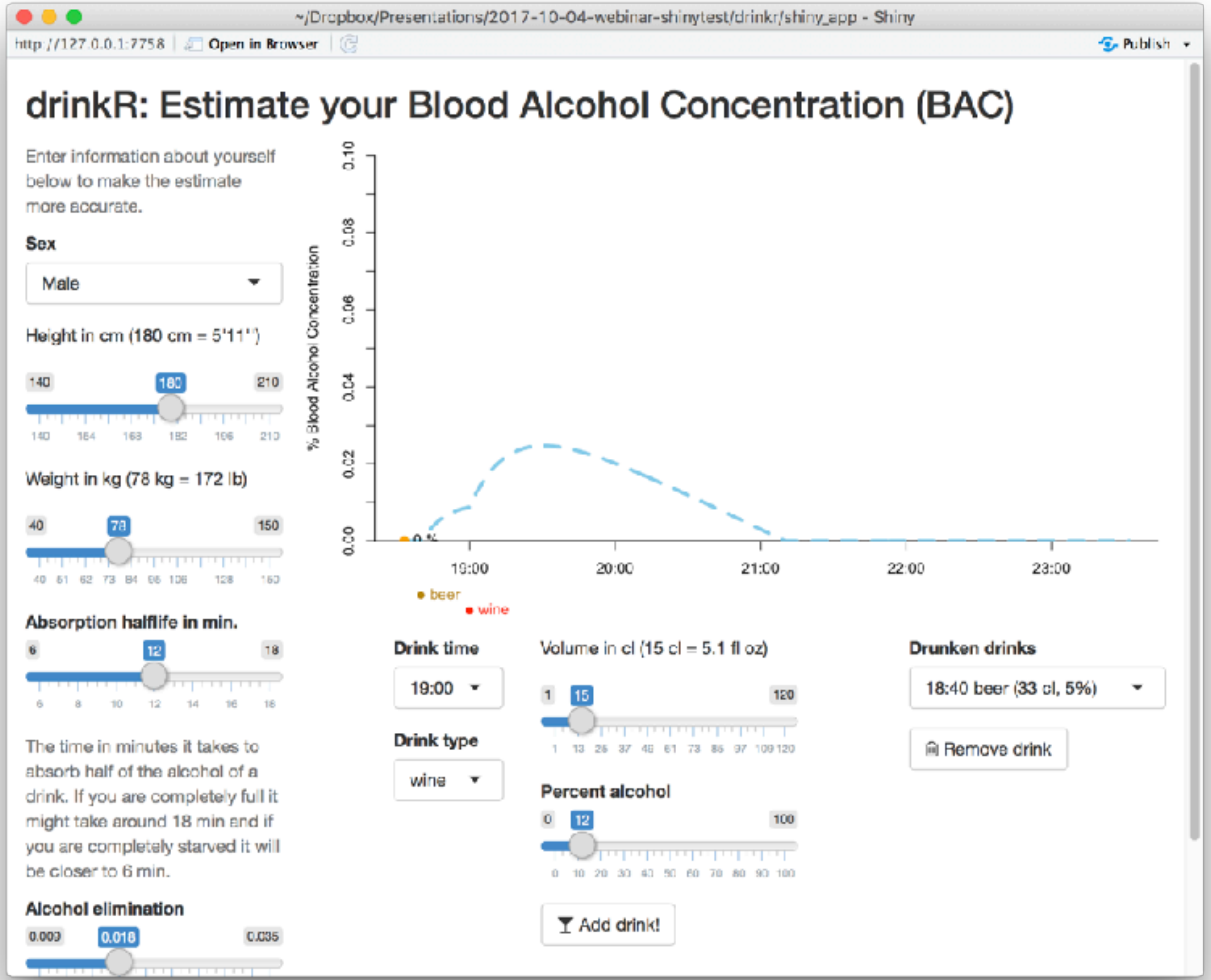
Testing Shiny applications with Shinytest

Winston Chang

RStudio

Webinar series

2017-10-04


~/Dropbox/Presentations/2017-10-04-webinar-shinytest/drinkr/shiny_app - Shiny

http://127.0.0.1:7758 | Open in Browser | Publish

drinkR: Estimate your Blood Alcohol Concentration (BAC)

Enter information about yourself below to make the estimate more accurate.

