PRACTICAL INTRO TO SHINY

AND OTHER LESSONS FROM RSTUDIO::CONF 2019

CLARA OROMENDIA – COMPUTING CLUB 1/31/2019

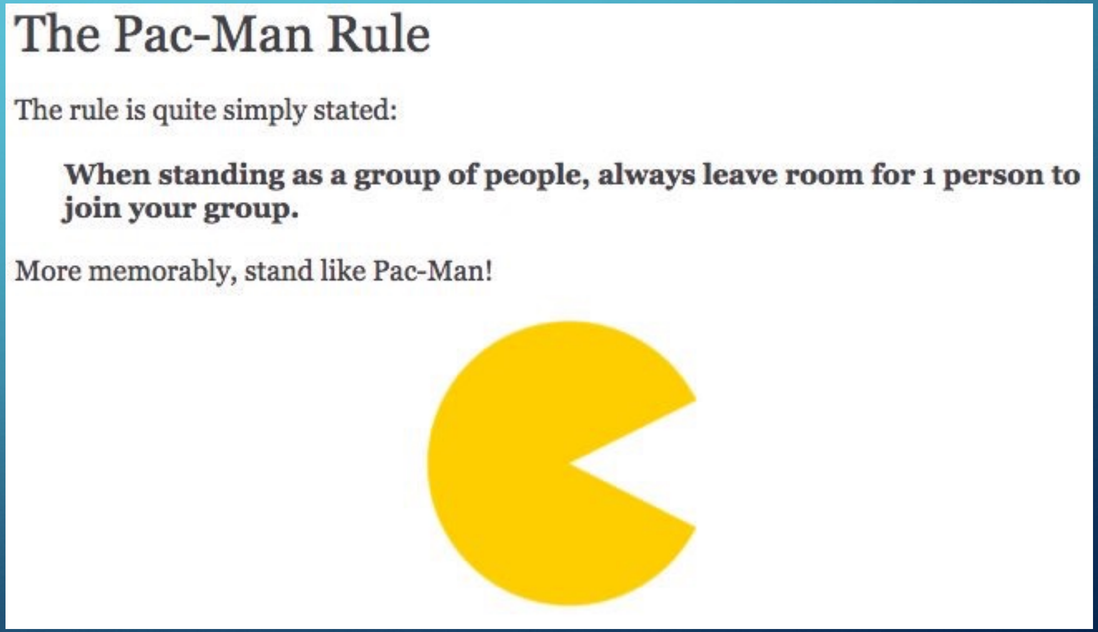
LESSONS FROM RSTUDIO::CONF

ALL LECTURES : HTTPS://RESOURCES.RSTUDIO.COM/RSTUDIO-CONF-2019

WORKSHOP MATERIALS: HTTPS://GITHUB.COM/RSTUDIO/RSTUDIO-CONF/TREE/MASTER/2019

GENERAL

- People like stickers.
- Intentional inclusivity makes a difference
- Twitter is huge
- It’s exhausting to meet people



