Lessons learned from testing 2500+ Shiny Apps every day

A brief history in testing Shiny

Barret Schloerke
Posit / Shiny Team
@schloerke

Testing?
…Never heard of them

- No testing experience?
  - 📖 R Packages, Chapter 14: Testing
    https://r-pkgs.org/testing-basics.html
  - 📖 Mastering Shiny, Chapter 21: Testing
    https://mastering-shiny.org/scaling-testing.html
- Need a refresher on `{shinytest2}` and `{testthat}`?
  - ▶️ {shinytest2}: Testing Shiny with {testthat}

Do not be discouraged!

- Shiny's approach to testing is exhausting
  - As a core UI package, Shiny requires pixel perfect outputs
- Goals of testing Shiny != Goals of App testing
  - `{shinytest2}` should be enough for standard App/package testing

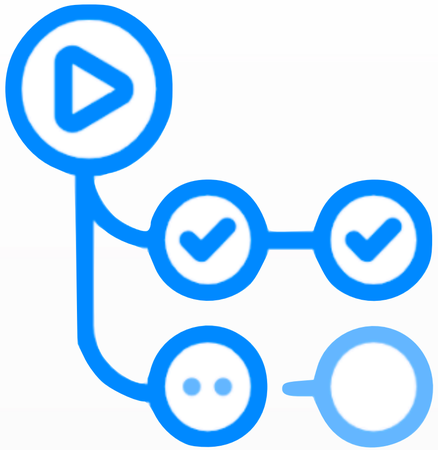
Testing Shiny in 2018 🐣

- Repo: rstudio/shiny-examples
  - ~150 apps
- Everyone on the team manually tested Apps for about 2 weeks
  - Must restart manual testing process for every new bug
  - Didn’t know of all across-repo issues until release time (~6 months)
- No guarantee that everyone has consistent testing environments
- Release time: (~10 team members) * (3+ weeks of time)

Testing Shiny in 2020 💪

- Repos: rstudio/shinycoreci-apps and rstudio/shinycoreci
- Consistent install methods
  - ✅ Increased trust in testing environments
  - ❌ Tested against a zip file of rstudio/shinycoreci-apps repo
- Updating snapshots was a laborious / manual process
- Some across repo bugs are not found until release time
- Release time: (~8 team members) * 2 weeks

Continuous Integration: GitHub Actions



- Documentation: https://github.com/features/actions
- Automate your workflows within world-class CI/CD
  - Build, test, and deploy your code right from GitHub
- Integrated within GitHub
- Pricing: Free for public repos!