Sex: Male

Height in cm (180 cm = 5'11''): 180

Weight in kg (78 kg = 172 lb): 78

Absorption halflife in min.: 12

The time in minutes it takes to absorb half of the alcohol of a drink. If you are completely full it might take around 18 min and if you are completely starved it will be closer to 6 min.

Alcohol elimination: 0.018

Drink time: 19:00

Drink type: wine

Volume in cl (15 cl = 5.1 fl oz): 15

Percent alcohol: 12

Drunken drinks: 18:40 beer (33 cl, 5%)

App by Rasmuth Bååth
http://www.sumsar.net/blog/2014/07/estimate-your-bac-using-drinkr/

You've developed a nice app.

You've put it in production.

You want to be confident that it will keep running in the future.

Things that can change or break a Shiny application:

Modifying your application code

Upgrading Shiny

Upgrading other packages

Upgrading R

External data source changes or fails

The answer is… testing!

Manual testing: takes a lot of time, is inconsistent.

Automated testing: is really hard.

Why?

Because it requires a web browser, simulating user interactions with the browser, and writing tests for graphical elements.

Shinytest

Expectation-based testing

```
expect_equal(app$getValue("x"), 1234)
expect_equal(app$getValue("y"), "Some text")
expect_true(app$getValue("z") < 100)
```

Snapshot-based testing

```
app$snapshot()
```

Records state of application

```
snapshotCompare()
```

Compare this snapshot to a previous good snapshot

Expectation-based testing

- More precise: can target very granular pieces of code
- Can only test pieces of an application
- Harder to create tests
- Very hard to test graphical elements

Snapshot-based testing

- Can be easier to create tests
- Can test an entire application
- More sensitive to spurious changes

Shinytest procedure

- **Create a test:** Record user interactions with the app, saves them in a *test script*.
- **Make baseline (expected) snapshots:** Run the test script, which replays the interactions on a headless browser. This takes snapshots of application state along the way and saves them for later comparison.
- **Do your work:** Modify your app, modify your data, upgrade packages, upgrade R.
- **Re-run the test script and compare:** Run the test script again and take snapshots, then compare the new snapshots to the expected snapshots.

Installing shinytest

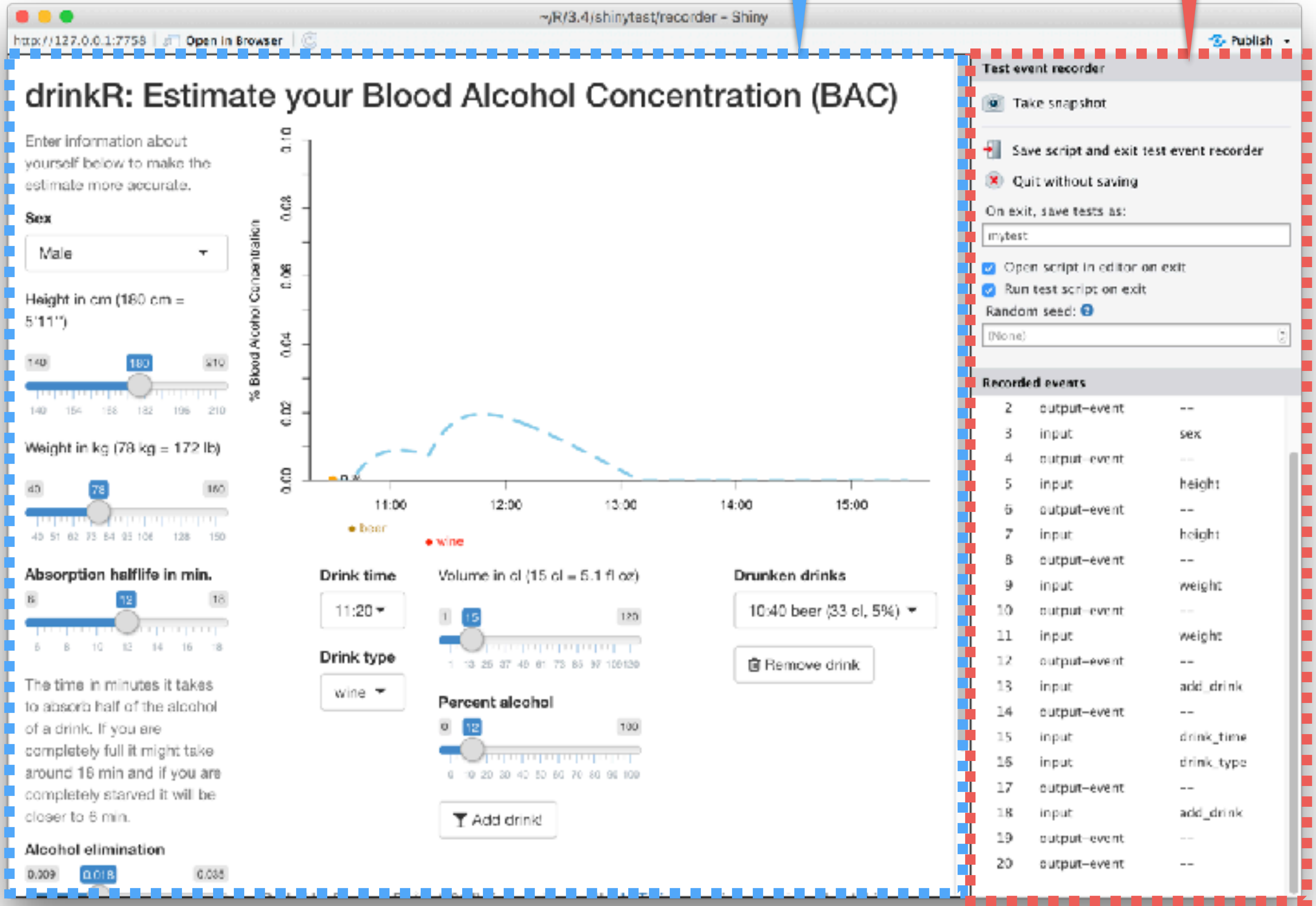
```
install.packages("devtools")

devtools::install_github("rstudio/shinytest")
```