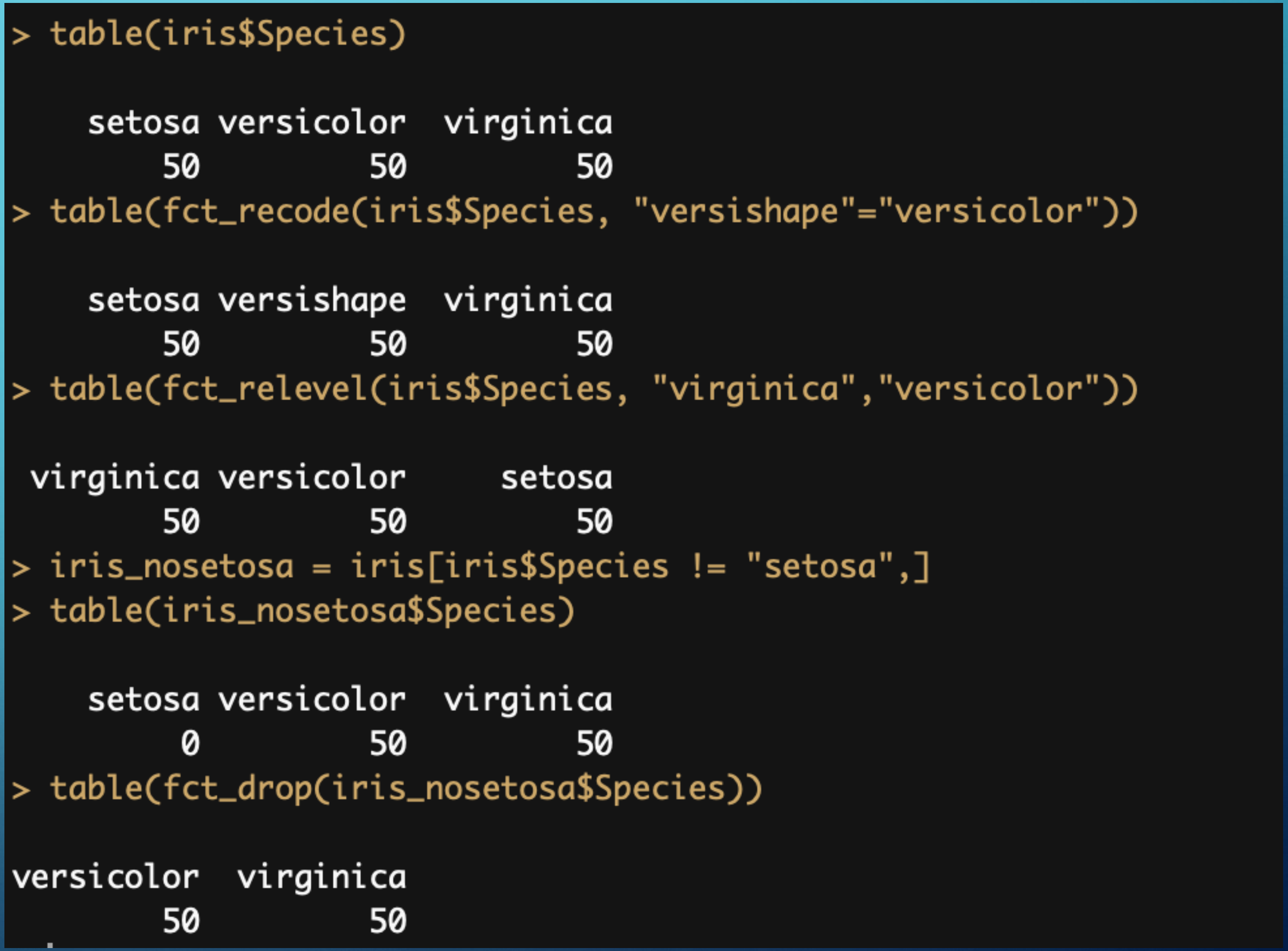
The Pac-Man Rule

The rule is quite simply stated:

> **When standing as a group of people, always leave room for 1 person to join your group.**