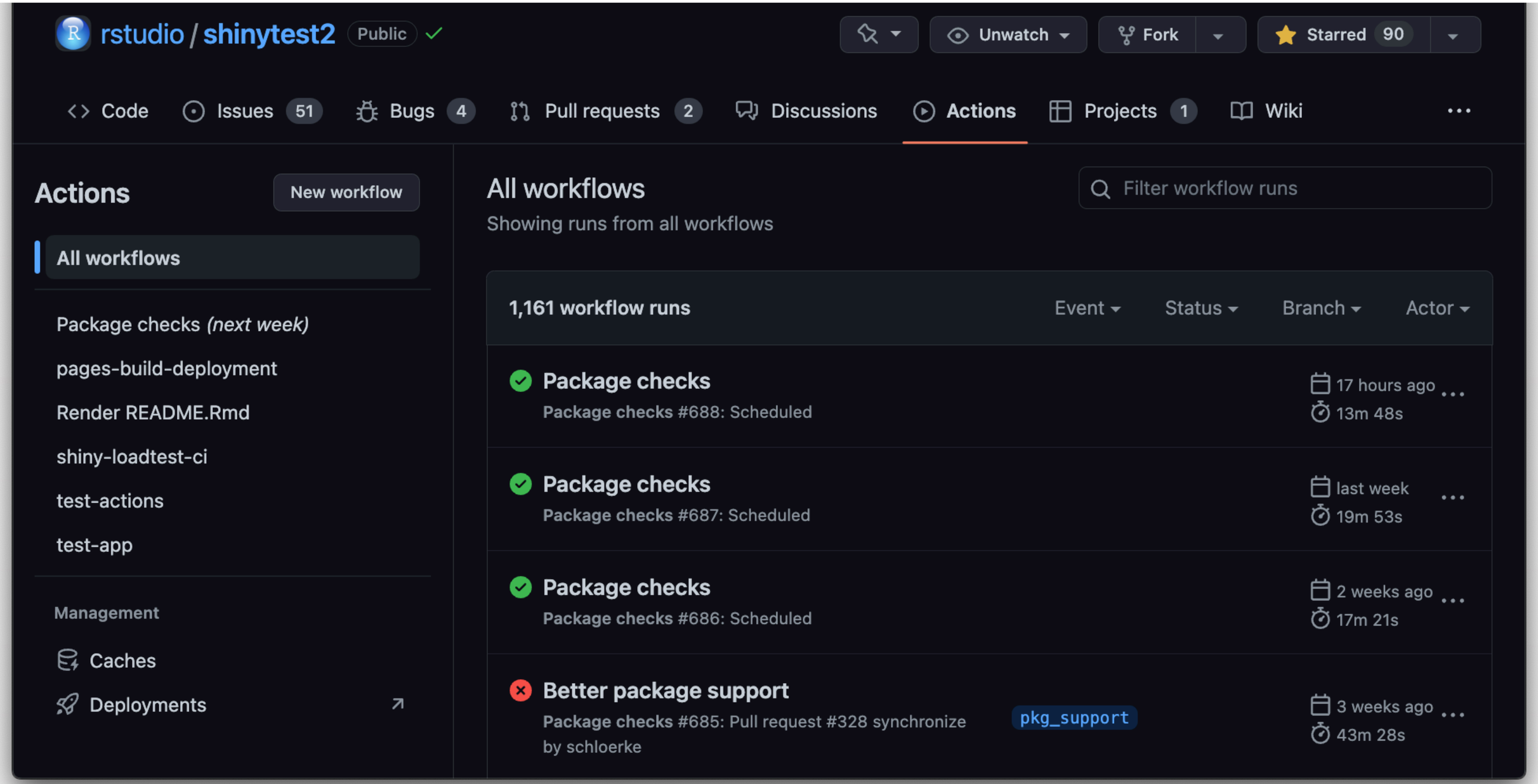
r-lib/actions

- GitHub Actions for the R language
  - Maintained by Gábor Csárdi

- Contains many **actions** to compose your workflow
  - Setup pandoc
  - Setup R
  - Setup R dependencies
  - Check R package

- Contains many **example workflows**

{shinytest2} Actions



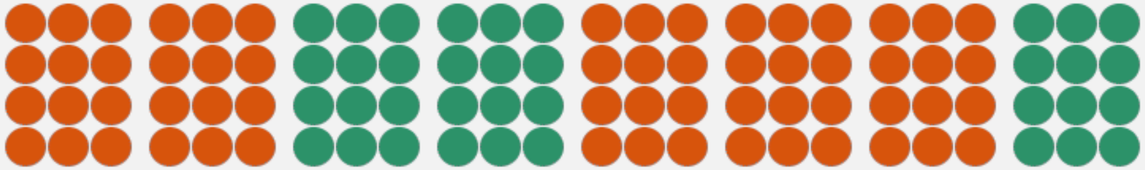
Testing Shiny in 2023

- Repo: rstudio/shinycoreci
  - Leverages GitHub Actions and r-lib/actions
- Consistent install and deploy methods
  - Full trust in every testing environment
- Nightly testing of 170x Shiny apps
  - 5x R versions: `release`, `oldrel-1`, `oldrel-2`, `oldrel-3`, `oldrel-4`
  - 3x Operating systems: `macos`, `ubuntu`, `windows`
  - 2,550 combinations in total!
  - Bugs are exposed within 24hrs of merging PR

https://rstudio.github.io/shinycoreci/results

rstudio/shinycoreci results: 2023-02-12 - 2023-02-22

2023-02-22

App	Performance	Fail	Pass	Can't install	Skip test	Not exec	Total fail	Total pass	Total can't install	Total skip test	Total not exec
All	All	A	A	All	A	Al	Al	Al	All	Al	Al
118-highcharter-births							0	12	0	0	0
138-icon-fontawesome							0	12	0	0	0
135-bookmark-uioutput							0	12	0	0	0

Testing Shiny in 2023

- Computing cost:
  45 mins * 15 combos * 5 days * 52 weeks
  > 3000 CI hours per year

