Testing Shiny: Why, what, and how

2020-06 - 18 - ERUM

Colin Fay - ThinkR

$ whoami

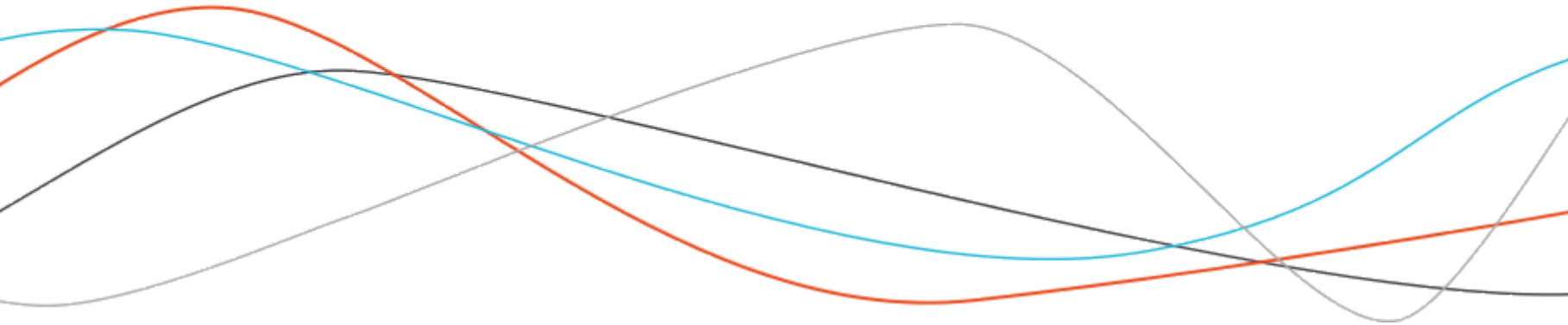
Colin FAY

Data Scientist & R-Hacker at ThinkR, a french company focused on Data Science & R.

Hyperactive open source developer.

- https://thinkr.fr
- https://rtask.thinkr.fr
- https://twitter.com/_colinfay
- https://github.com/colinfay
- https://colinfay.me

ThinkR

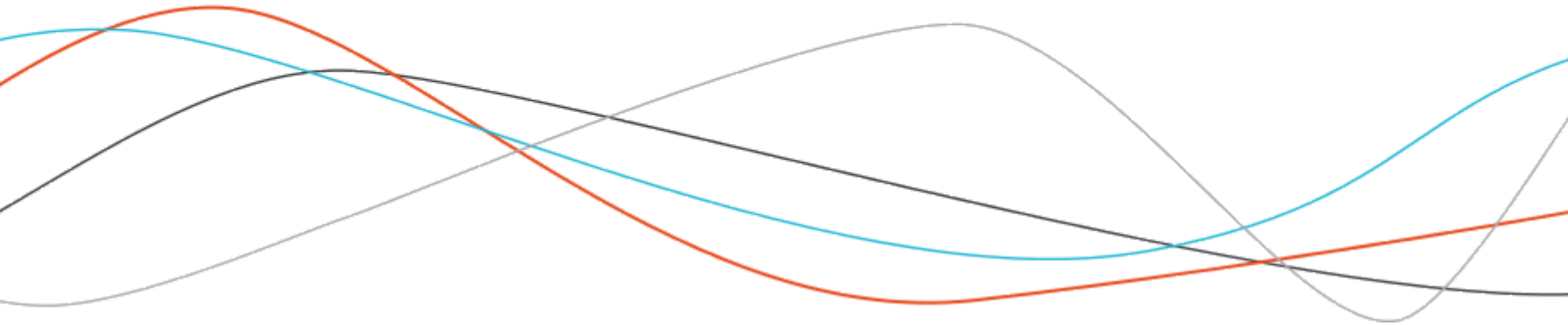


ThinkR

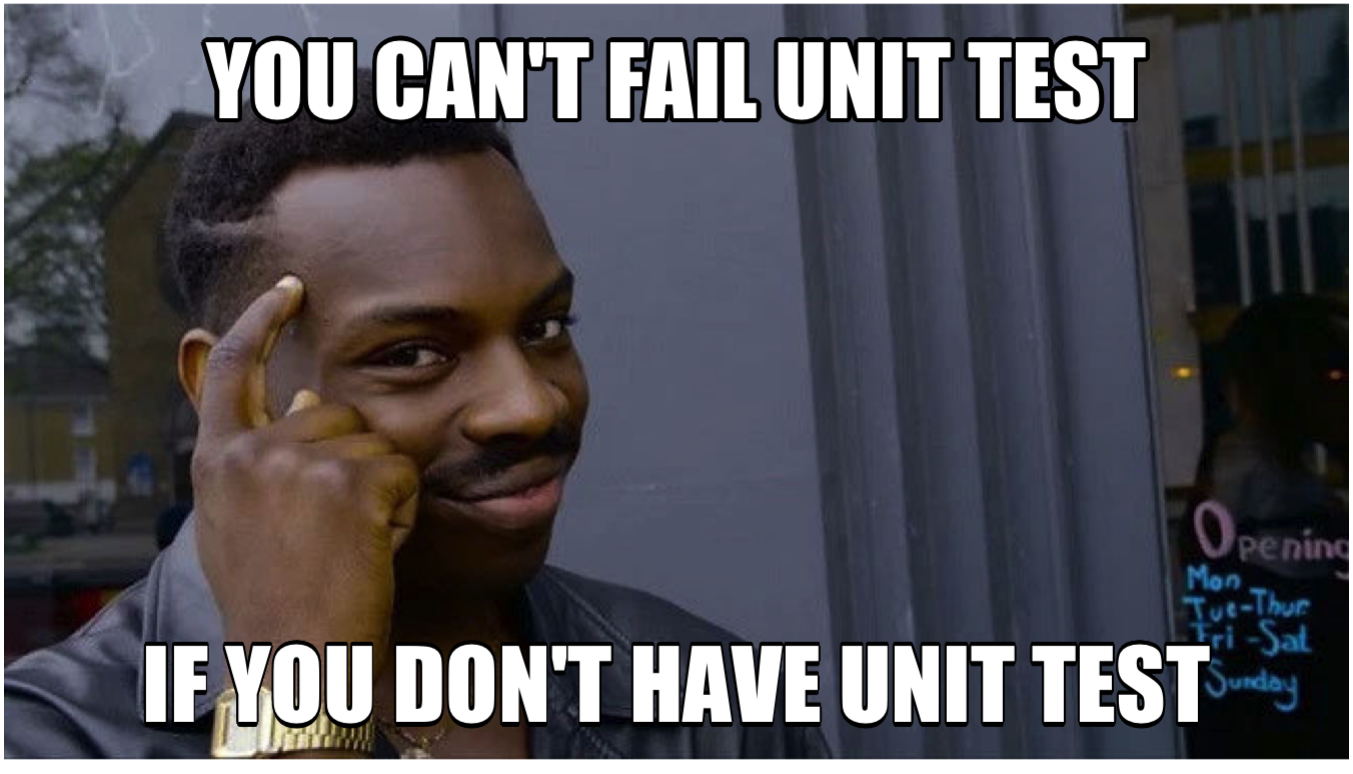
Data Science engineering, focused on R.