Creating a test script

```
recordTest("path/to/app")
```

Target app

Recorder app



drinkR: Estimate your Blood Alcohol Concentration (BAC)

Enter information about yourself below to make the estimate more accurate.

Sex: Male

Height in cm (180 cm = 5'11"): 180

Weight in kg (78 kg = 172 lb): 78

Absorption halflife in min.: 12

The time in minutes it takes to absorb half of the alcohol of a drink. If you are completely full it might take around 18 min and if you are completely starved it will be closer to 6 min.

Alcohol elimination: 0.018

Drink time: 11:20

Drink type: wine

Volume in cl (15 cl = 5.1 fl oz): 15

Percent alcohol: 12

Add drink!

Drunken drinks: 10:40 beer (33 cl, 5%)

Remove drink

Test event recorder

Take snapshot

Save script and exit test event recorder

Quit without saving

On exit, save tests as: mytest

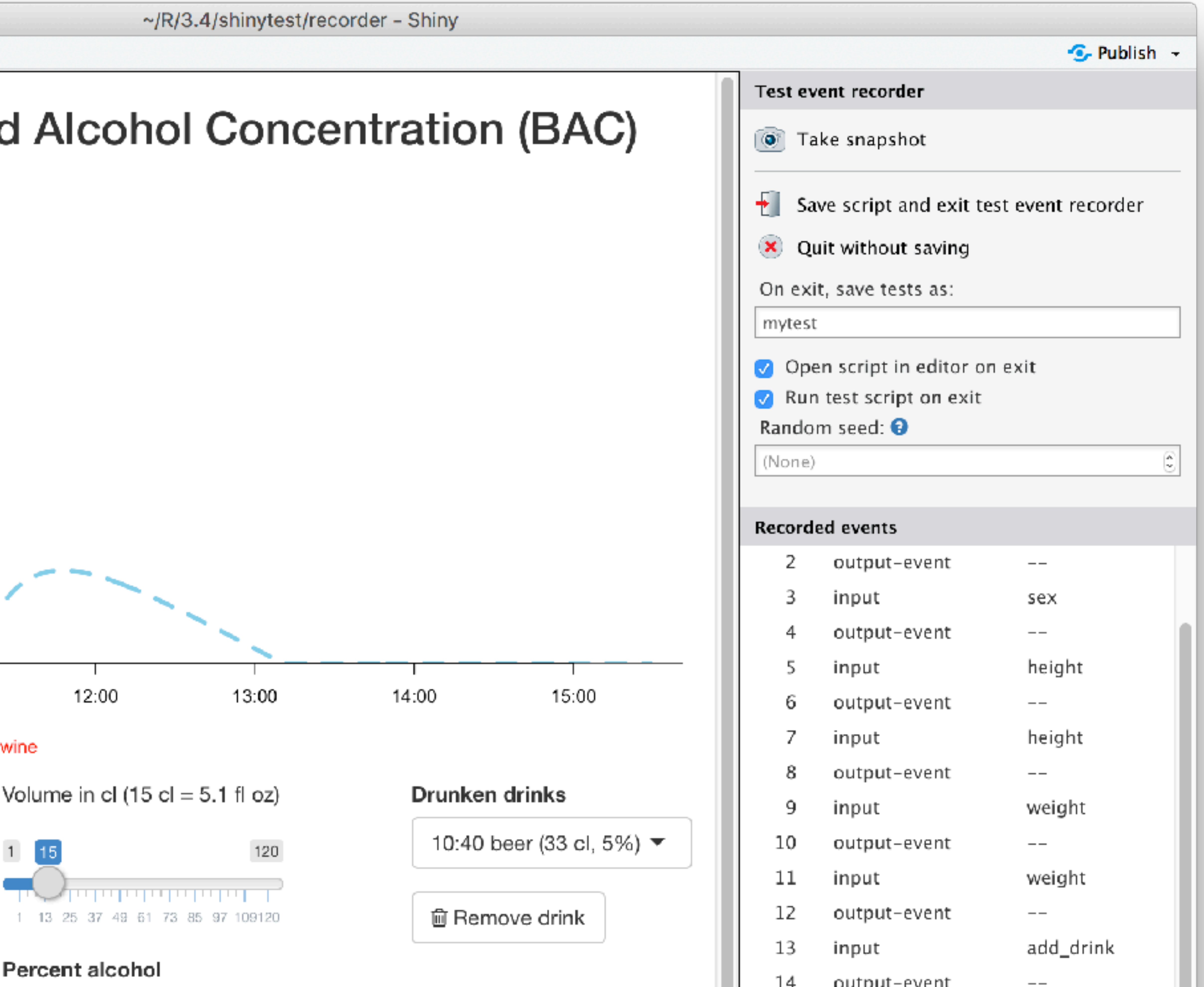
☑ Open script in editor on exit

☑ Run test script on exit

Random seed: (None)

Recorded events

2	output-event	--
3	input	sex
4	output-event	--
5	input	height
6	output-event	--
7	input	height
8	output-event	--
9	input	weight
10	output-event	--
11	input	weight
12	output-event	--
13	input	add_drink
14	output-event	--
15	input	drink_time
16	input	drink_type
17	output-event	--
18	input	add_drink
19	output-event	--
20	output-event	--



On exit, save tests as: mytest

- [✓] Open script in editor on exit
- [✓] Run test script on exit

Random seed: (None)

Recorded events

```
 2   output-event   --
 3   input          sex
 4   output-event   --
 5   input          height
 6   output-event   --
 7   input          height
 8   output-event   --
 9   input          weight
10   output-event   --
11   input          weight
12   output-event   --
13   input          add_drink
14   output-event   --
```

Example test script

```r
app <- ShinyDriver$new("../")
app$snapshotInit("mytest")

app$setInputs(sex = "male")
app$setInputs(height = 174)
app$setInputs(weight = 70)
app$setInputs(weight = 78)
app$setInputs(add_drink = "click")
app$snapshot()
app$setInputs(drink_time = "1507116600")
app$setInputs(drink_type = "wine")
app$setInputs(add_drink = "click")
app$snapshot()
```

Snapshots

Snapshots are numbered 001, 002, etc.