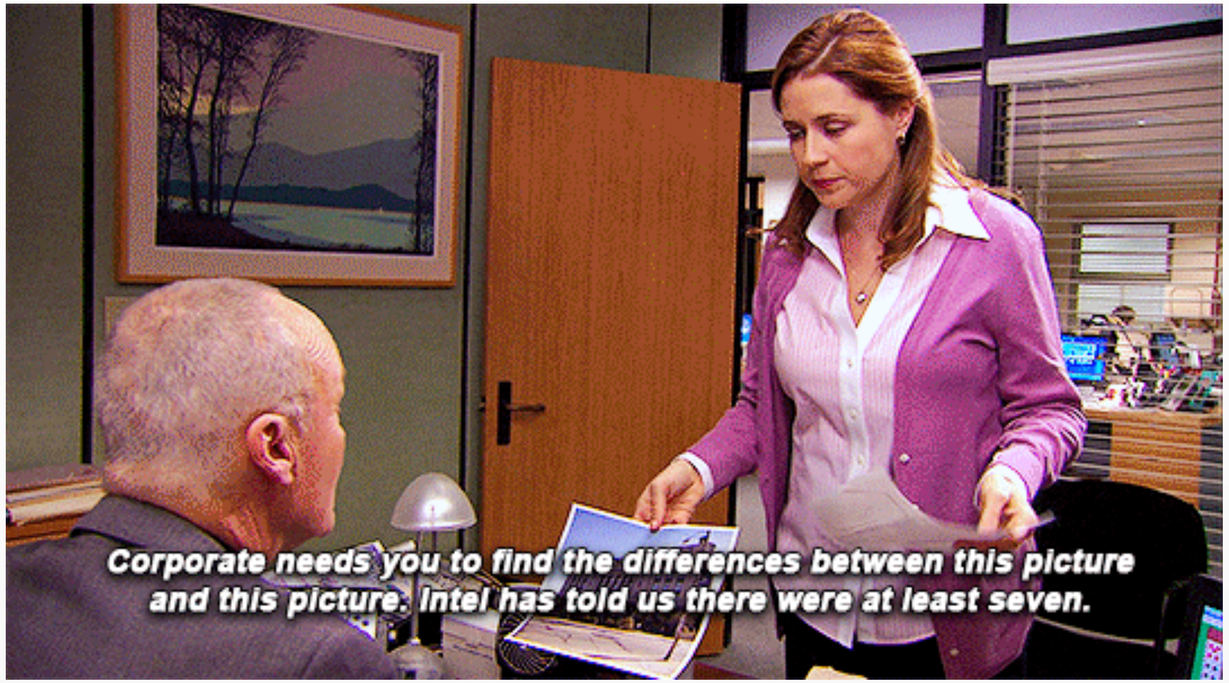
- GitHub Action Pricing
  - Open source: Free
  - Private: ~ $3k month (~75% for macOS testing)

- Release time: < 2 days * 4 developers
  - R, RStudio IDE, Connect, Workbench, Shinyapps.io, SSO / SSP
- Weekly maintenance: < 30 minutes