- Training
- Software Engineering
- R in production
- Consulting

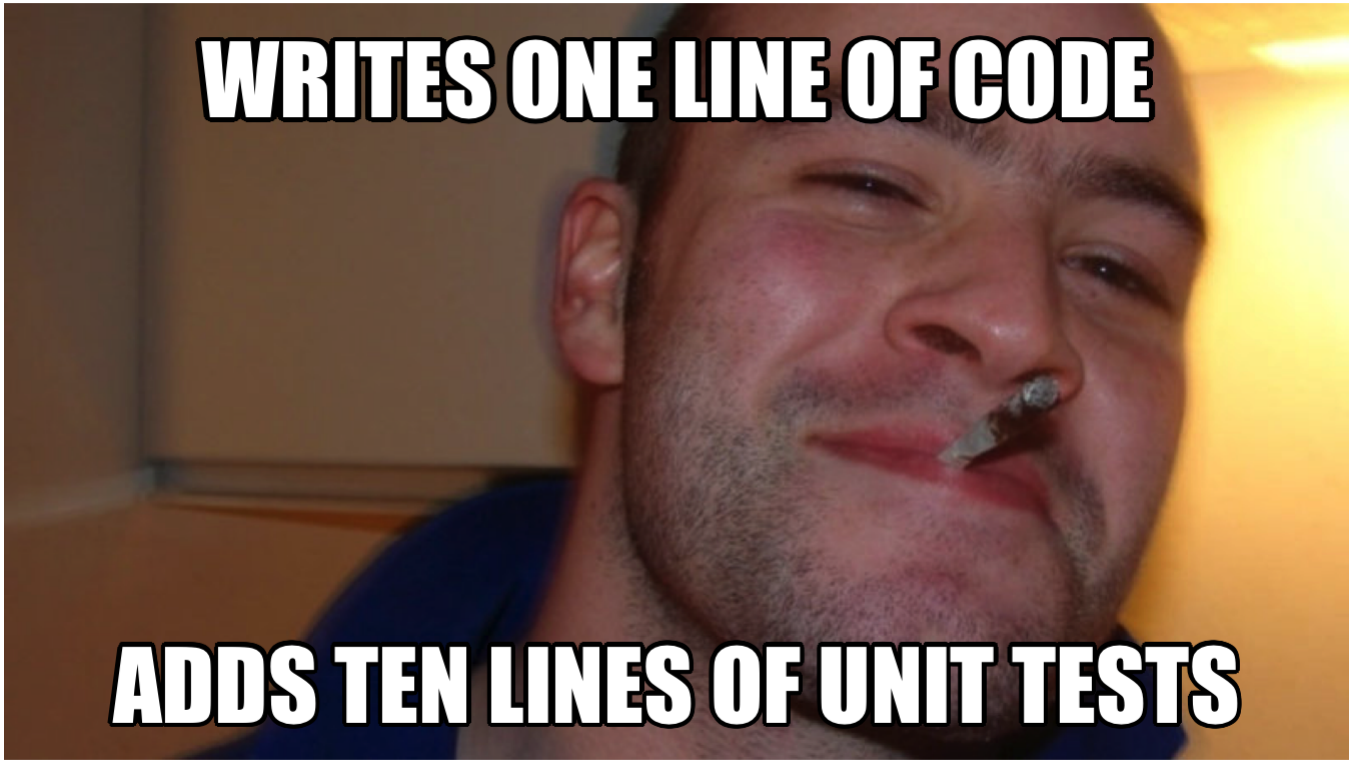
About Testing Applications



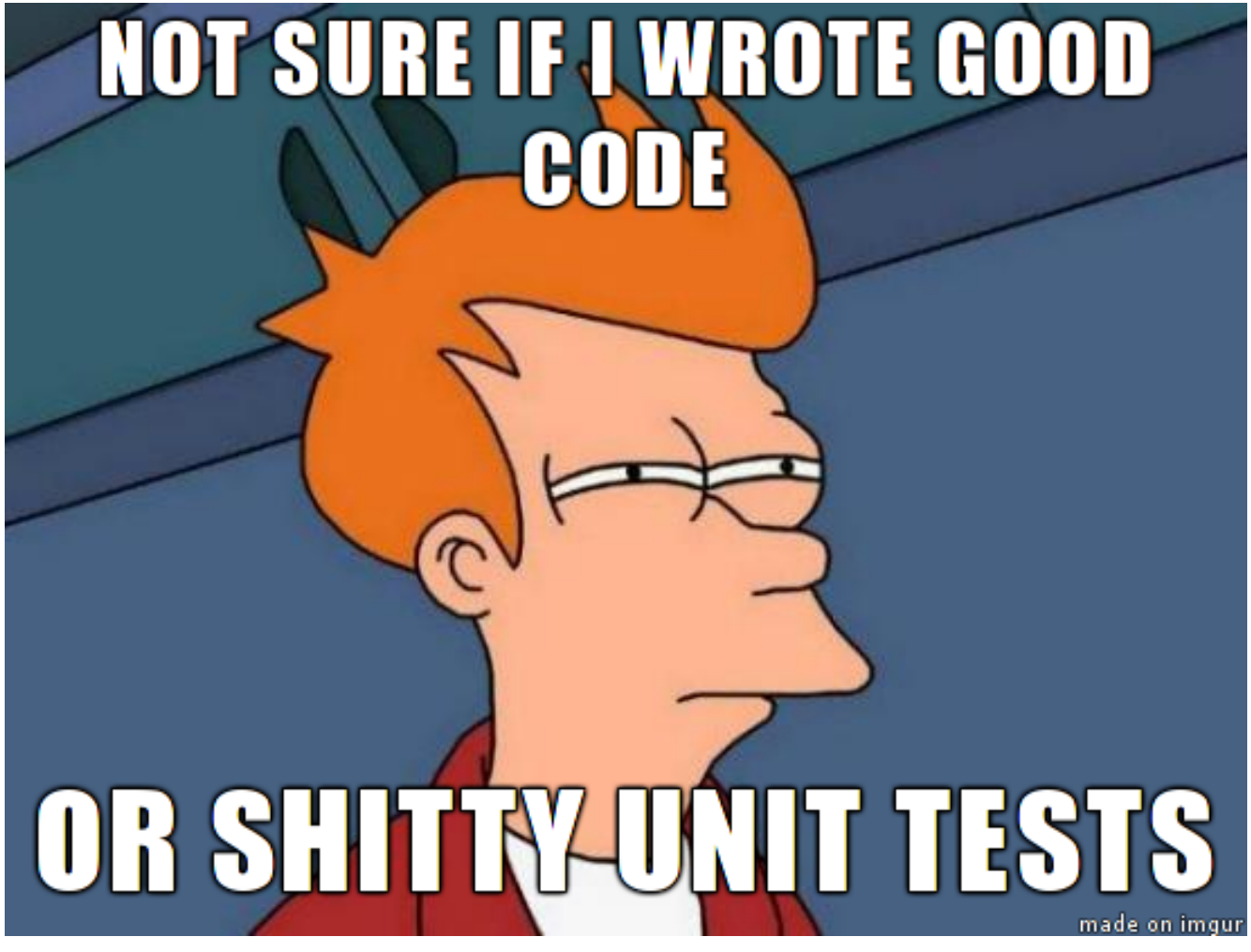
The two states of unit test



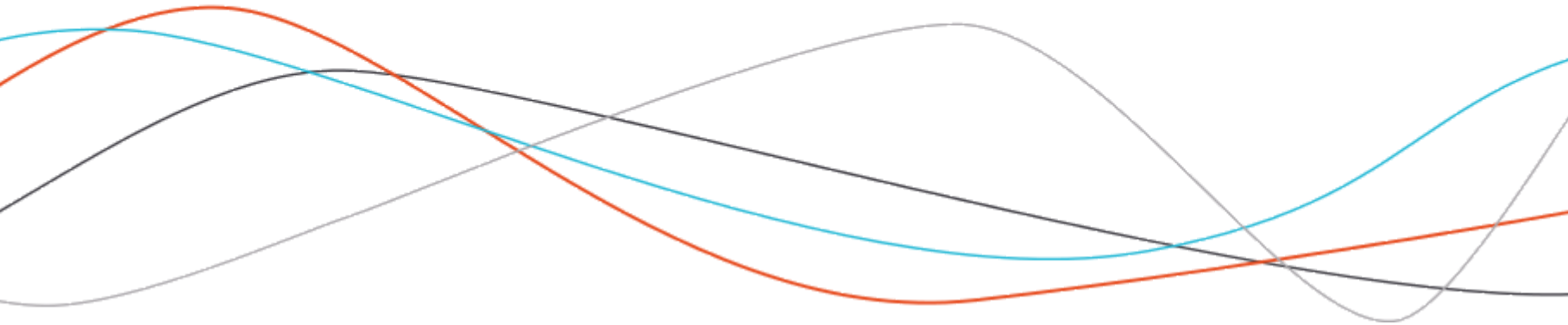
The two states of unit test



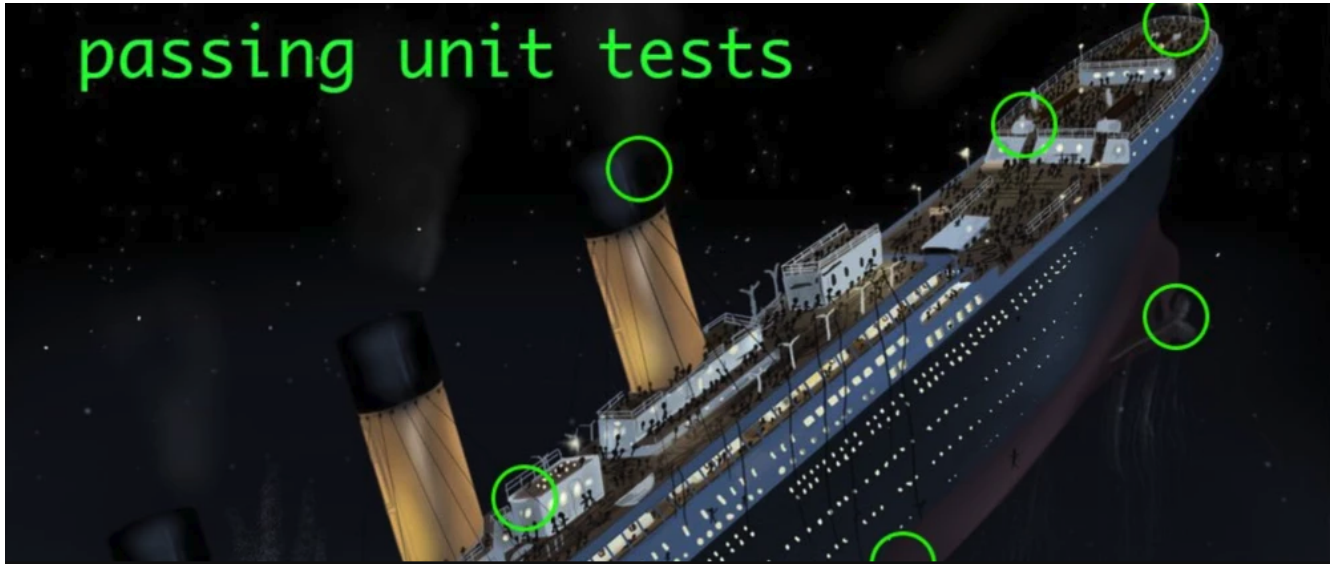
Or somewhere in the middle?



Testing Shiny: Why, What, and How



Why

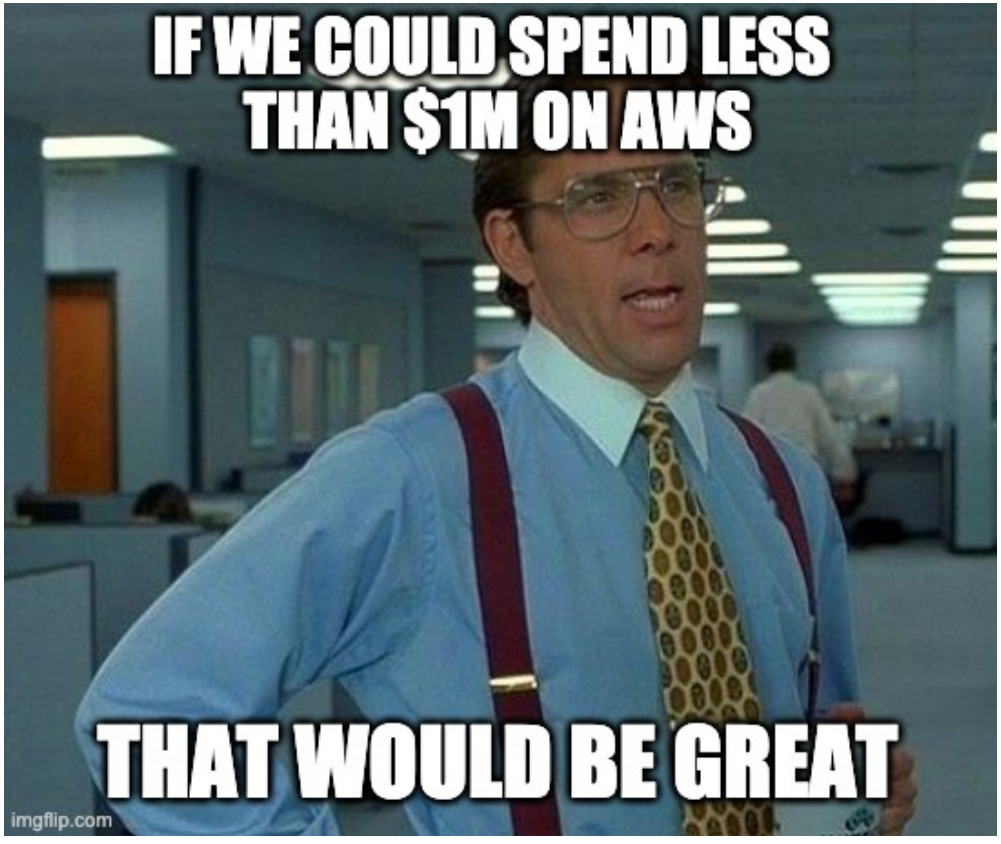


> Everything That's Not Tested Will Eventually Break

Why

> Don't let your users be your unit test

Why



> Control the application load
>
> See also: Don't DoS your own server

Why

- 🔒 Safely collaborate on a project

Why

- 🔒 Safely collaborate on a project
- 🧩 Safely change elements in the project

Why

- 🔒 Safely collaborate on a project
- 🧩 Safely change elements in the project
- 👁️ Safely serve application to the end users

Why

- 🔒 Safely collaborate on a project
- 🧩 Safely change elements in the project
- 👁️ Safely serve application to the end users
- 💵 Don't spend 1 million bucks on AWS (Jeff Bezos is rich enough)

Why

- 🔒 Safely collaborate on a project
- 🧩 Safely change elements in the project
- 👁️ Safely serve application to the end users
- 💵 Don't spend 1 million bucks on AWS (Jeff Bezos is rich enough)
- 🏋️ Don't clutter your own server

What

User Interface:

- What the end user sees
- What the end user interacts with

What

User Interface:

- What the end user sees
- What the end user interacts with

> Your time is limited, so if you have to choose don't focus too much on pure UI

What

User Interface:

