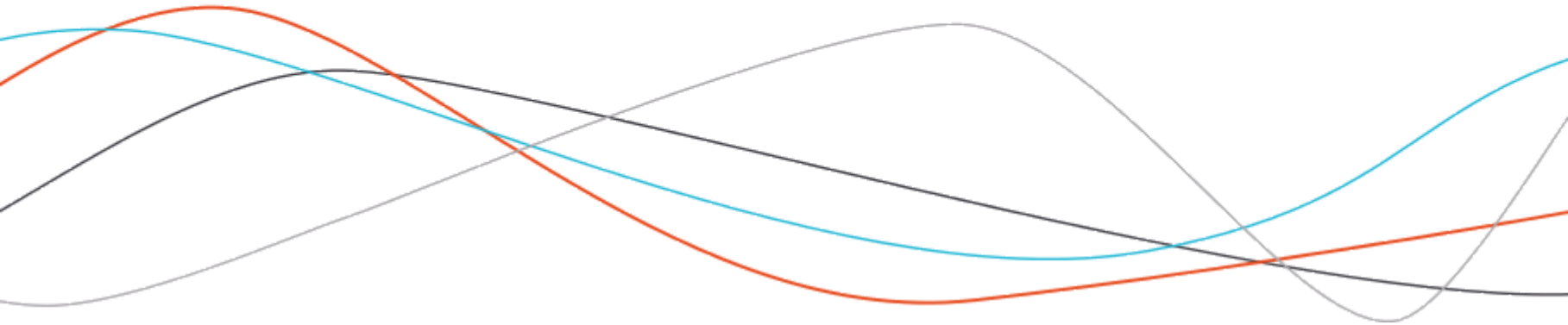
THINKR

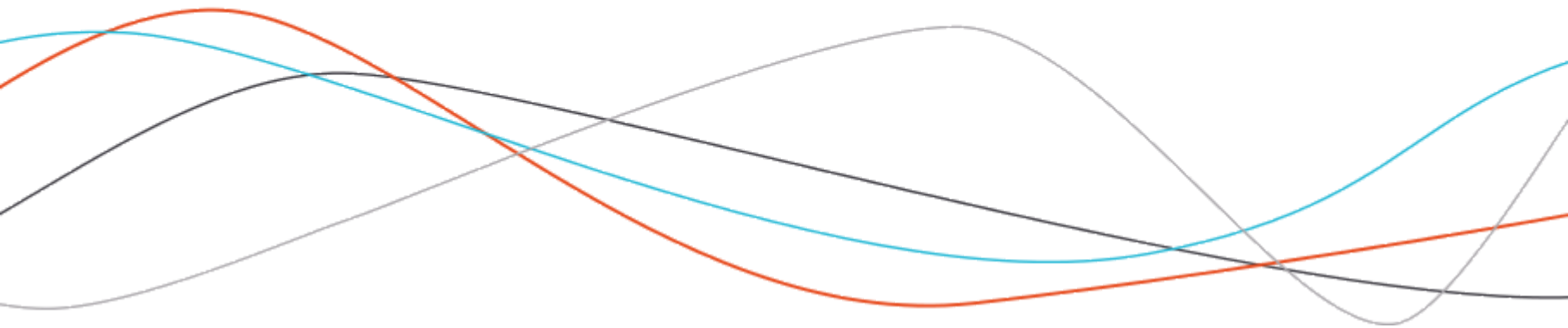
Building Successful Shiny Apps with {golem}