Testing lessons learned

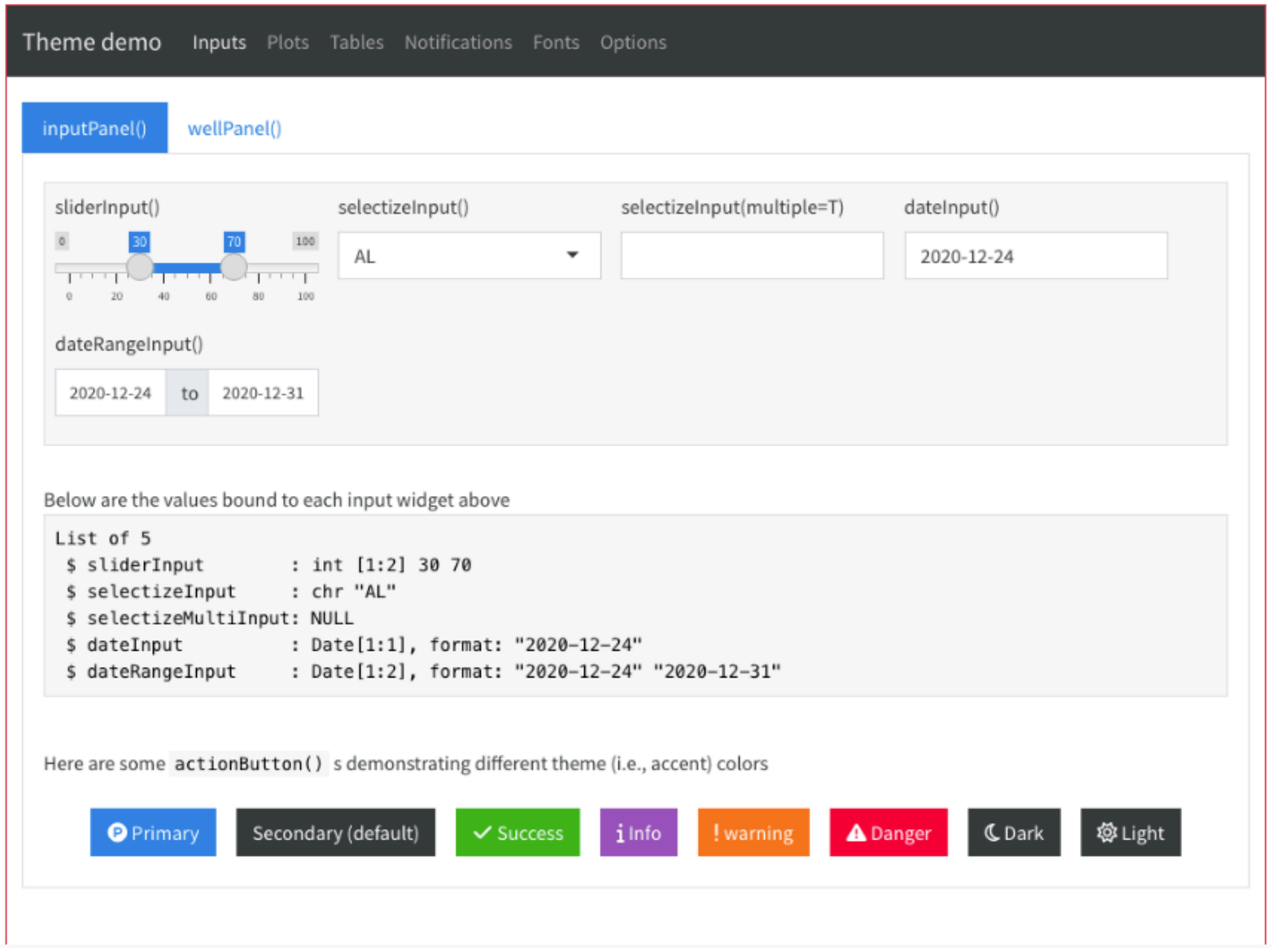
Lesson #1

Minimize number of screenshot expectations

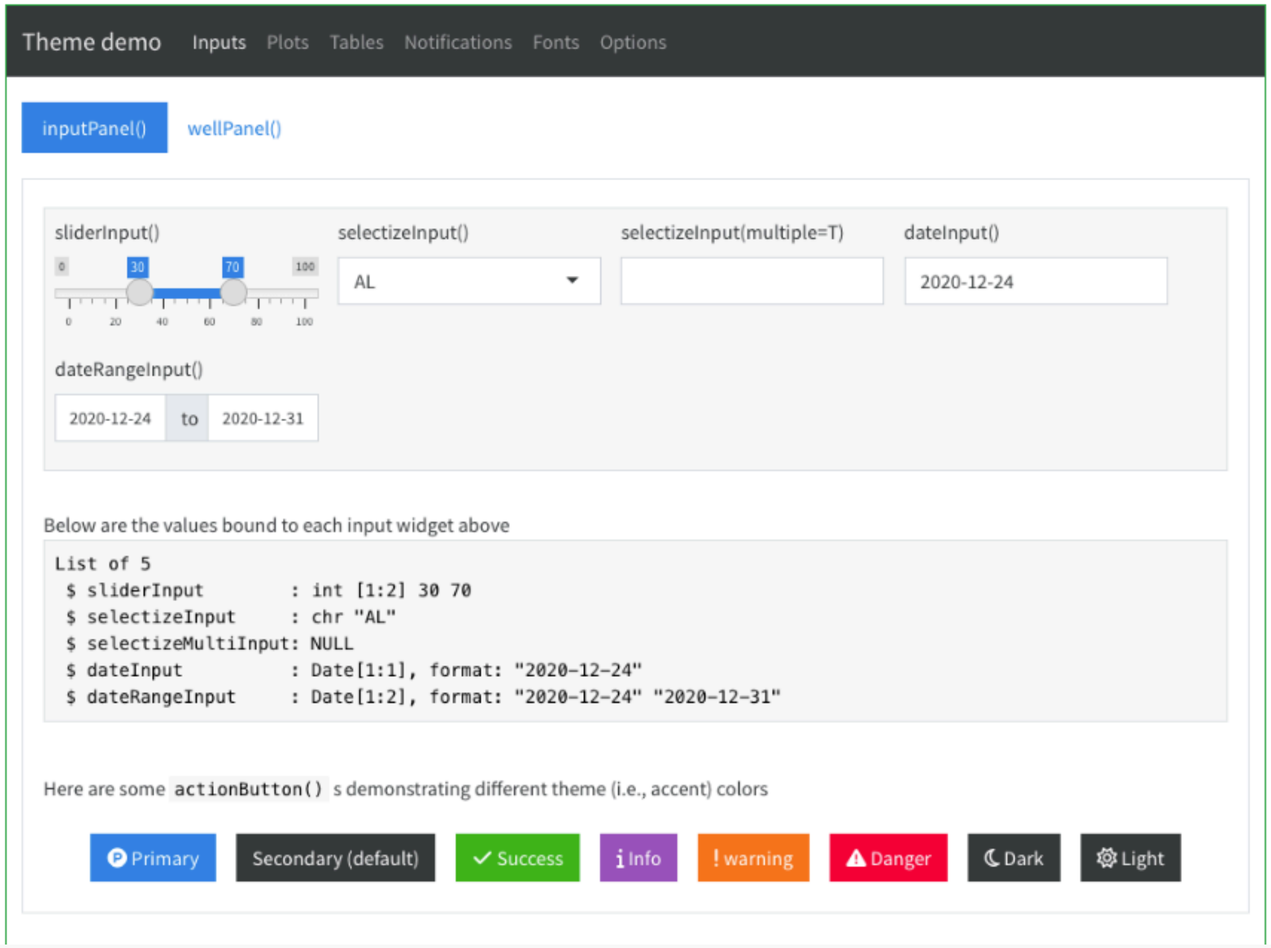


Valid screenshot change

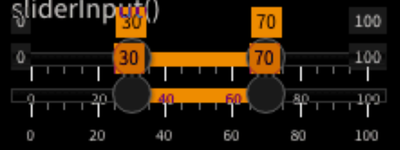
Deleted



Added



sliderInput()



dateRangeInput()