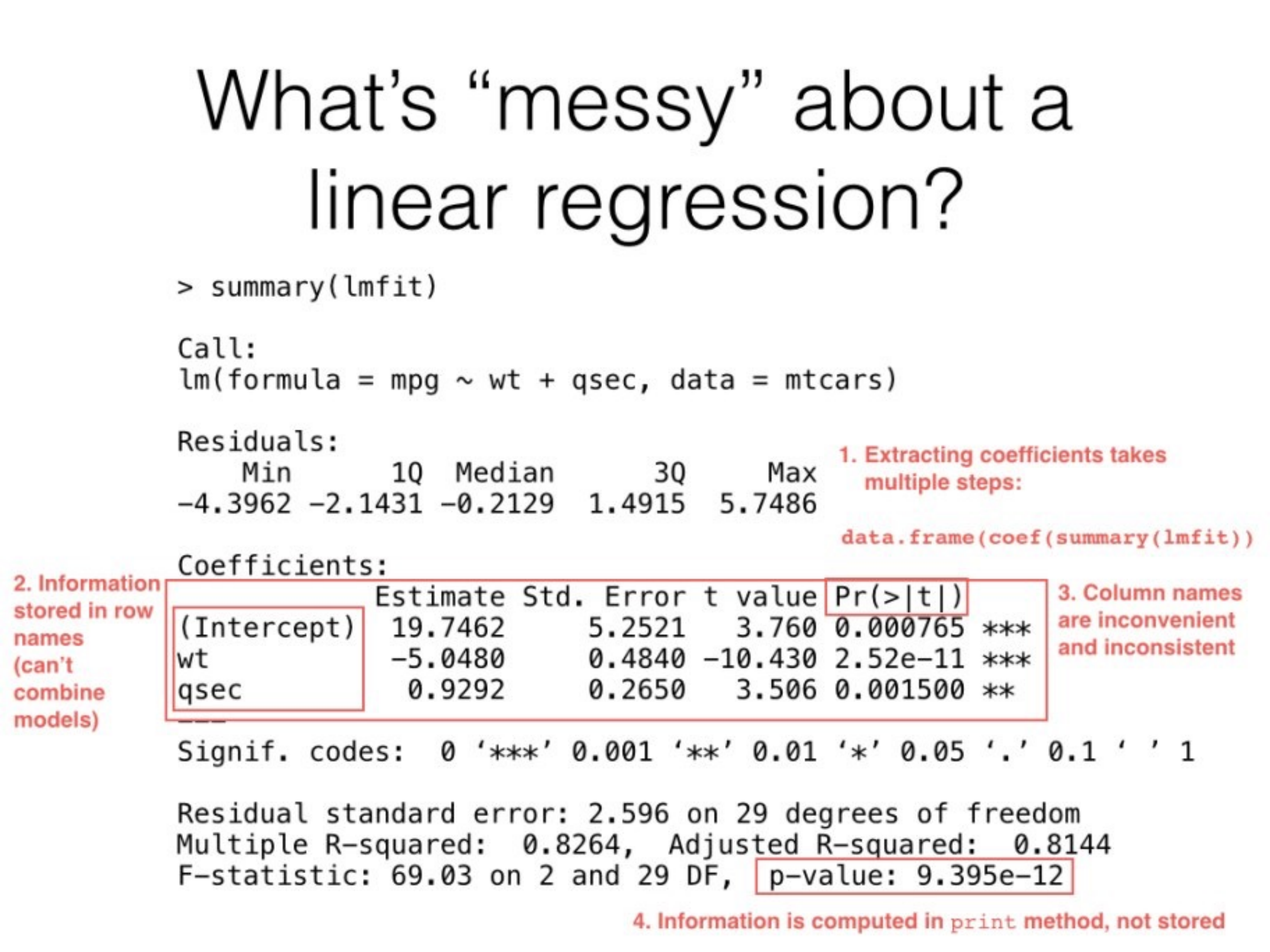
More memorably, stand like Pac-Man!

FORCATS



```
> table(iris$Species)

    setosa versicolor  virginica 
        50         50         50 
> table(fct_recode(iris$Species, "versishape"="versicolor"))

    setosa versishape  virginica 
        50         50         50 
> table(fct_relevel(iris$Species, "virginica","versicolor"))

 virginica versicolor     setosa 
        50         50         50 
> iris_nosetosa = iris[iris$Species != "setosa",]
> table(iris_nosetosa$Species)

    setosa versicolor  virginica 
         0         50         50 
> table(fct_drop(iris_nosetosa$Species))

versicolor  virginica 
        50         50 
```

BROOM



What's "messy" about a linear regression?

```
> summary(lmfit)

Call:
lm(formula = mpg ~ wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.3962 -2.1431 -0.2129  1.4915  5.7486 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  19.7462     5.2521   3.760 0.000765 ***
wt           -5.0480     0.4840 -10.430 2.52e-11 ***
qsec          0.9292     0.2650   3.506 0.001500 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,	Adjusted R-squared:  0.8144 
F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12
```