Each snapshot has two files:

001.json: The state of inputs, outputs, and exported values

001.png: A screenshot of the browser

Snapshot JSON file

```
{
  "input": {
    "add_drink": 1,
    "alc_perc": 5,
    "drink_type": "beer",
    "sex": "male",
    "volume": 33,
    "weight": 78
  },
  "output": {
    "volume_text": "Volume in cl (33 cl = 11.2 fl oz)",
    "weight_text": "Weight in kg (78 kg = 172 lb)",
    "bac_plot": {
      "src": "[image data sha1:
9a5f04d247365dacb6d3de1208c437e5057ea8da]",
      "width": 744,
      "height": 400,
...
```

Running tests

Runs all test scripts for an app

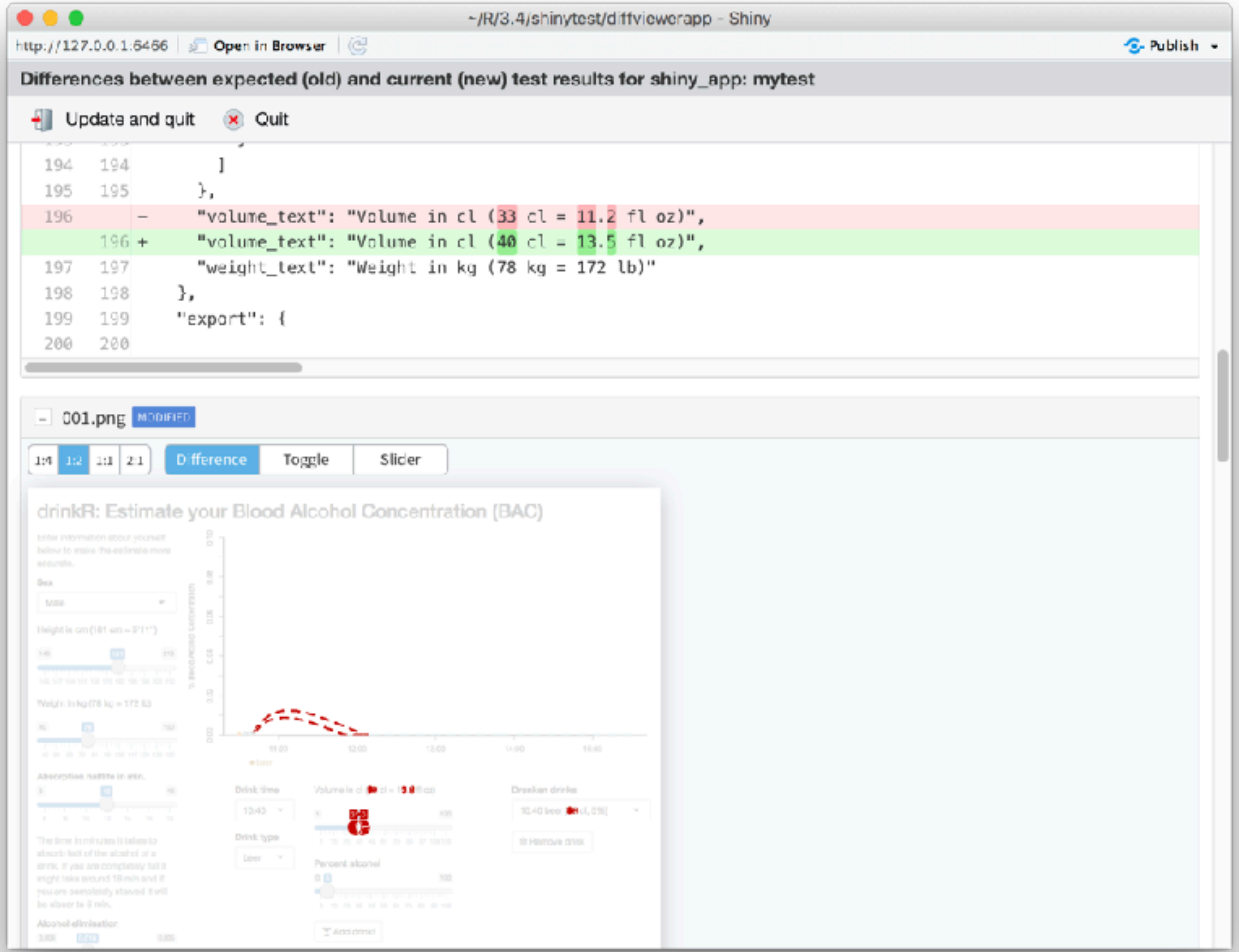
```
> testApp("path/to/app")
Running mytest.R
====== Comparing mytest... Passed.
```