- What the end user sees
- What the end user interacts with

> Your time is limited, so if you have to choose don't focus too much on pure UI

- Business logic

- Backend function

What

User Interface:

- What the end user sees
- What the end user interacts with

> Your time is limited, so if you have to choose don't focus too much on pure UI

- Business logic

- Backend function

- Application load

- How much RAM & CPU does my app need

How - Business logic/backend

Shiny App as a package 📦

-> Leverage standard testing frameworks

```
test_that("The meaning of life is
42", {
  expect_equal(
    meaning_of_life(),
    42
  )
})
```

How - User interface/frontend

`{shinytests}`

- 👓 Test visual regression of your application

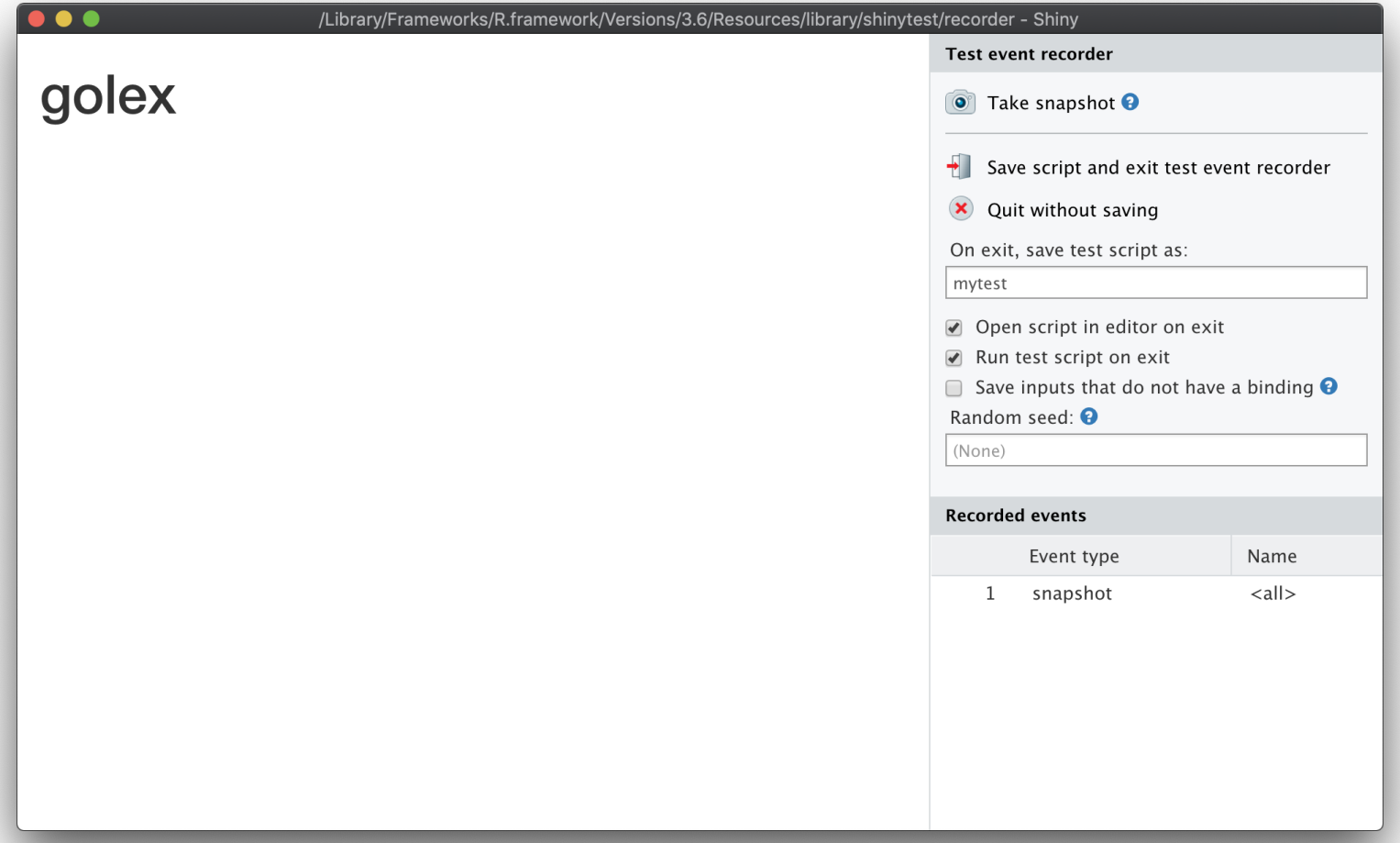
`puppeteer`

- 🎡 Command line tool to mock a web session, in NodeJS

`{crrry}`

- 🚗 R tool to drive a `{shiny}` session

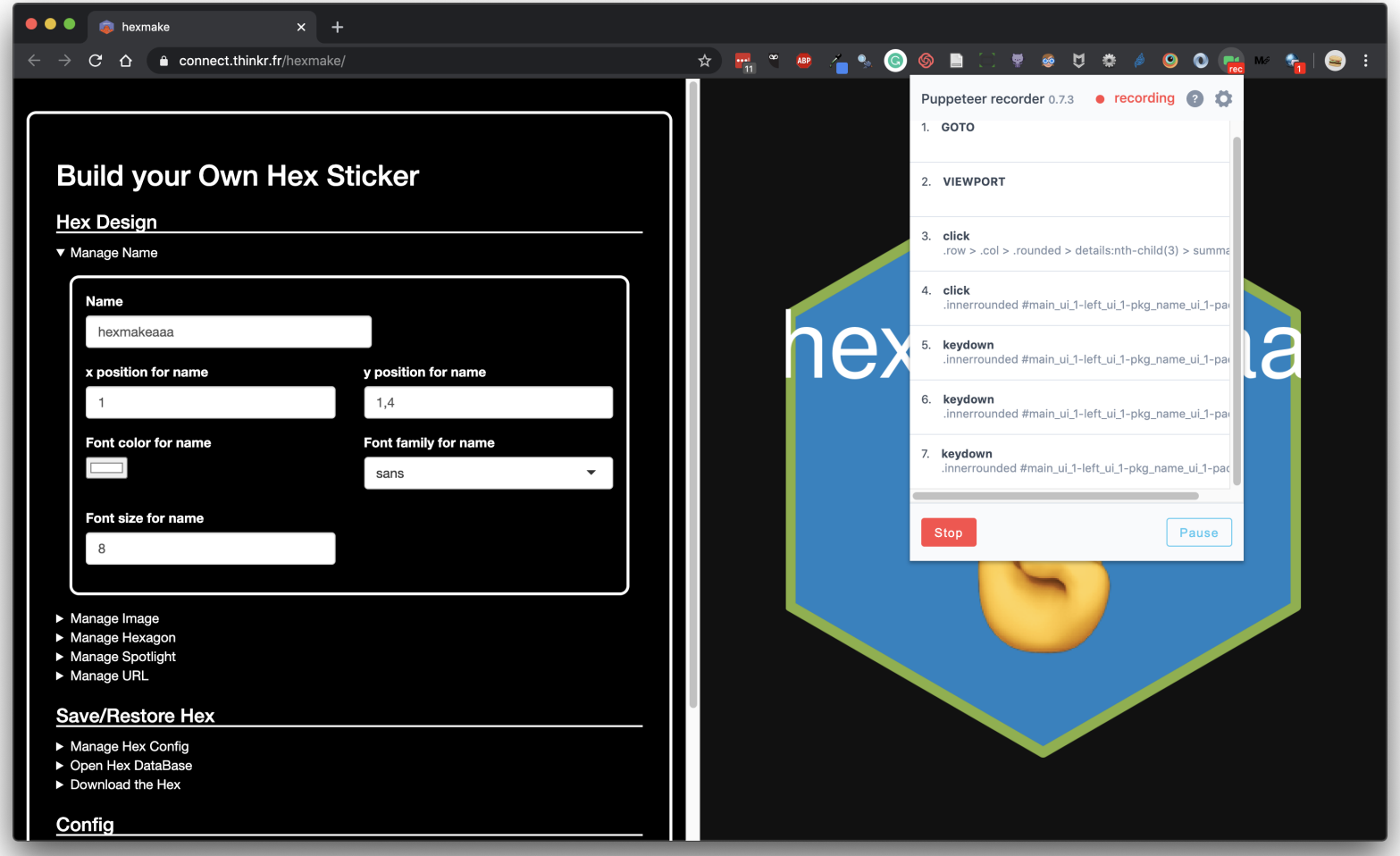
How - {shinytests}



How - puppeteer

```
const puppeteer = require('puppeteer');
(async () => {
  const browser = await puppeteer.launch()
  const page = await browser.newPage()

  await page.goto('http://localhost:2811/')
  await page.setViewport({ width: 1440, height: 766 })

  await page.waitForSelector('.row > .col > .rounded > details:nth-
child(3) > summary')
  await page.click('.row > .col > .rounded > details:nth-child(3) >
summary')

  await page.waitForSelector('.innerrounded #main_ui_1-left_ui_1-
pkg_name_ui_1-package')
  await page.click('.innerrounded #main_ui_1-left_ui_1-pkg_name_ui_1-
package')

  await browser.close()
})()
```

How - puppeteer



How - {crrry}

```
test <- crrry::CrrryOnPage$new(
  chrome_bin = pagedown::find_chrome(),
  chrome_port = httpuv::randomPort(),
  url = "https://connect.thinkr.fr/hexmake/",
  headless = TRUE
)
```

```
Running /usr/bin/google-chrome --no-first-run \
  --headless \
  '--user-data-dir=/home/runner/.local/share/r-crrri/chrome-data-dir-
gyjcgfuz' \
  '--remote-debugging-port=9598'
```

```
test$wait_for_shiny_ready()
```

