Mazda RX4 Wag    21   6  160 110   3.9 2.875 17.02  0  1    4    4
|#2|OUTPUT|str(mtcars[, 1:3])|FIRED|
|#2|'data.frame':    32 obs. of  3 variables:
|#2| $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
|#2| $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
|#2| $ disp: num  160 160 108 258 360 ...
```

9 / 32

Logging outcomes of unit tests

```
set_logging()
shinyApp(
  ui = fluidPage(log_init()),
  server = function(input, output) {
    set_logging_session()
    log_test(testthat::expect_true(TRUE))
    log_test(testthat::expect_true(FALSE))
  })
```

```
|#1|TEST|testthat::expect_true(TRUE)|SUCCESS|
|#2|TEST|testthat::expect_true(FALSE)|ERROR|
|#2|Error: FALSE isn't true.
```

10 / 32

Logging traditional diagnostic messages

```
set_logging()
shinyApp(
  ui = fluidPage(log_init()),
  server = function(input, output) {
    set_logging_session()
    log_message("Example of a message")
    log_warning("Example of a warning")
    log_error("Example of an error")
  })
```

```
|#1|MESSAGE|Example of a message|FIRED|
Example of a message
|#2|WARNING|Example of a warning|FIRED|
Warning in log_warning("Example of a warning") : Example of a warning
|#3|ERROR|Example of an error|FIRED|
Warning: Error in log_error: Example of an error
52: stop
```

11 / 32

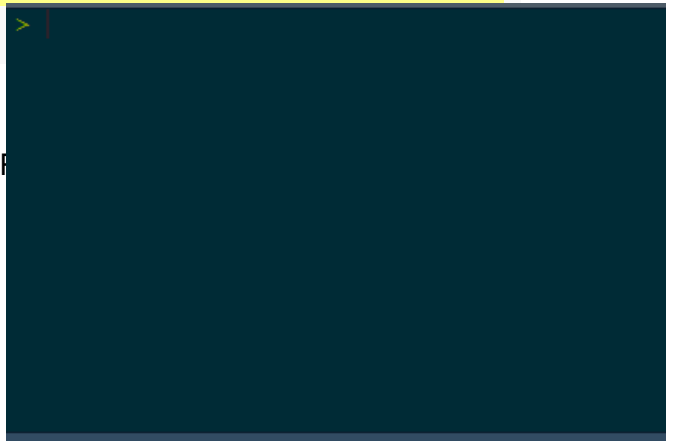
Timing events

12 / 32

Logging & timing events

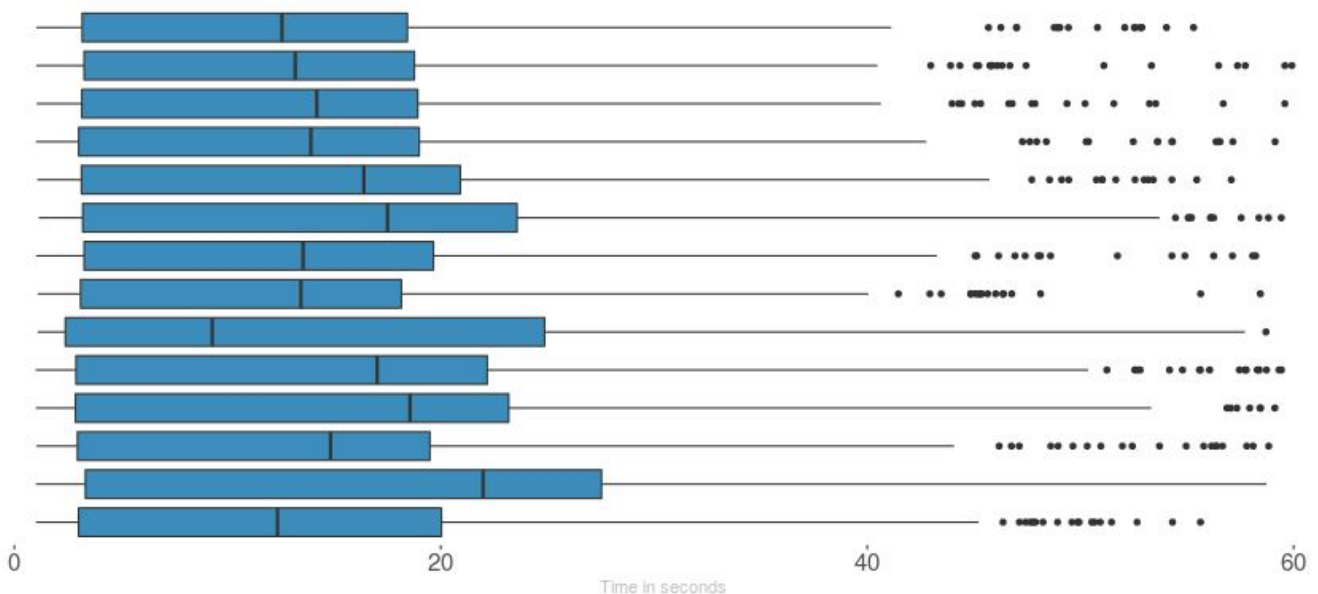
```
shinyApp(  
  ui = shiny::fluidPage(log_init()),  
  server = function(input, output) {  
    set_logging_session()  
    log_started("Event 1")  
    log_event("Sleeping for 3 secs...")  