1. Extracting coefficients takes multiple steps:

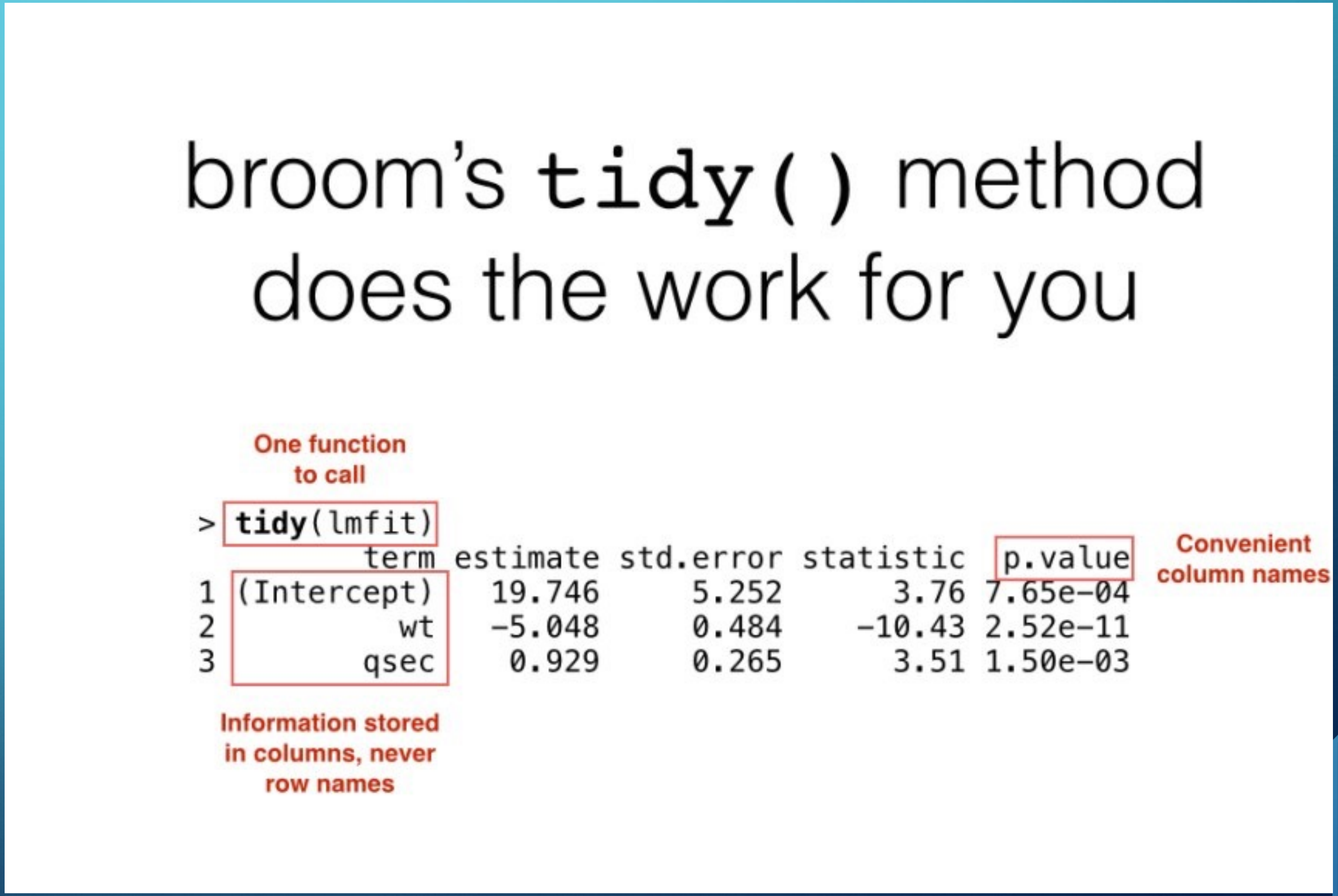
`data.frame(coef(summary(lmfit))`

2. Information stored in row names (can't combine models)

3. Column names are inconvenient and inconsistent

4. Information is computed in `print` method, not stored

BROOM



broom's **tidy()** method does the work for you

One function to call

```
> tidy(lmfit)
         term estimate std.error statistic  p.value