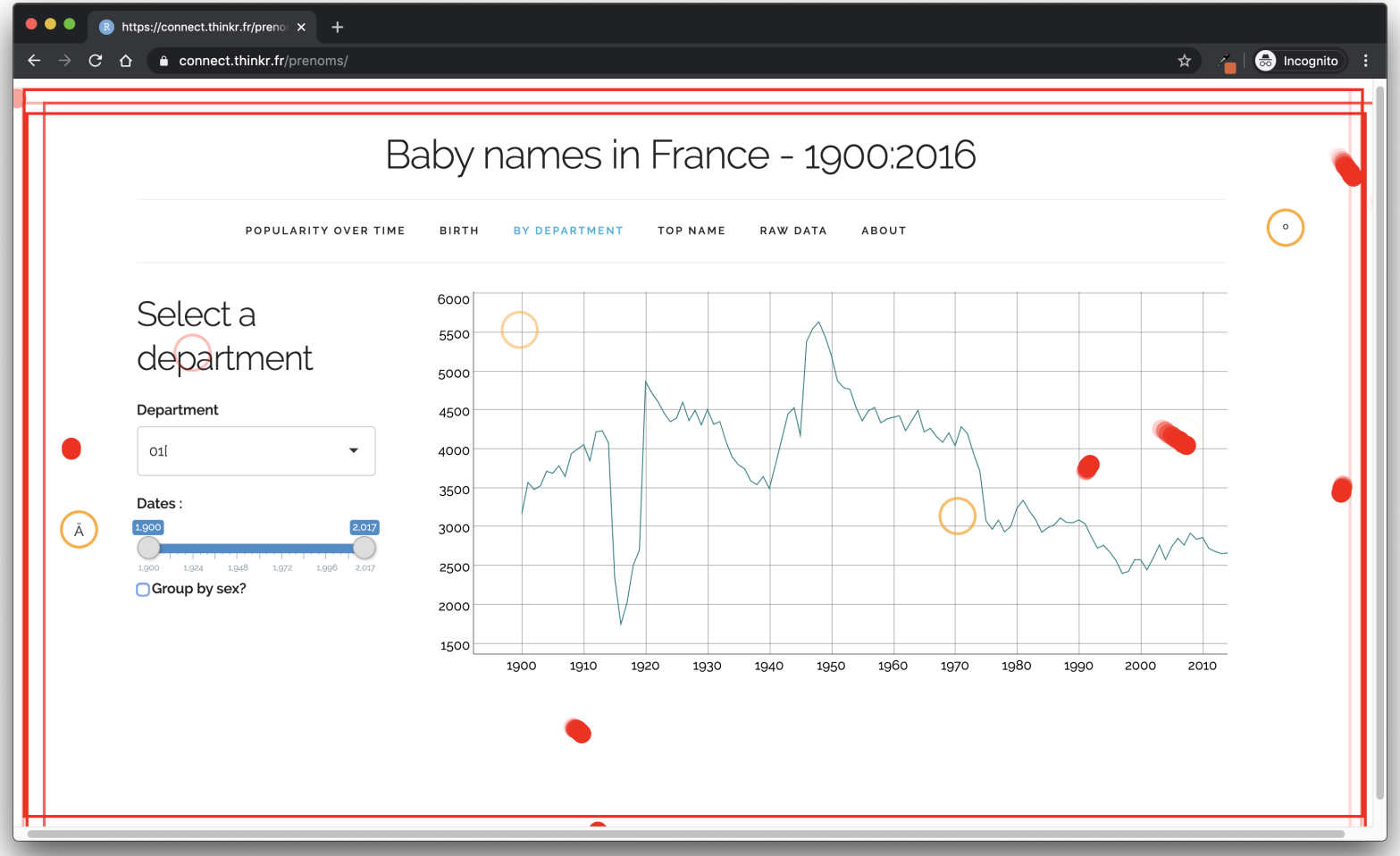
```
Shiny is computing
✔ Shiny is still running
```

How - {crrry}

```
for (i in letters[1:5]){
  test$shiny_set_input("main_ui_1-left_ui_1-pkg_name_ui_1-package", i)
}
```

```
── Setting id main_ui_1-left_ui_1-pkg_name_ui_1-package
Shiny is computing
✔ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_1-package
Shiny is computing
✔ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_1-package
Shiny is computing
✔ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_1-package
Shiny is computing
✔ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_1-package
Shiny is computing
✔ Shiny is still running
```

How - gremlins



How - gremlins

```
test <- crrry::CrrryOnPage$new(
  chrome_bin = pagedown::find_chrome(),
  chrome_port = httpuv::randomPort(),
  url = "https://connect.thinkr.fr/hexmake/",
  headless = TRUE
)
test$wait_for_shiny_ready()
test$gremlins_horde()
test$stop()
```

How - Testing the app load

`{shinyloadtest}`

- 🏋: native R package + Cli to record and replay load tests

`{dockerstats}`

- 🐋: get Docker stats inside R

`{crrry}` + `{dockerstats}`

- 👩‍❤️‍👨: replay session and watch the Docker stats

How - {shinyloadtest}

```
# Starting your app in another process
p <- processx::process$new(
  "Rscript",
  c(
    "-e",
    "options('shiny.port'= 2811);hexmake::run_app()"
  )
)
# Check that the process is alive
Sys.sleep(5) # We wait for the app to be ready
p$is_alive()
browseURL("http://localhost:2811")
```

How - {shinyloadtest}

```
fs::dir_create("shinylogs")
withr::with_dir(
  "shinylogs", {
    shinyloadtest::record_session(
      "http://localhost:2811",
      port = 1234
    )
  }
)
```

After recording:

```
shinycannon shinylogs/recording.log \
  http://localhost:2811 --workers 10 \
  --output-dir shinylogs/run1
```

How - {dockerstats}

```
system(
  "docker run --name hexmake --rm -p 2811:80 colinfay/hexmake",
  wait = FALSE
)
Sys.sleep(10)
```

```
dockerstats::dockerstats("hexmake")
```

```
  Container    Name           ID CPUPerc
1   hexmake hexmake 8f87dbe75312    0.08
  MemUsage MemLimit MemPerc NetI NetO BlockI
1   111MiB  7.78GiB    1.39 766B   0B     0B
  BlockO PIDs         record_time extra
1     0B    3 2020-06-18 13:36:18
```

How - {dockerstats}

```
test <- crrry::CrrryOnPage$new(
  chrome_bin = pagedown::find_chrome(),
  chrome_port = httpuv::randomPort(),
  url ="http://localhost:2811",
  headless = TRUE
)
```

```
Running \
  '/Applications/Google Chrome.app/Contents/MacOS/Google Chrome' \
  --no-first-run --headless \
  '--user-data-dir=/Users/colin/Library/Application Support/r-crrri/chrome-data-dir-
ieozazql' \
  '--remote-debugging-port=6976'
<simpleError in parse_block(g[-1], g[1], params.src): duplicate label 'setup'>
```

```
test$wait_for_shiny_ready()
```