2020-12-24 to 2020-12-31

selectizeInput()

AL

selectizeInput(multiple=T)

dateInput()

2020-12-24

Below are the values bound to each input widget above

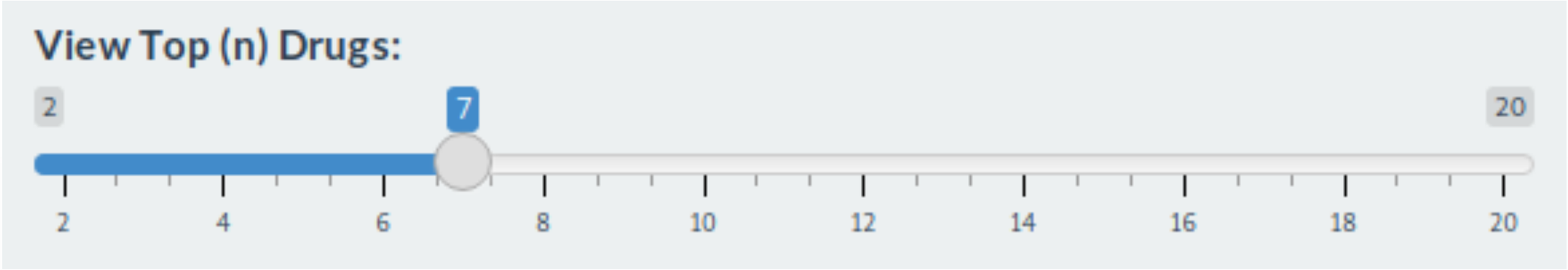
```
List of 5
 $ sliderInput     : int [1:2] 30 70
 $ selectizeInput  : chr "AL"
 $ selectizeMultiInput: NULL
 $ dateInput       : Date[1:1], format: "2020-12-24"
 $ dateRangeInput  : Date[1:2], format: "2020-12-24" "2020-12-31"
```

Here are some `actionButton()` s demonstrating different theme (i.e., accent) colors

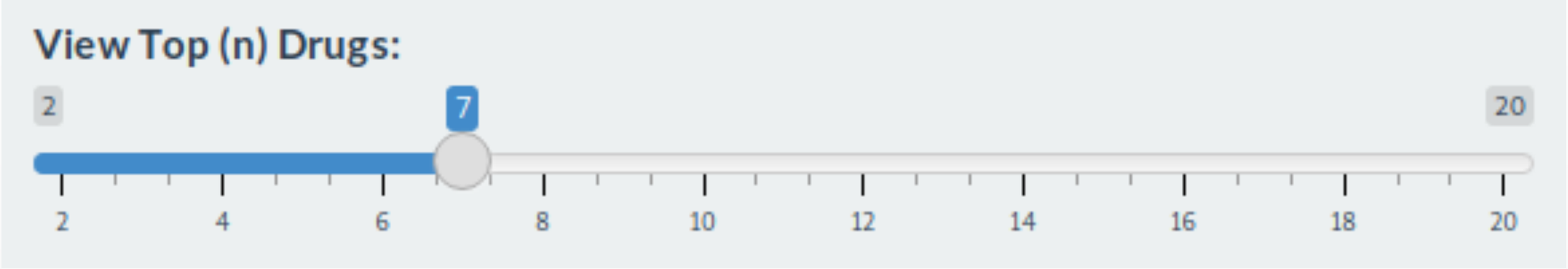
Primary Secondary (default) Success Info Warning Danger Dark Light