    Sys.sleep(3)  
    log_done("Event 1")  
  })
```

```
|#1|EVENT|Event 1|STARTED|  
|#2|EVENT|Sleeping for 3 secs...|FINISHED|  
|#1|EVENT|Event 1|DONE|  
|#1|PARAMS|list(secs = 3)
```



13 / 32

Logging & timing in parallel R processes



14 / 32

Logging the context of the events

15 / 32

Logging events with parameters

```
set_logging(app = "demo")
shinyApp(
  ui = fluidPage(log_init(),
    selectInput("number", "Select number", choices = c(1, 2))
  server = function(input, output) {
    set_logging_session(unique_id = runif(1))
    observe({
      log_event("number changed",
        params = list(number = input$number))
    })
  })
})
```

```
|#1|EVENT|number changed|FIRED|
|#1|PARAMS|list(number = "2", unique_id = 0.104973568115383, app = "d
```

16 / 32

Logging events with parameters

```
+ log_init(),  
+ selectInput("number", "Select number", choices = c(1, 2, 3), selected = 2)),  
+ server = function(input, output) {  
+   {{set_logging_session(unique_id = runif(1))}}  
+   observe({  
+     {{log_event("number changed", params = list(number = input$number))}}  
+   })  
+ })
```

Listening on http://127.0.0.1:7185

|#1|EVENT|number changed|FIRED|

|#1|PARAMS|list(number = "2", unique_id = 0.54154440946877, app = "demo")