1 (Intercept)   19.746     5.252      3.76 7.65e-04
2          wt   -5.048     0.484    -10.43 2.52e-11
3        qsec    0.929     0.265      3.51 1.50e-03
```

Convenient column names

Information stored in columns, never row names

RSTUDIO TIPS

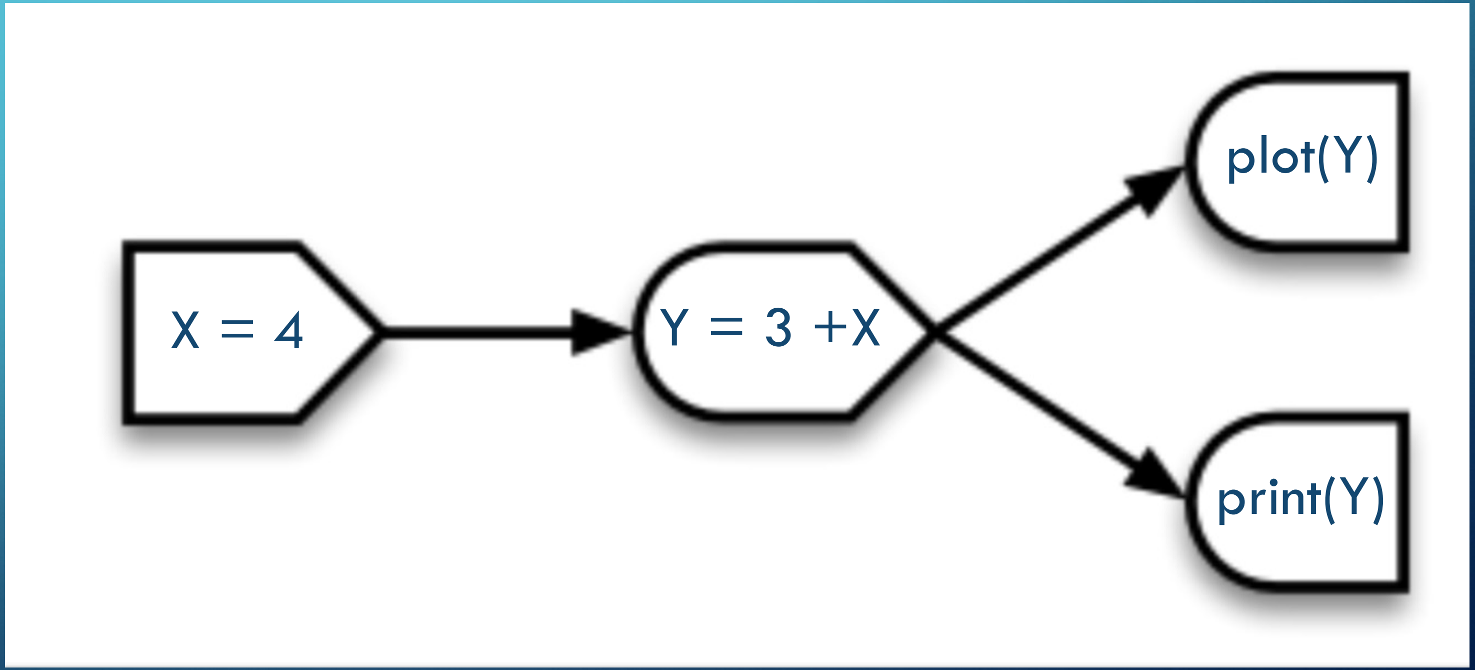
- Use projects
- Faster typing
  - Snippets: lib + tab → library(…)
  - Shortcuts
- Debug using browser()