Success (snapshots are as expected)

Maybe failure (snapshots differ from expected)

```
> testApp("path/to/app")
Running mytest.R
====== Comparing mytest...
  Differences detected between mytest-current/ and
mytest-expected/:

    Name              Status
    001.json   !=  Files differ
    001.png    !=  Files differ

Would you like to view the differences between expected
and current results [y/n]?
```

Difference viewer

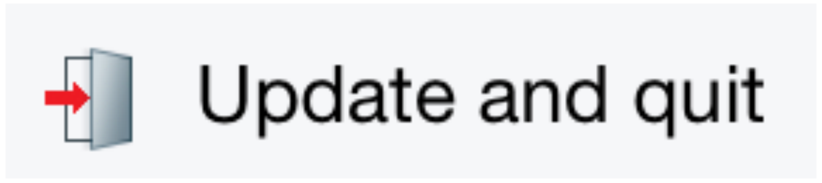


Differences between expected (old) and current (new) test results for shiny_app: mytest

```
          ]
        },
-       "volume_text": "Volume in cl (33 cl = 11.2 fl oz)",
+       "volume_text": "Volume in cl (40 cl = 13.5 fl oz)",
        "weight_text": "Weight in kg (78 kg = 172 lb)"
      },
      "export": {
```

```
# Alternate method of launching diff viewer
viewTestDiff("path/to/app", "mytest")
```

What kind of changes can happen?

Benign $\longrightarrow$ 

- Minor plot rendering change
- Minor print output change
- New outputs (after modifying app)

Problematic $\longrightarrow$

$\longrightarrow$ **Fix app**

- App crashes or doesn't start
- Plots don't render
- Wrong output

Editing test scripts

Consolidate - set multiple inputs at once for faster tests

```
app$setInputs(height = 181)
app$setInputs(weight = 78)
```

$\longrightarrow$

```
app$setInputs(height = 181,
              weight = 78)
```

Remove redundant/unnecessary calls to setInput()

```
app$setInputs(height = 175)
app$setInputs(height = 181)
```

$\longrightarrow$

```
app$setInputs(height = 181)
```

Add setInput() calls

```
app$setInputs(height = 181)
```

$\longrightarrow$

```
app$setInputs(sex = "male")
app$setInputs(height = 181)
```

Editing test scripts

Add snapshots

```
app$setInputs(height = 175)
app$setInputs(height = 181)
```

$\rightarrow$

```
app$setInputs(height = 175)
app$snapshot()
app$setInputs(height = 181)
```

Enable/disable screenshots

```
app$snapshotInit("mytest")
```

$\rightarrow$

```
app$snapshotInit("mytest",
            screenshot = FALSE)
```

Add delay between steps

```
app$setInputs(height = 175)
app$setInputs(height = 181)
```

$\rightarrow$

```
app$setInputs(height = 175)
Sys.sleep(4)
app$setInputs(height = 181)
```

When should I run tests?

Am I doing something that could cause the behavior of my application to change?

- Modifying your application
- Upgrading packages
- Upgrading R
- Changes to external data

Continuous Integration

Test application with each commit using platforms like Travis CI

CI Challenges

- Graphical output can differ between development platform and CI platform, so comparing screenshots might not be possible.
- Can be difficult or impossible to retrieve snapshots for inspection.

https://rstudio.github.io/shinytest/articles/ci.html

https://github.com/rstudio/shinytest-ci-example

https://github.com/rstudio/shinytest-ci-example-multi

Future support in RStudio Connect and Shiny Server Pro

- Automatically run tests when apps are deployed
- Automatically run tests on a schedule

Limitations

- Recorder is pretty good, but not perfect.
- Recorder does not capture input values from htmlwidgets (like leaflet, plotly).
- Applications that use a dynamic external data source are harder to test.

Dealing with dynamic data

- Don't use screenshots
- Snapshot targeted parts of an application
- Detect test mode and use a dummy data set

```
if (isTRUE(getOption("shiny.testmode")))
```

Future plans

- CRAN release by end of year
- RStudio Connect and Shiny Server Pro integration
- Tools to make it easier to work with dynamic data
- Better integration with RStudio IDE

Thanks!

https://rstudio.github.io/shinytest/