Colin Fay - ThinkR

PART 04 TEST AND SEND



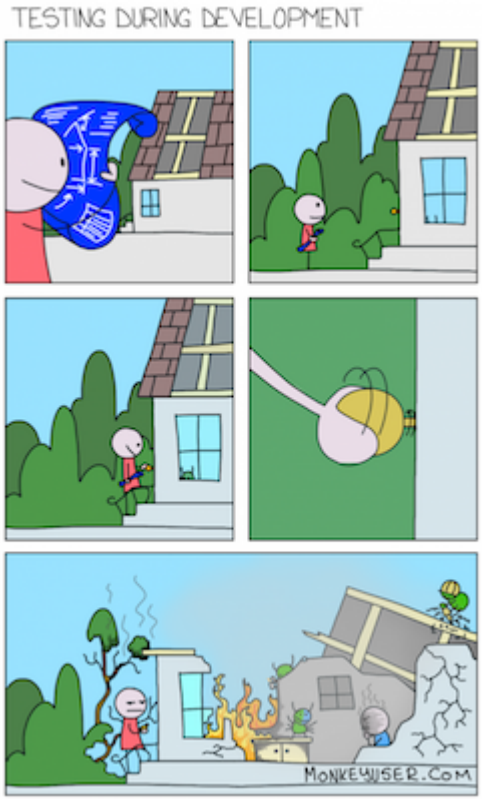
Everything which is not tested will, at last, break.



Why automate code testing?

- To save time!
- Work with others
- Transfer the project
- Long term stability

Write tests before it's too late



Use test in a "defensive programming" approach to prevent bugs.

Don't trust yourself in 6 months.

Be sure to send in production a package with minimum bugs.

Good news: **you're already writing test**, you just didn't know that before.

What do we test

- Focus on business logic testing: it's more important to test that the algorithm is still accurate than testing the UI
- As we've separated businness logic from application logic, we use package development testing tools
- We'll use the `{testthat}` 📦

01_start.R

```
## Init Testing Infrastructure ----
## Create a template for tests
golem::use_recommended_tests()
```

> Recommended tests

02_dev.R

```
## Tests ----
## Add one line by test you want to create
usethis::use_test( "app" )
```

> Add custom test

Writing custom tests

```
test_that( "details series 1", {
  test1a
  test1b
  }
)

test_that("details series 2", {
  test2a
  test2b
  }
)
```

Test functions

- Start with `expect_*`
- Take two elements: Actual result & the expected result.
- If the test is not passed, the function returns an error. If the test passes, the function returns nothing.

```
library(testthat, warn.conflicts = FALSE)

expect_equal(10, 10)

a <- sample(1:10, 1)
b <- sample(1:10, 1)
expect_equal(a+b, 200)
```

```
Error: a + b not equal to 200.
1/1 mismatches
[1] 12 - 200 == -188
```

{testthat} expectations

```
library(testthat)
grep("^expect", ls("package:testthat"), value = TRUE)
```

```
 [1] "expect"
 [2] "expect_condition"
 [3] "expect_cpp_tests_pass"
 [4] "expect_equal"
 [5] "expect_equal_to_reference"
 [6] "expect_equivalent"
 [7] "expect_error"
 [8] "expect_failure"
 [9] "expect_false"
[10] "expect_gt"
[11] "expect_gte"
[12] "expect_identical"
[13] "expect_invisible"
[14] "expect_is"
[15] "expect_known_failure"
[16] "expect_known_hash"
[17] "expect_known_output"
[18] "expect_known_value"
[19] "expect_length"
[20] "expect_less_than"
```

{golem} test helpers

- `golem::expect_html_equal()`
- `golem::expect_shinytag()`
- `golem::expect_shinytaglist()`

Launch tests

```
devtools::test()
```

> Launches only the tests infrastructure

```
devtools::check()
```

> Launches the tests + a series of default tests

R CMD check

To test the code more globally, in the command line (i.e. in the terminal): R CMD check.

Or simply the `devtools::check()` function in your R session.

More tests are performed with `check` than with `devtools::test()`, which "only" performs the tests in the test folder.

This command runs around 50 different tests.

Is performed when you click the "Check" button on the Build tab of RStudio.