Environment Files Plots Packages Help Git Viewer

Publish

Select number

1

2

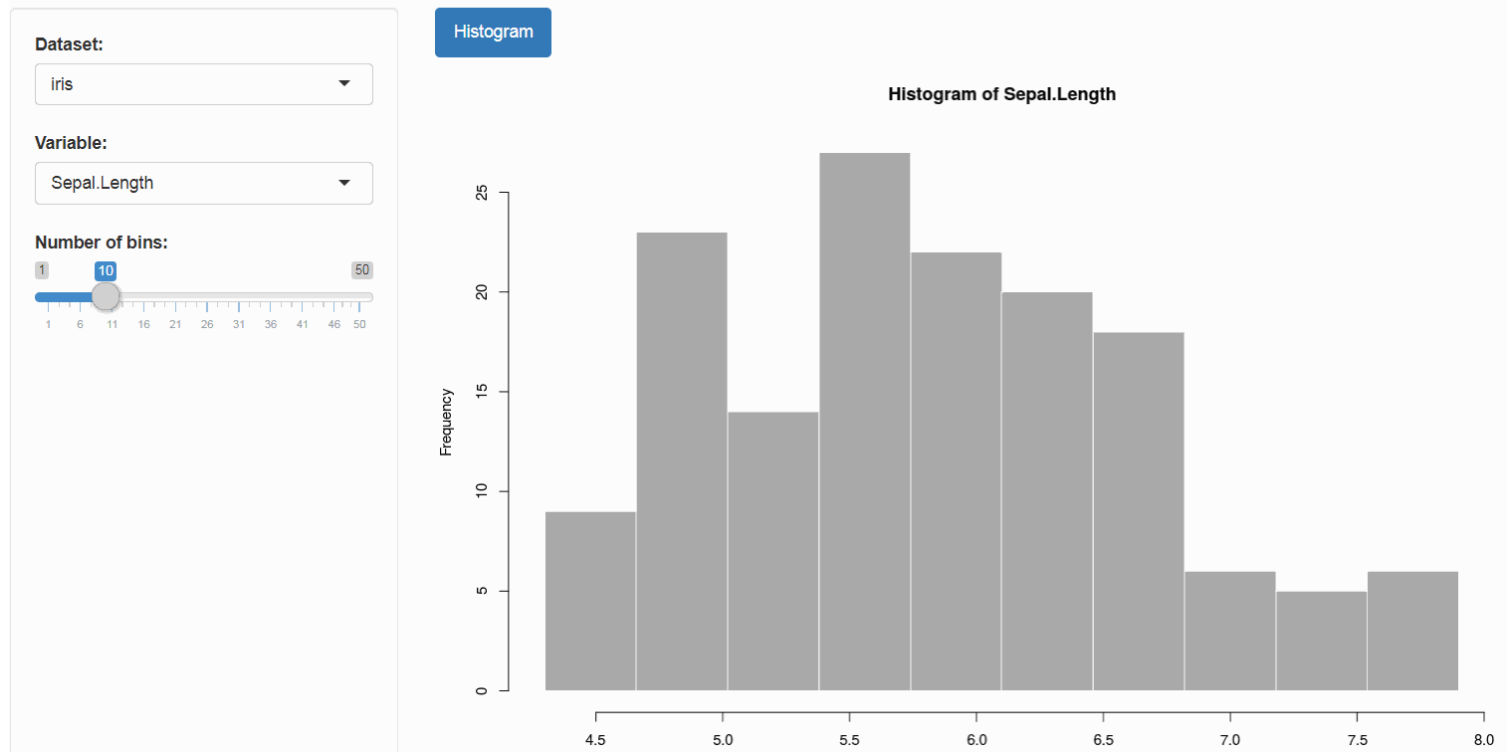
17 / 32

Where to log?

18 / 32

Demo app with event logging

ShinyEventLogger: DEMO APP

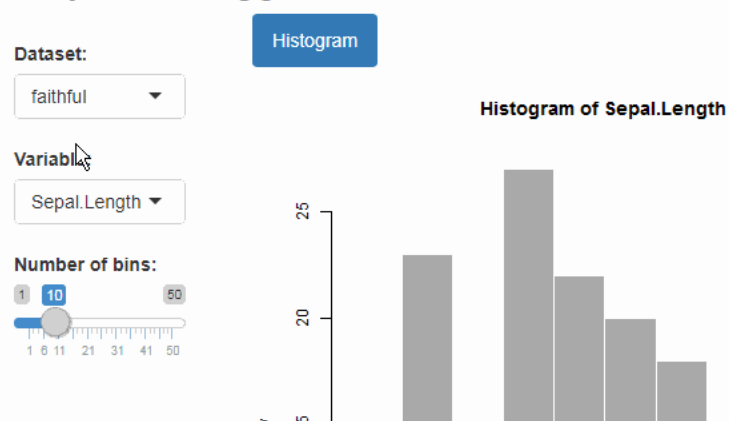


19 / 32

R console

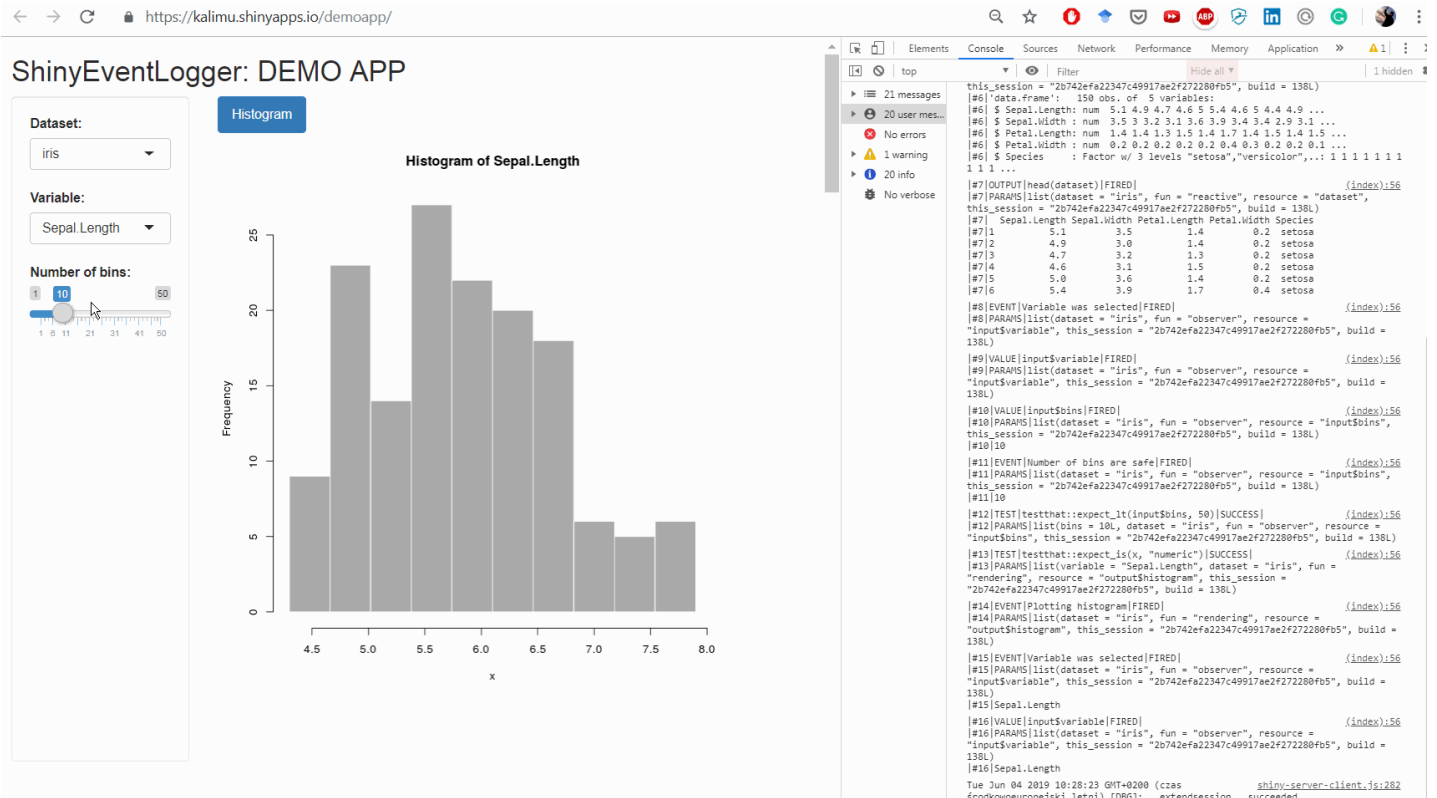
```
#62|VALUE|input$bins|FIRED|
|#62|PARAMS|list(dataset = "iris", fun = "observer", resource = "input$bins")
|#62|10
|#63|EVENT|Number of bins are safe|FIRED|
|#63|PARAMS|list(dataset = "iris", fun = "observer", resource = "input$bins")
|#63|10
|#64|TEST|testthat::expect_lt(input$bins, 50)|SUCCESS|
|#64|PARAMS|list(bins = 10L, dataset = "iris", fun = "observer", resource = "input$bins")
|#65|TEST|testthat::expect_is(x, "numeric")|SUCCESS|
|#65|PARAMS|list(variable = "Sepal.Length", dataset = "iris", fun = "rendering", resource = "output$histogram")
|#66|EVENT|Plotting histogram|FIRED|
|#66|PARAMS|list(dataset = "iris", fun = "rendering", resource = "output$histogram")
|#67|EVENT|Variable was selected|FIRED|
|#67|PARAMS|list(dataset = "iris", fun = "observer", resource = "input$variable")
|#67|Sepal.Length
|#68|VALUE|input$variable|FIRED|
|#68|PARAMS|list(dataset = "iris", fun = "observer", resource = "input$variable")