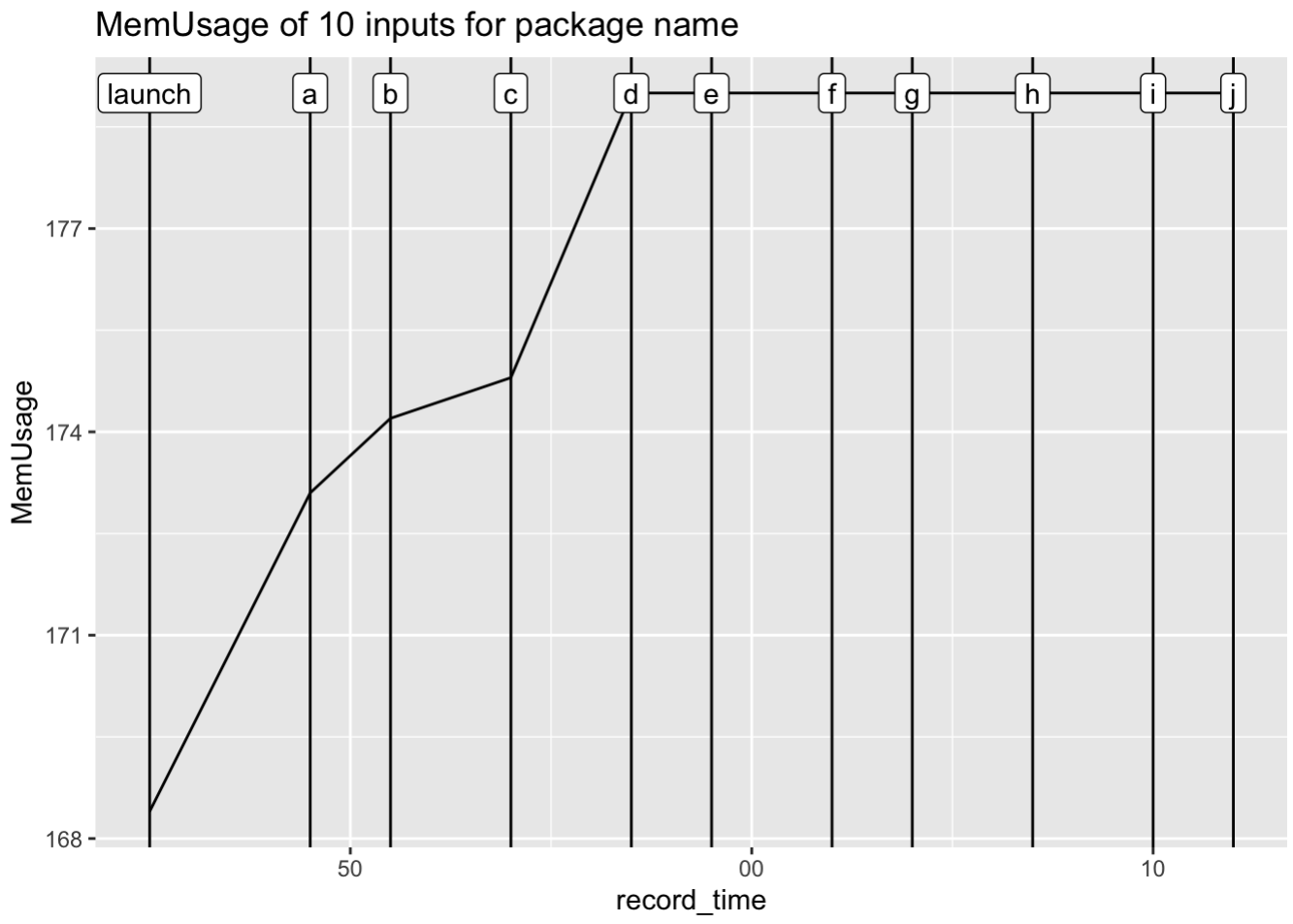
```
Shiny is computing
✓ Shiny is still running
```

How - {dockerstats}

```
results <- dockerstats::dockerstats("hexmake", extra = "launch")

for (i in letters[1:10]){
  test$shiny_set_input(
    "main_ui_1-left_ui_1-pkg_name_ui_1-package",
    i
  )
  results <- rbind(
    results,
    dockerstats::dockerstats("hexmake", extra = i)
  )
}
```

```
── Setting id main_ui_1-left_ui_1-pkg_name_ui_
Shiny is computing
✓ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_
Shiny is computing
✓ Shiny is still running
── Setting id main_ui_1-left_ui_1-pkg_name_ui_
Shiny is computing
```

How - {dockerstats}



engineering-shiny.org

Engineering Production-Grade Shiny Apps

Colin Fay, Sébastien Rochette, Vincent Guyader, Cervan Girard

2020-04-25

Introduction

This book is currently under development. It will be published in 2020 in the R Series by Chapman & Hall.

Motivation

This book will not **get you started with Shiny**, nor **talk about how to deploy into production and scale your app**. What we'll see is **the process of building the app**. Why? Lots of blog posts and books talk about starting to use `{shiny}` (**???**) or putting apps in production. Very few (if any) talk about this grey area between getting started and pushing into production.

So this is what this book is going to talk about: building Shiny application. We'll focus on the process, the workflow, and the tools we use at ThinkR when building big Shiny Apps.

Hence, if you are starting to read this book, we assume you have a working knowledge of how to build a small application, and want to know how to go one step further.

- testthat.r-lib.org
- rstudio.github.io/shinytest
- pptr.dev
- github.com/ColinFay/crrry
- github.com/marmelab/gremlins.js
- rstudio.github.io/shinyloadtest
- github.com/ColinFay/dockerstats

Thx! Questions?

Colin Fay

Online

- colin@thinkr.fr
- http://twitter.com/_colinfay
- http://twitter.com/thinkr_fr
- https://github.com/ColinFay
- https://thinkr.fr/
- https://rtask.thinkr.fr/
- https://colinfay.me/

Related projects

- engineering-shiny.org
- {golem}
- {shinipsum}
- {fakir}
- {shinysnippets}