Test with rhub

`{rhub}` is a package that allows you to test for several OS:

```
library(rhub)
ls("package:rhub")
```

```
 [1] "check"
 [2] "check_for_cran"
 [3] "check_on_centos"
 [4] "check_on_debian"
 [5] "check_on_fedora"
 [6] "check_on_linux"
 [7] "check_on_macos"
 [8] "check_on_solaris"
 [9] "check_on_ubuntu"
[10] "check_on_windows"
[11] "check_with_rdevel"
[12] "check_with_roldrel"
[13] "check_with_rpatched"
[14] "check_with_rrelease"
[15] "check_with_sanitizers"
[16] "check_with_valgrind"
[17] "get_check"
```

Test with rhub

```r
remotes::install_github("r-hub/rhub")
# or
install.packages("rhub")
# verify your email
library(rhub)
validate_email()
```

```
— Choose email address to (re)validate (or 0 to exit)

1:    colin@thinkr.fr
2:    New email address

Selection: 1
Please check your emails for the r-hub access token.
Token:
```

Test with rhub

```
rhub::check()
```

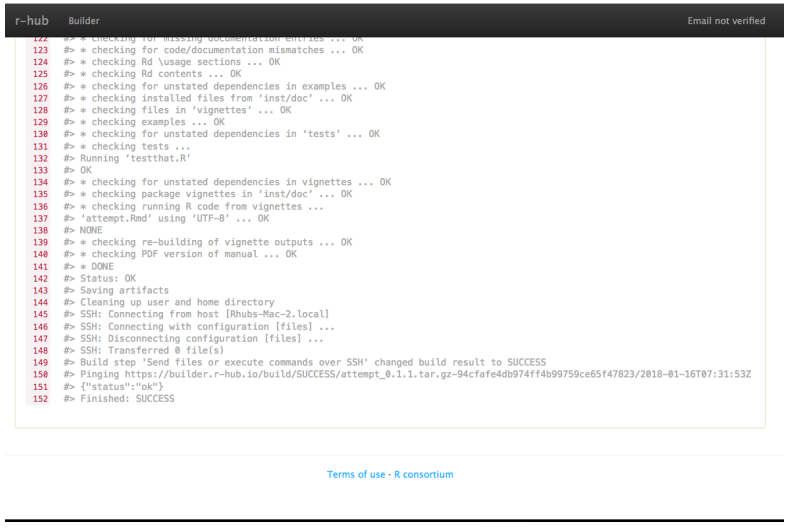
Which platforms are supported?

```
rhub::platforms()
```

```
debian-clang-devel:
  Debian Linux, R-devel, clang, ISO-8859-15 locale
debian-gcc-devel:
  Debian Linux, R-devel, GCC
debian-gcc-devel-nold:
  Debian Linux, R-devel, GCC, no long double
debian-gcc-patched:
  Debian Linux, R-patched, GCC
debian-gcc-release:
  Debian Linux, R-release, GCC
fedora-clang-devel:
  Fedora Linux, R-devel, clang, gfortran
fedora-gcc-devel:
  Fedora Linux, R-devel, GCC
linux-x86_64-centos6-epel:
  CentOS 6, stock R from EPEL
```