|#68|Sepal.Length
```

ShinyEventLogger: DEMO APP



20 / 32

JavaScript console



21 / 32

Text file

```

#1|EVENT|App (re)started|FIRED|list(build = 139L)|186706c08e00bd1181
#2|EVENT|Dataset was selected|FIRED|list(dataset = "iris", fun = "ob
#3|VALUE|input$dataset|FIRED|list(dataset = "iris", fun = "observer"
#4|EVENT|Loading dataset|STARTED|list(dataset = "iris", fun = "react
#4|EVENT|Loading dataset|DONE|list(secs = 0.05, dataset = "iris", fu
#5|VALUE|NROW(dataset)|FIRED|list(n_rows = 150L, dataset = "iris", f

```

```

events.log
1 | #1|EVENT|App (re)started|FIRED|list(build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:13|
2 | #2|EVENT|Dataset was selected|FIRED|list(dataset = "iris", fun = "observer", resource = "input$dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:13|iris|
3 | #3|VALUE|input$dataset|FIRED|list(dataset = "iris", fun = "observer", resource = "input$dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:13|iris|
4 | #4|EVENT|Loading dataset|STARTED|list(dataset = "iris", fun = "reactive", resource = "dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:13|
5 | #4|EVENT|Loading dataset|DONE|list(secs = 0.05, dataset = "iris", fun = "reactive", resource = "dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:13|