False-positive difference

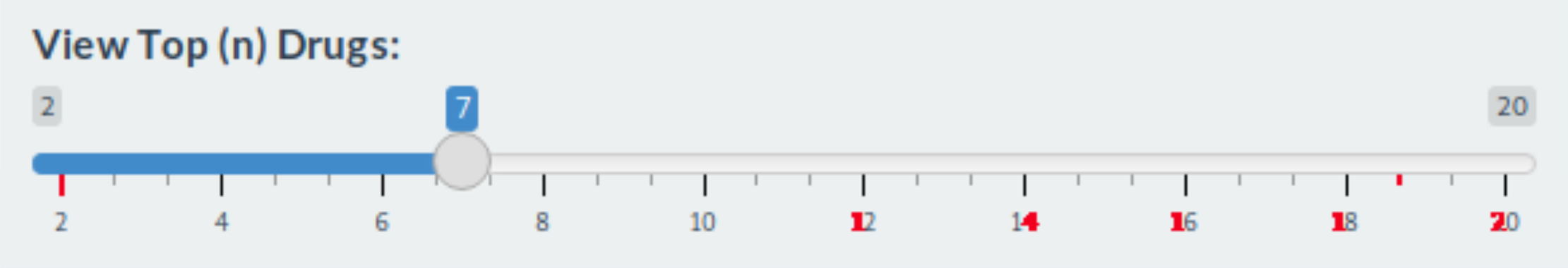
Old



New

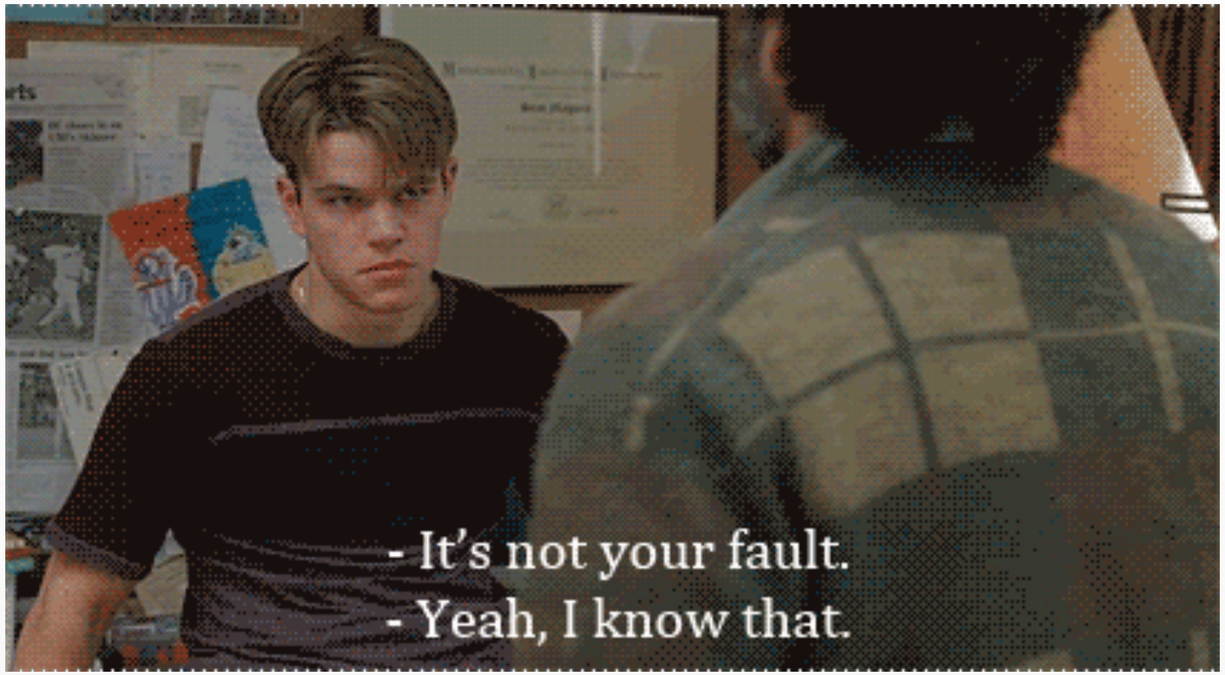


False-positive difference



Lesson #2

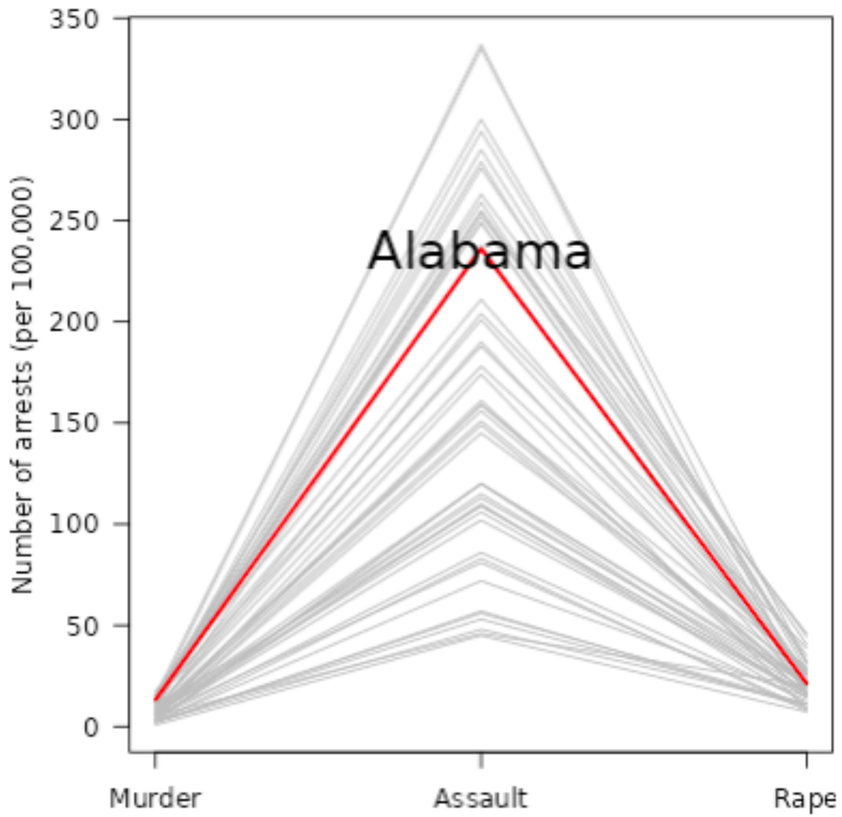
A test failure may not be your fault!