Test with rhub

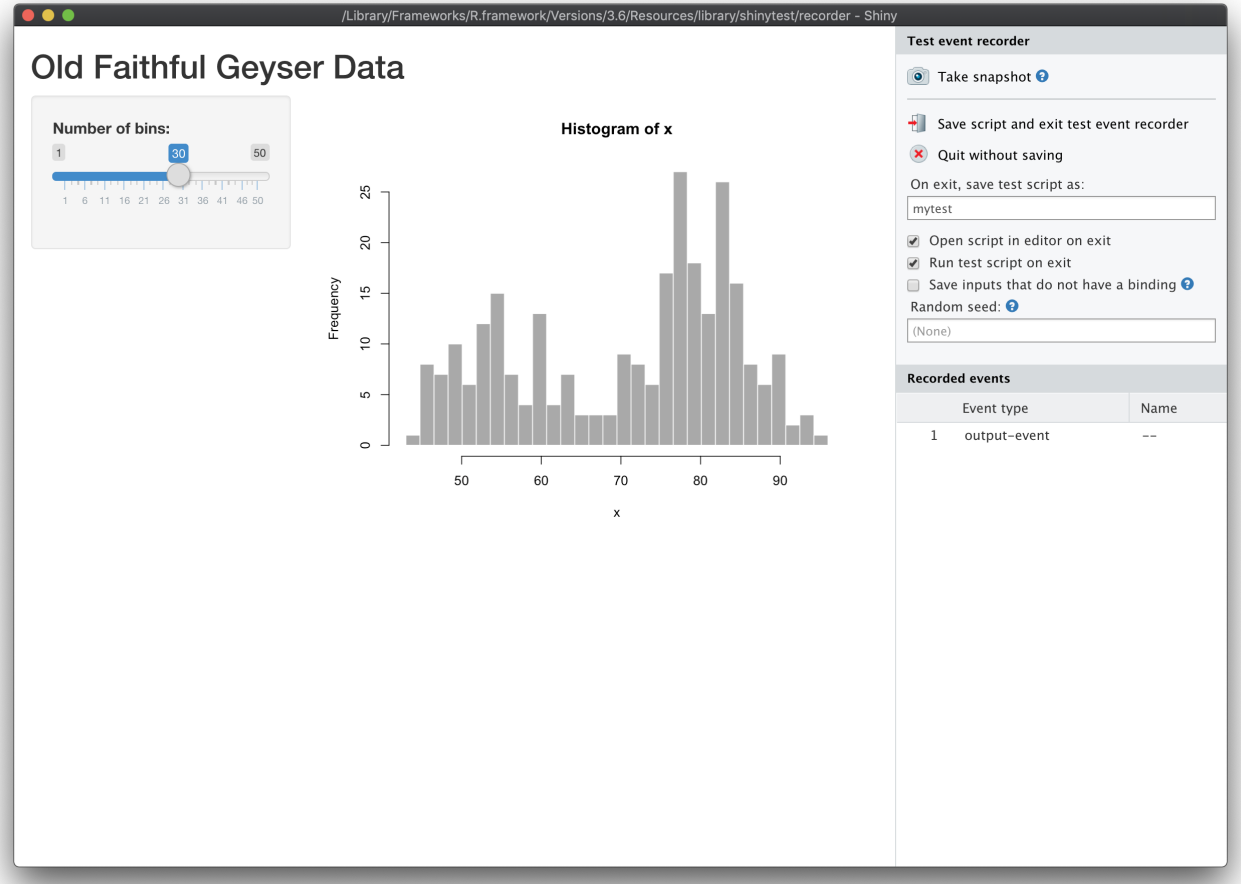
```
> rhub::check()

── Choose build platform

 1: Debian Linux, R-devel, GCC (debian-gcc-devel)
 2: Debian Linux, R-patched, GCC (debian-gcc-patched)
 3: Debian Linux, R-release, GCC (debian-gcc-release)
 4: Fedora Linux, R-devel, clang, gfortran (fedora-clang-devel)
 5: Fedora Linux, R-devel, GCC (fedora-gcc-devel)
 6: CentOS 6, stock R from EPEL (linux-x86_64-centos6-epel)
 7: CentOS 6 with Redhat Developer Toolset, R from EPEL (linux-x86_64-centos6-epel-rdt)
 8: Debian Linux, R-devel, GCC ASAN/UBSAN (linux-x86_64-rocker-gcc-san)
 9: macOS 10.11 El Capitan, R-release (experimental) (macos-elcapitan-release)
10: macOS 10.9 Mavericks, R-oldrel (experimental) (macos-mavericks-oldrel)
11: Oracle Solaris 10, x86, 32 bit, R-patched (experimental) (solaris-x86-patched)
12: Ubuntu Linux 16.04 LTS, R-devel, GCC (ubuntu-gcc-devel)
13: Ubuntu Linux 16.04 LTS, R-release, GCC (ubuntu-gcc-release)
14: Ubuntu Linux 16.04 LTS, R-devel with rchk (ubuntu-rchk)
15: Windows Server 2008 R2 SP1, R-devel, 32/64 bit (windows-x86_64-devel)
16: Windows Server 2008 R2 SP1, R-oldrel, 32/64 bit (windows-x86_64-oldrel)
17: Windows Server 2008 R2 SP1, R-patched, 32/64 bit (windows-x86_64-patched)
18: Windows Server 2008 R2 SP1, R-release, 32/64 bit (windows-x86_64-release)

Selection: 19
Enter an item from the menu, or 0 to exit
Selection: 10

-  Building package
-  Uploading package
-  Preparing build, see status at
   http://builder.r-hub.io/status/attempt_0.1.1.tar.gz-94cfafe4db974ff4b99759ce65f47823
-  Build started

>
```