6 | #5|VALUE|NROW(dataset)|FIRED|list(n_rows = 150L, dataset = "iris", fun = "reactive", resource = "dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|150
7 | #6|OUTPUT[head(dataset)]|FIRED|list(dataset = "iris", fun = "reactive", resource = "dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|data.frame: 150 o
variables: $ Sepal.Length: num 5.1 4.9 4.7 4.6 5.5 4.4 6.5 4.4 4.9 ... $ Sepal.Width: num 3.5 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ... $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
8 | #7|OUTPUT[head(dataset)]|FIRED|list(dataset = "iris", fun = "reactive", resource = "dataset", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14| Sepal.Length Sep
Petal.Length Petal.Width Species 1 5.1 3.5 1.4 0.2 setosa 2 4.9 3.0 1.4 0.2 setosa 3 4.7 3.2 1.3 0.2 setosa 4 4.6 3.1 1.5 0.2 setosa 5 5.0 3.6 1.4 0.2 setosa 6 5.4 3.9 1.7 0.4 setosa
9 | #8|EVENT|Variable was selected|FIRED|list(dataset = "iris", fun = "observer", resource = "input$variable", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|
10 | #9|VALUE[input$variable]|FIRED|list(dataset = "iris", fun = "observer", resource = "input$variable", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|