… but it's still your problem

- GitHub routinely updates CI machines
- Slow machines make client side timing difficult
  - CSS animations are tricky!
- Lots of trouble with `ubuntu` due to system font changes

Baseline image



Alabama

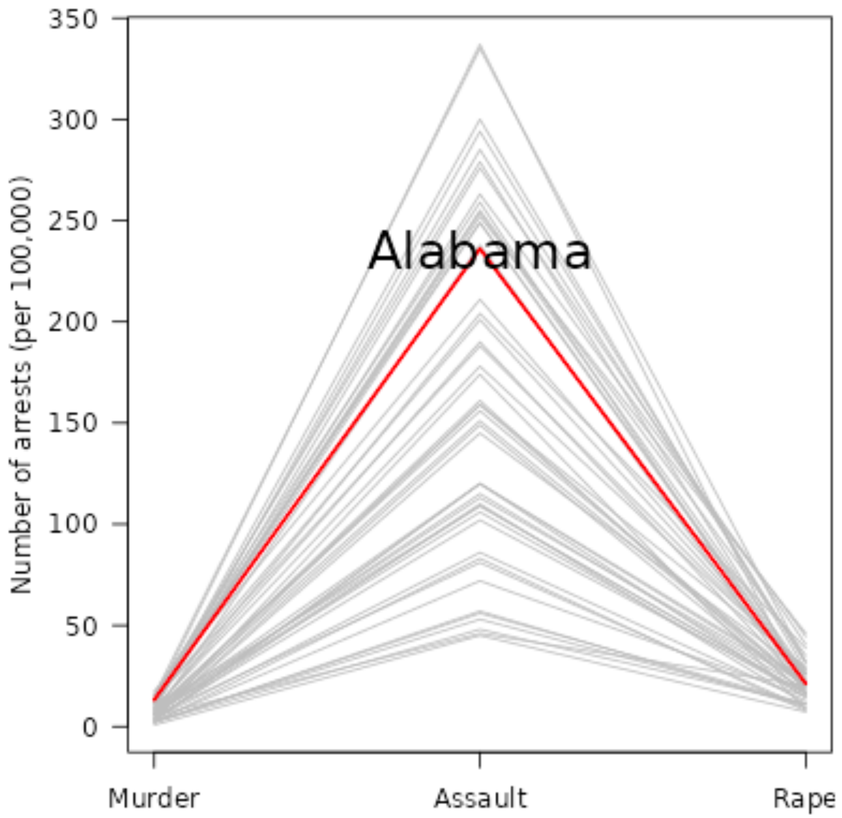
Show 10 entries

Search:

State	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	263	48	44.5
Arizona	8.1	294	80	31
Arkansas	8.8	190	50	19.5
California	9	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8

Showing 1 to 10 of 50 entries Previous 1 2 3 4 5 Next

With font change



Alabama

Show 10 entries

Search:

State	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	263	48	44.5
Arizona	8.1	294	80	31
Arkansas	8.8	190	50	19.5
California	9	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8

Showing 1 to 10 of 50 entries Previous 1 2 3 4 5 Next

Use `threshold=`!

- Allow for minor differences when comparing screenshots
- Example expectation from test app 114-modal-dialog

```
app$expect_screenshot(threshold = 2)
```

- Example expectation from test app 302-bootswatch-themes

```
app$expect_screenshot(
  # Try to get the sliders to settle
  delay = 1,
  # 3% tolerance with 10k pixels over 3 channels
  threshold = 900,
  kernel_size = 100
)
```

Lesson #3

Test values that you can control

Test values you can control

- Changes in a dependencies can produce new output values
  - Use `shiny::exportTestValues()` to test Shiny `server` values!
- Save only necessary information in your expected values
  - Use `shiny::snapshotExclude()`!
  - Use `shiny::snapshotPreprocessInput()`!
  - Use `shiny::snapshotPreprocessOutput()`!