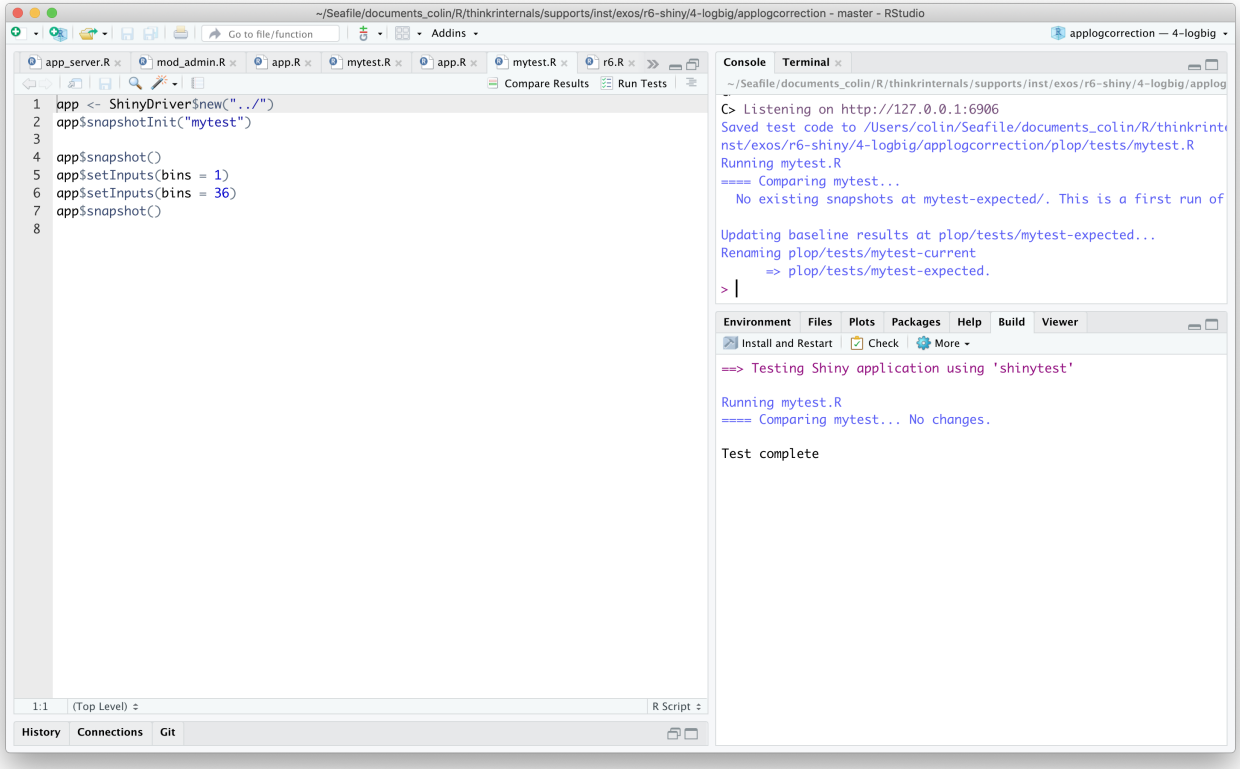
{shinytest}

```
golem::add_rstudioconnect_file()
shinytest::recordTest(".")
```