11 | #10|VALUE[input$bins]|FIRED|list(dataset = "iris", fun = "observer", resource = "input$bins", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|10|
12 | #11|EVENT|Number of bins are safe|FIRED|list(dataset = "iris", fun = "observer", resource = "input$bins", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|10|
13 | #12|TEST[testthat::expect_lt(input$bins, 50)]|SUCCESS|list(bins = 10L, dataset = "iris", fun = "observer", resource = "input$bins", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|
14 | #13|EVENT|Variable was selected|FIRED|list(dataset = "iris", fun = "observer", resource = "input$variable", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|Sep
15 | #14|VALUE[input$variable]|FIRED|list(dataset = "iris", fun = "observer", resource = "input$variable", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|Sepal.Leng
16 | #15|TEST[testthat::expect_is(x, "numeric")]|SUCCESS|list(variable = "Sepal.Length", dataset = "iris", fun = "rendering", resource = "output$histogram", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:14|
17 | #16|EVENT|Plotting histogram|FIRED|list(dataset = "iris", fun = "rendering", resource = "output$histogram", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:15|
18 | #17|VALUE[input$bins]|FIRED|list(dataset = "iris", fun = "observer", resource = "input$bins", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:19|22|
19 | #18|TEST[testthat::expect_lt(input$bins, 50)]|SUCCESS|list(bins = 22L, dataset = "iris", fun = "observer", resource = "input$bins", build = 139L)|186706c08e00bd118152e7b70c0ab64e|2019-01-29 07:15:19|22|

```

22 / 32

MongoDB

```
_id: ObjectId("5c4ffd81b44fe82424003df8")
event_counter: 2
event_type: "EVENT"
event_name: "Dataset was selected"
event_status: "FIRED"
✓ event_params: Object
  ✓ dataset: Array
    0: "iris"
  ✓ fun: Array
    0: "observer"
  ✓ resource: Array
    0: "input$dataset"
  ✓ build: Array
    0: 139
event_body: "iris"
event_timestamp: 2019-01-29T07:15:13.772+00:00
session_id: "186706c08e00bd118152e7b70c0ab64e"
```

23 / 32

How to analyze the eventlog?

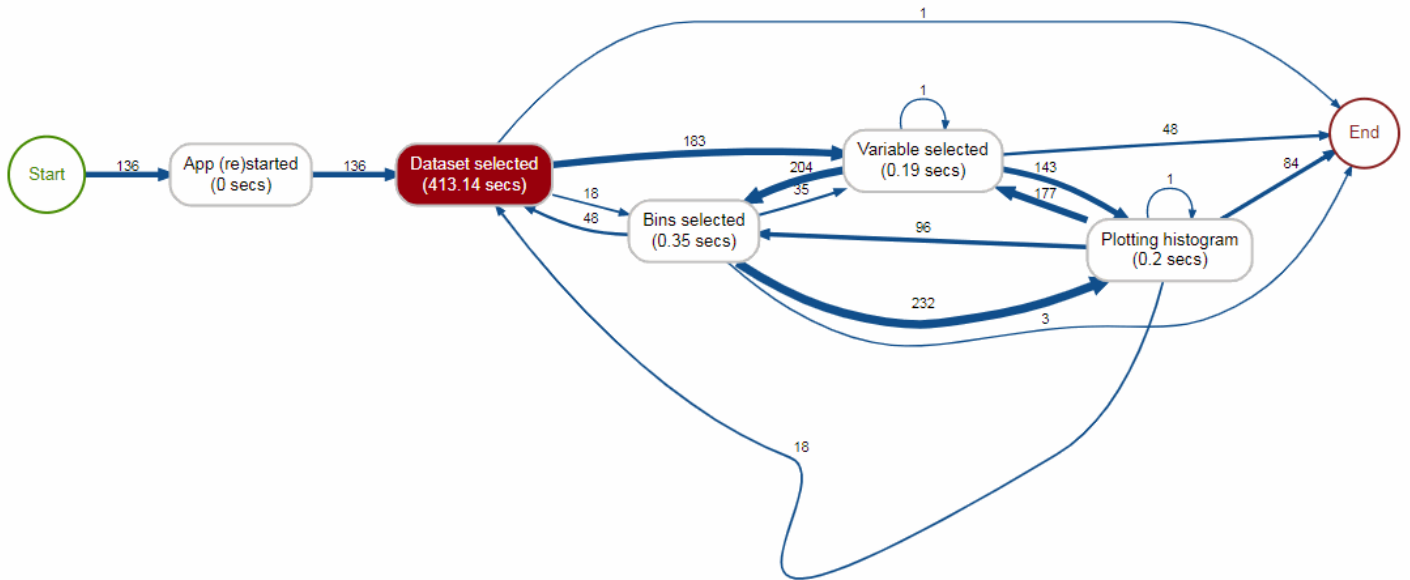
with process-mining techniques (bupaR package)

24 / 32

Process maps

- patterns and paths in end-users behaviors

Maximum time



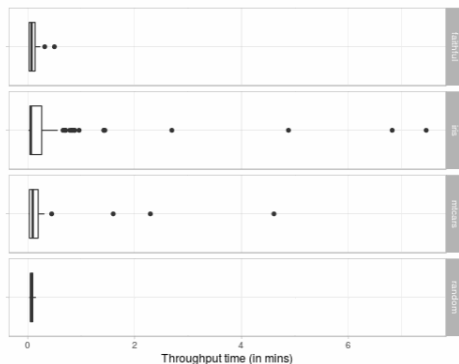
Mean time

25 / 32

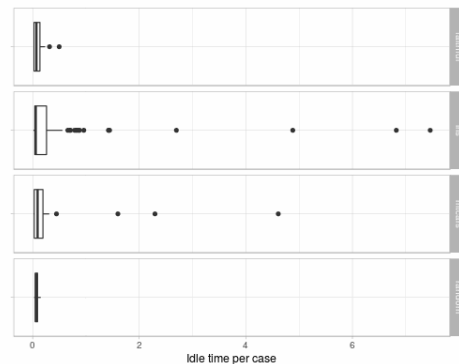
Time analysis

- throughput, idle, and processing time per dataset