{plotly} output example

- Only save the `x` and `y` values of the first dataset in the plot

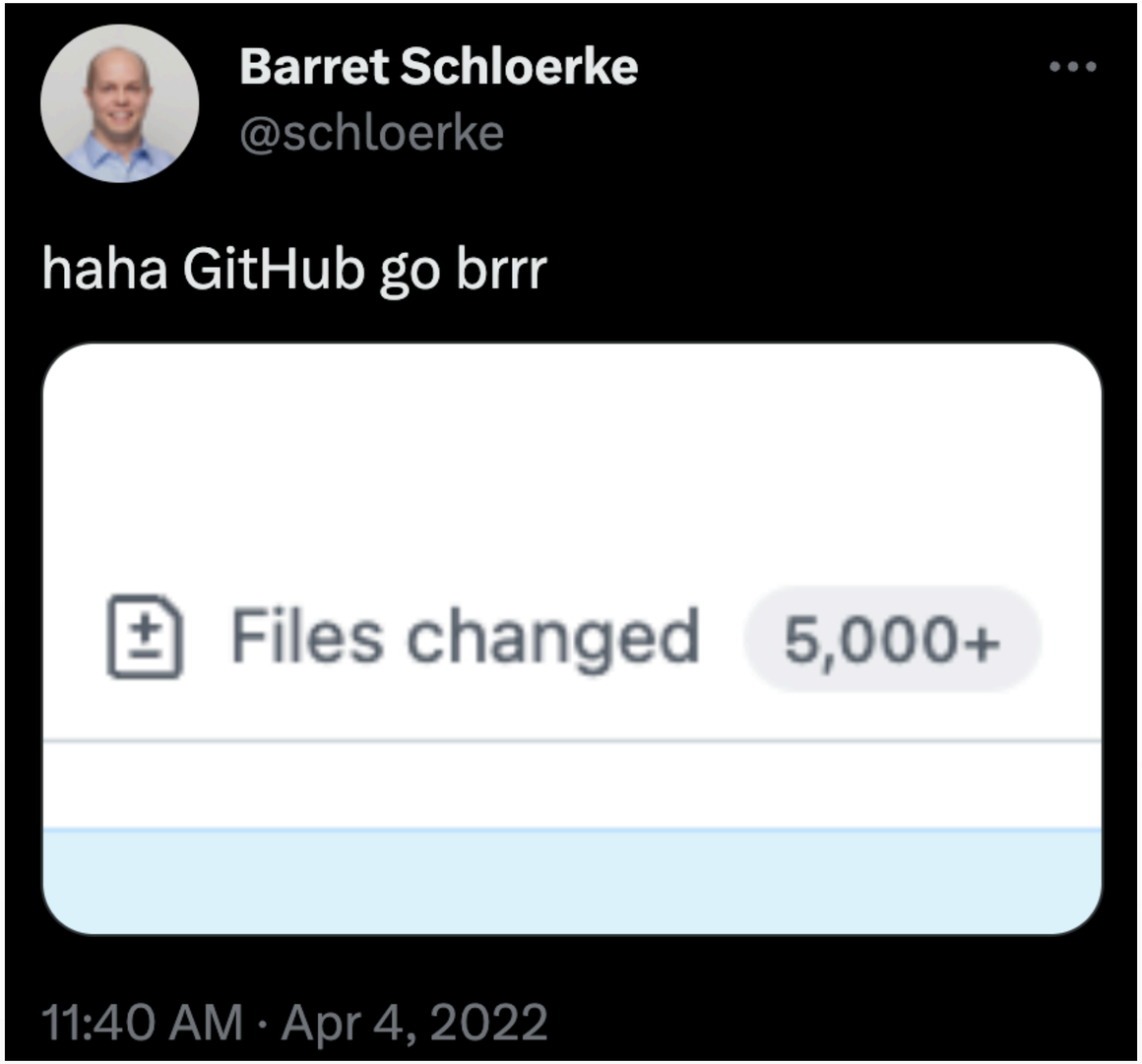
```
server <- function(input, output) {

  output$p <-
    plotly::renderPlotly({
      plotly::plot_ly(x = cars[, 1], y = cars[, 2],
                      type = "scattergl", mode = "markers")
    }) |>
      shiny::snapshotPreprocessOutput(function(p) {
        jsonlite::parse_json(p)$x$data[[1]][c("x", "y")]
      })

}
```

Lesson #4

Routine maintenance is maintainable


Barret Schloerke
@schloerke
haha GitHub go brrr
Files changed
5,000+
11:40 AM · Apr 4, 2022

Lessons learned while testing Shiny

1. Minimize the number of screenshot expectations

2. Just because it failed, doesn’t mean it is your fault
   But it is your problem!

3. Test values that you can control

4. Perform routine test maintenance

Lessons learned from testing 2500+ Shiny Apps every day

A brief history in testing Shiny

- Slides: https://bit.ly/appsilon-testing-shiny
- Websites:
  - `{shinytest2}`: https://rstudio.github.io/shinytest2/
  - `r-lib/actions`: https://github.com/r-lib/actions
  - GitHub Actions: https://github.com/features/actions
- 📖 Mastering Shiny, Chapter 21: Testing https://mastering-shiny.org/scaling-testing.html
- 📖 R Packages, Chapter 14: https://r-pkgs.org/testing-basics.html

Barret Schloerke
Posit / Shiny Team
@schloerke

Questions?