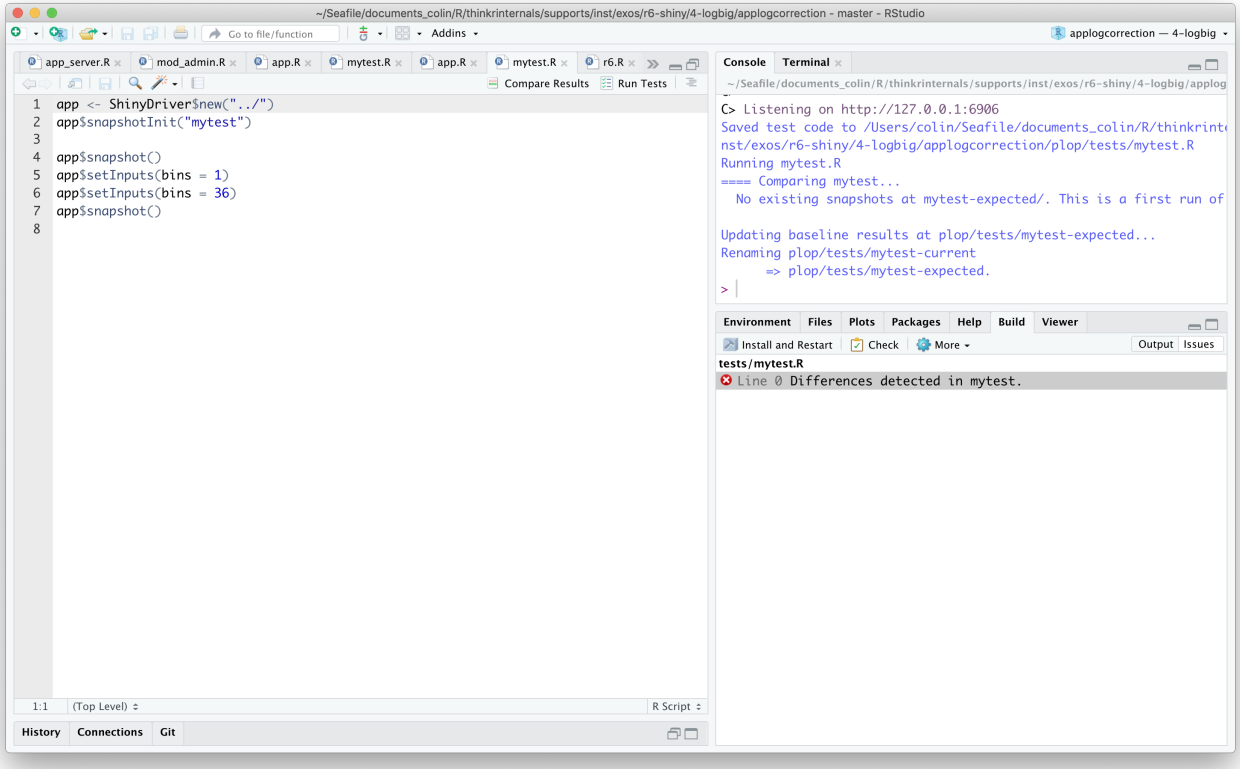
{shinytest}


/Library/Frameworks/R.framework/Versions/3.6/Resources/library/shinytest/recorder - Shiny
Old Faithful Geyser Data
Number of bins:
1
30
50
1 6 11 16 21 26 31 36 41 46 50
Histogram of x
Frequency
0 5 10 15 20 25
50 60 70 80 90
x
Test event recorder
Take snapshot
Save script and exit test event recorder
Quit without saving
On exit, save test script as:
mytest
Open script in editor on exit
Run test script on exit
Save inputs that do not have a binding
Random seed:
(None)
Recorded events
Event type
Name
1 output-event --