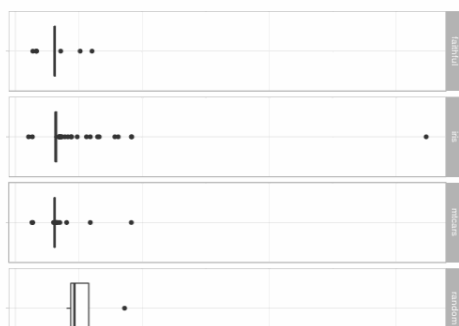
Throughput time per dataset



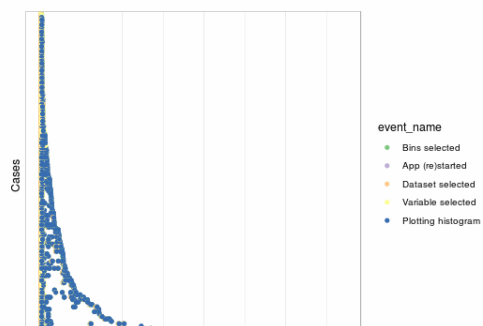
Idle time per dataset



Processing time per dataset

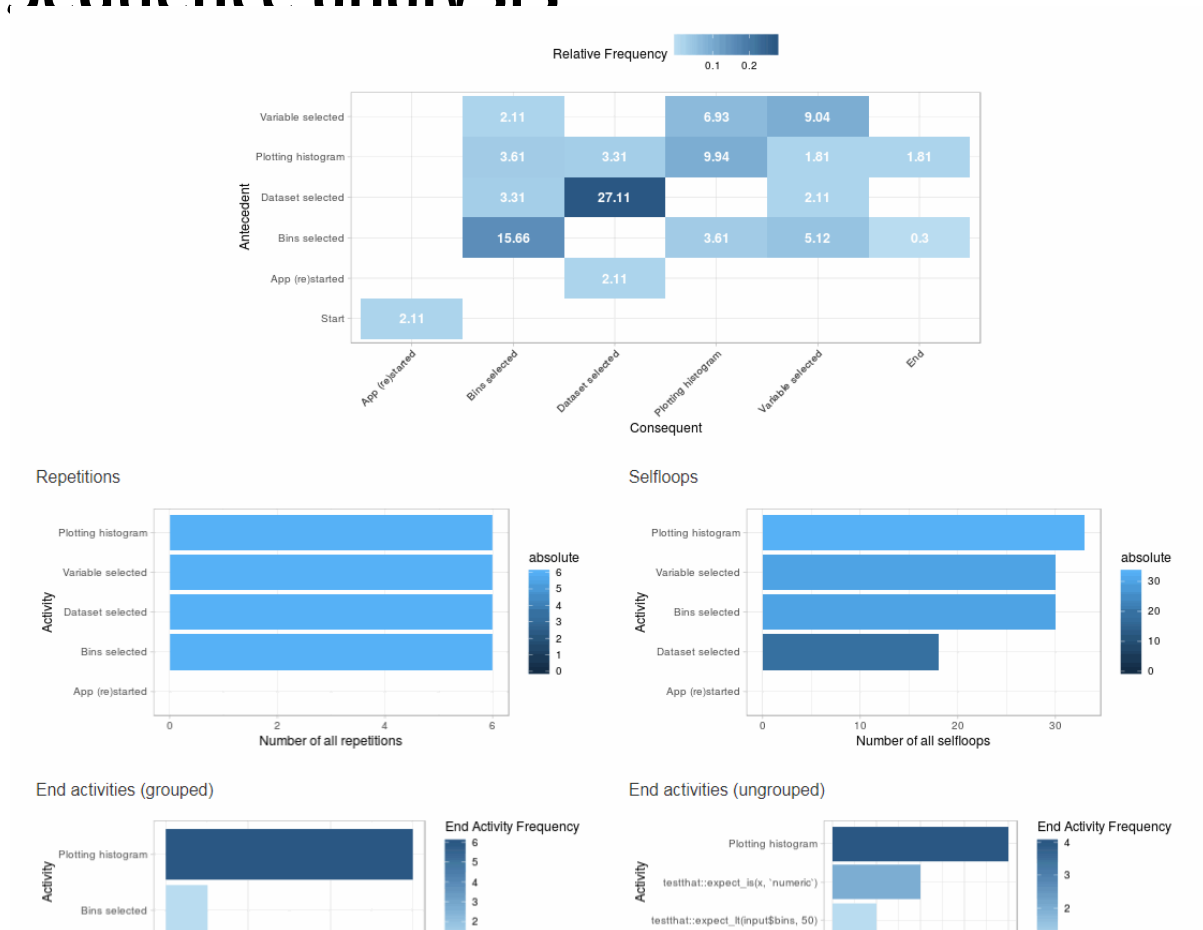


Dotted chart of cases



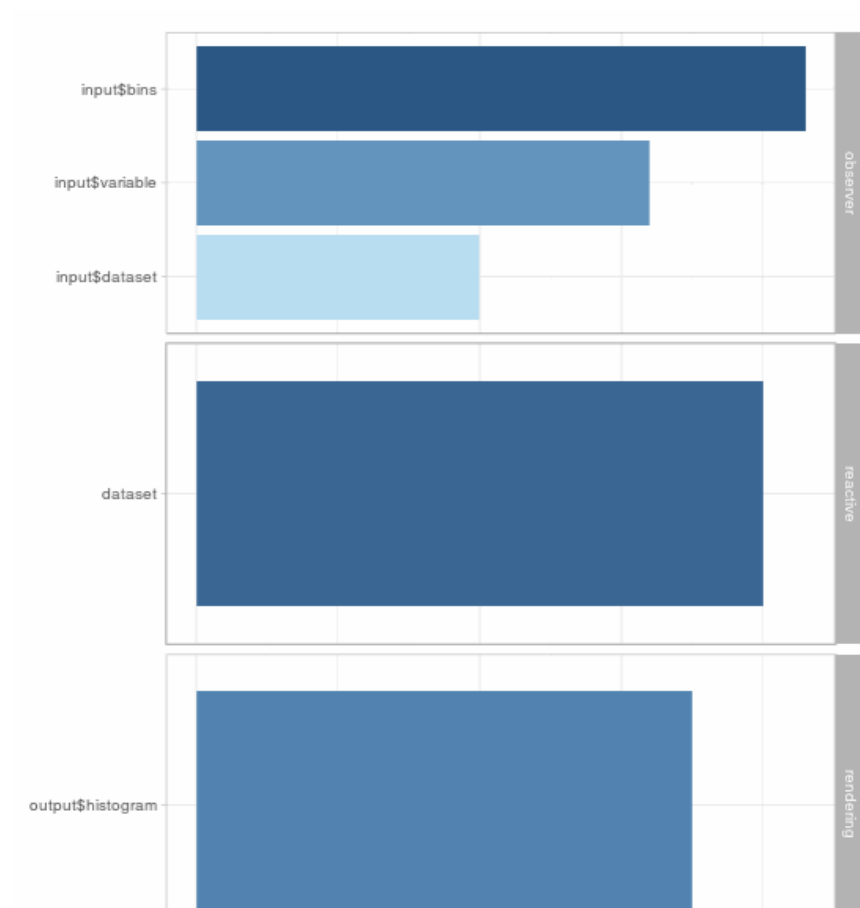
26 / 32

Sequence analysis



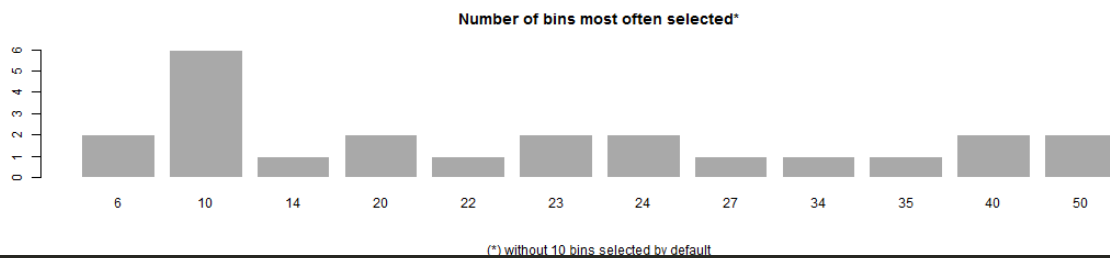
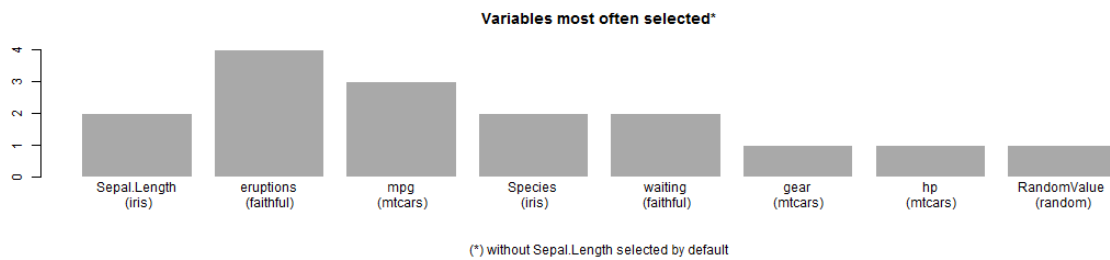
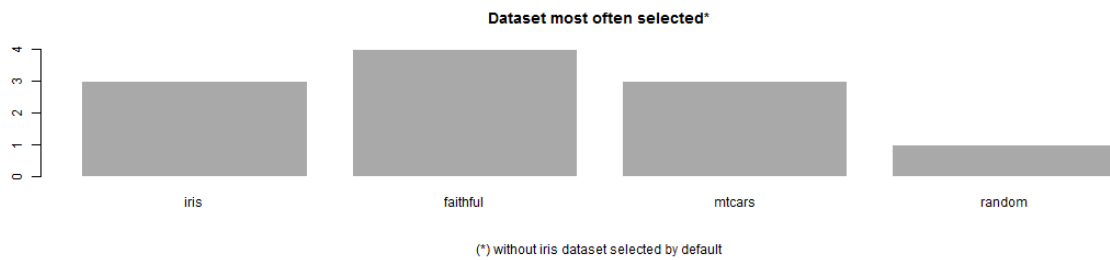
27 / 32

Resource analysis



28 / 32

Top users' actions



29 / 32

What's next?

30 / 32

Roadmap

- adding catching and logging **unexpected errors**
- adding **JSON** as default serialization format for storing events (also in a file)
- improving integration with **MongoDB**
- adding ability to **show or hide** selected part of event data
- adding a method for dynamically selecting which events should be logged **where and when**
 - permanently to a database **vs.** temporarily to a console
 - during developing & debugging **vs.** in production environment

31 / 32

Thank you!

- github.com/kalimu/shinyEventLogger - package repository
- kalimu.github.io/shinyEventLogger/ - package documentation

Contact:

- kalimu.github.io - personal homepage
- www.linkedin.com/in/kamilwais
- kamil.wais@gmail.com



32 / 32