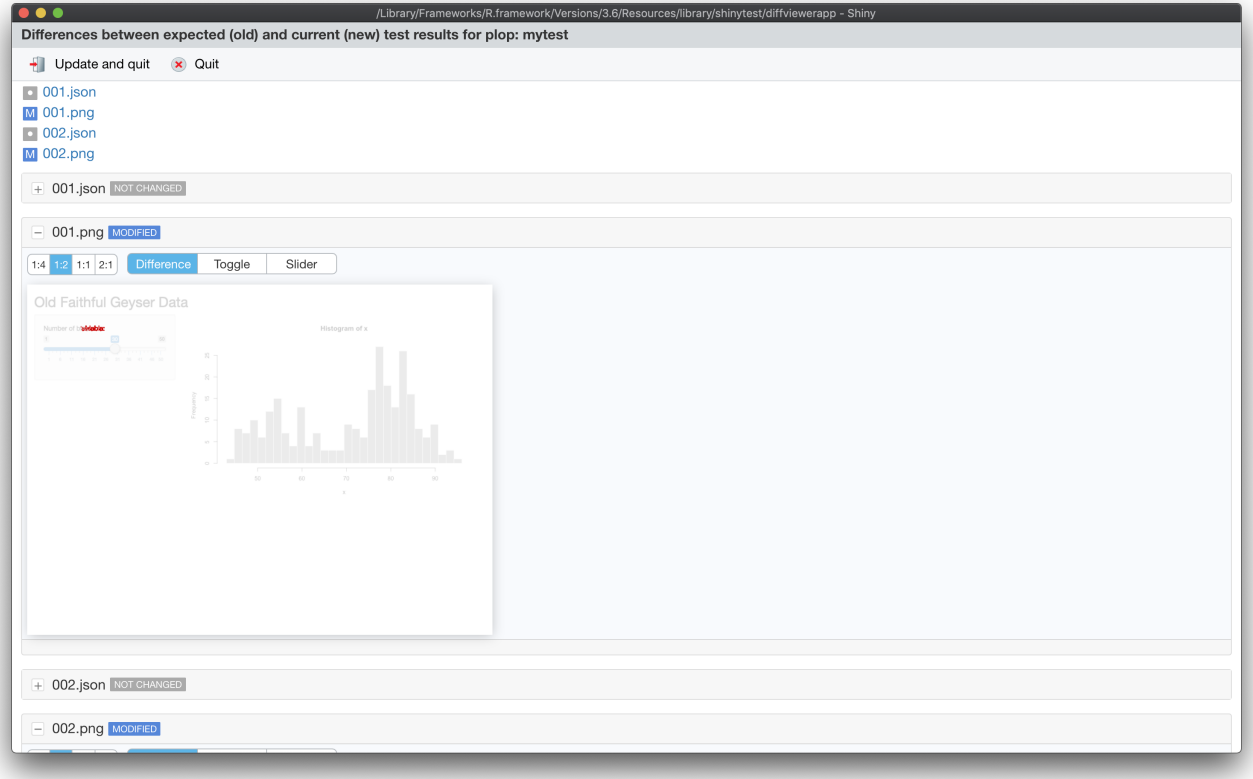
{shinytest}


~/Seafile/documents_colin/R/thinkrinternals/supports/inst/exos/r6-shiny/4-logbig/applogcorrection - master - RStudio
app_server.R
mod_admin.R
app.R
mytest.R
app.R
mytest.R
r6.R
Compare Results
Run Tests
1 app <- ShinyDriver$new("../")
2 app$snapshotInit("mytest")
3
4 app$snapshot()
5 app$setInputs(bins = 1)
6 app$setInputs(bins = 36)
7 app$snapshot()
8
1:1
(Top Level)
R Script
History
Connections
Git
Console
Terminal
C> Listening on http://127.0.0.1:6906
Saved test code to /Users/colin/Seafile/documents_colin/R/thinkrinternals/supports/inst/exos/r6-shiny/4-logbig/applogcorrection/plop/tests/mytest.R
Running mytest.R
==== Comparing mytest...
No existing snapshots at mytest-expected/. This is a first run of
Updating baseline results at plop/tests/mytest-expected...
Renaming plop/tests/mytest-current
=> plop/tests/mytest-expected.
>
Environment
Files
Plots
Packages
Help
Build
Viewer
Install and Restart
Check
More
==> Testing Shiny application using 'shinytest'
Running mytest.R
==== Comparing mytest... No changes.
Test complete

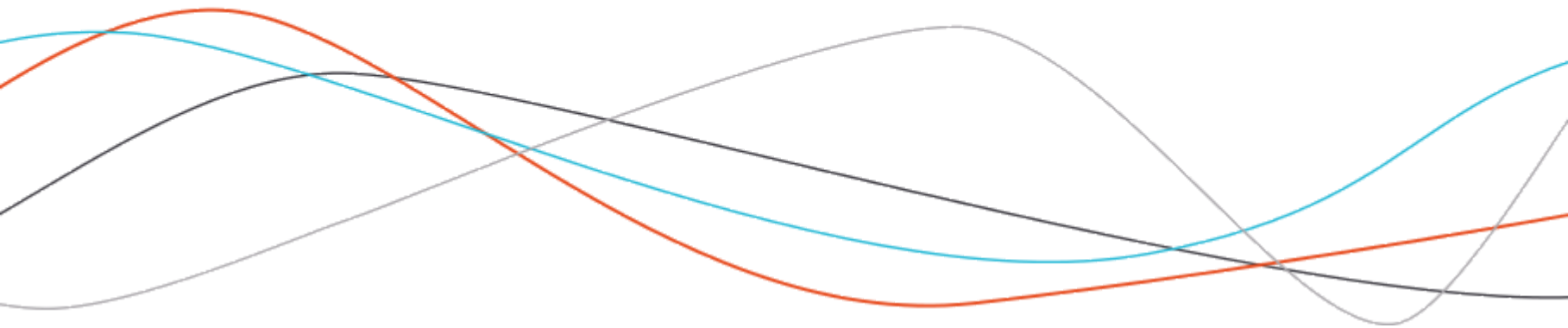
{shinytest}


~/Seafile/documents_colin/R/thinkrinternals/supports/inst/exos/r6-shiny/4-logbig/applogcorrection - master - RStudio
app_server.R
mod_admin.R
app.R
mytest.R
app.R
mytest.R
r6.R
Compare Results
Run Tests
1 app <- ShinyDriver$new("../")
2 app$snapshotInit("mytest")
3
4 app$snapshot()
5 app$setInputs(bins = 1)
6 app$setInputs(bins = 36)
7 app$snapshot()
8
1:1
(Top Level)
R Script
History
Connections
Git
Console
Terminal
C> Listening on http://127.0.0.1:6906
Saved test code to /Users/colin/Seafile/documents_colin/R/thinkrint
nst/exos/r6-shiny/4-logbig/applogcorrection/plop/tests/mytest.R
Running mytest.R
==== Comparing mytest...
No existing snapshots at mytest-expected/. This is a first run of
Updating baseline results at plop/tests/mytest-expected...
Renaming plop/tests/mytest-current
=> plop/tests/mytest-expected.
Environment
Files
Plots
Packages
Help
Build
Viewer
Install and Restart
Check
More
Output
Issues
tests/mytest.R
Line 0 Differences detected in mytest.

{shinytest}


/Library/Frameworks/R.framework/Versions/3.6/Resources/library/shinytest/diffviewerapp - Shiny
Differences between expected (old) and current (new) test results for plop: mytest
Update and quit
Quit
001.json
001.png
002.json
002.png
001.json NOT CHANGED
001.png MODIFIED
1:4 1:2 1:1 2:1
Difference
Toggle
Slider
Old Faithful Geyser Data
002.json NOT CHANGED
002.png MODIFIED

SEND TO PRODUCTION



Send to prod

- Once everything is tested, you can send to prod using `{golem}`
- Deploy on a server, or build as a `tar.gz` with `pkgbuild::build()`

```
## RStudio ----
## If you want to deploy on RStudio related platforms
golem::add_rstudioconnect_file()
golem::add_shinyappsio_file()
golem::add_shinyserver_file()
```

```
## Docker ----
## If you want to deploy via a generic Dockerfile
golem::add_dockerfile()
## If you want to deploy to ShinyProxy
golem::add_dockerfile_shinyproxy()
## If you want to deploy to Heroku
golem::add_dockerfile_heroku()
```

Demo time 🎉

Your turn!

https://github.com/ColinFay/golem-joburg/tree/master/exo-part-4