INTRO TO SHINY

SOME SLIDES TAKEN WITH PERMISSION FROM AIMEE GOTT AND RSTUDIO TEAM

SHINY BASICS

- Reactive programming with R
  - X = 4
  - Y = 3 + X
  - X = 7
  - Y = ?
- Runs as HTML



EXAMPLE 0: BUILT IN

- Rstudio has a (boring) shiny template
  - New File - > Shiny Web App

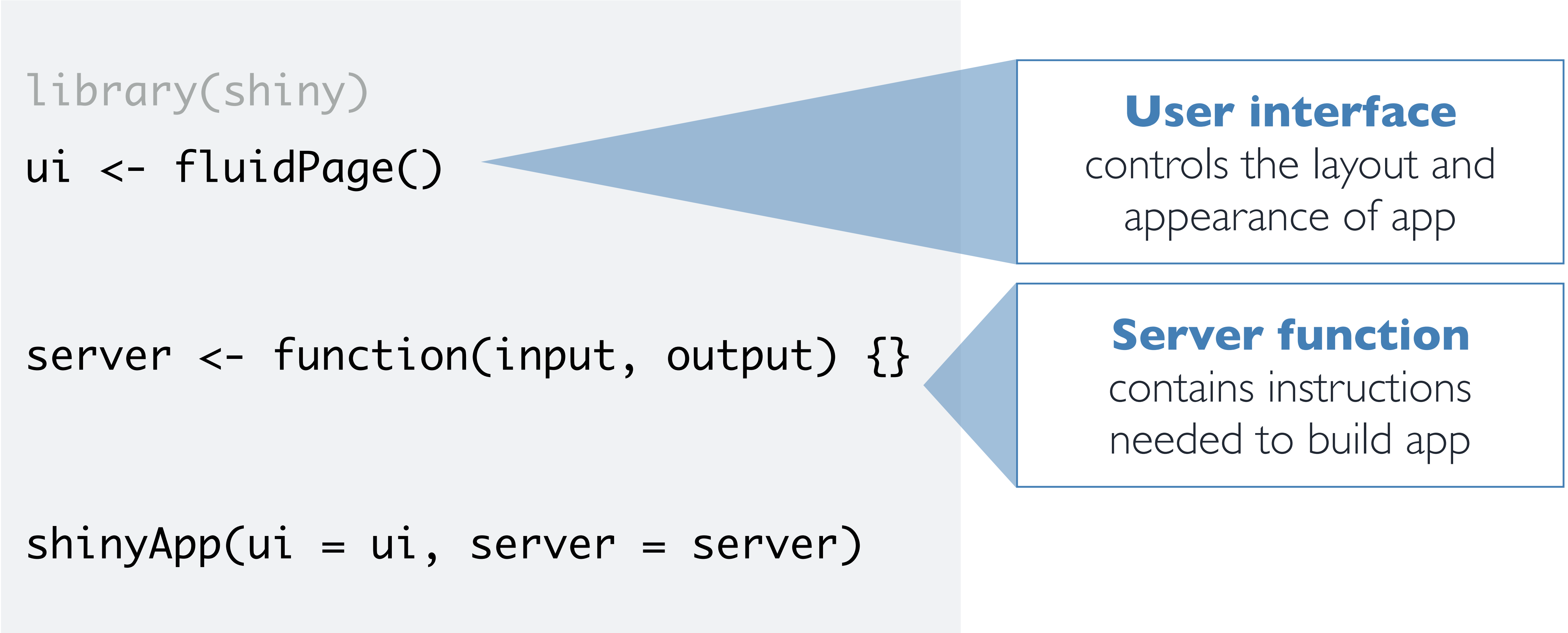
- App separated into UI and Server

print(Y)

EXAMPLE 0B: MOVIES

- Navigate to folder *app0b_movies* and open *app.r*
  - Green play button will appear – press to launch!
- App separated into UI and Server

WHAT’S IN AN APP?



```
library(shiny)
ui <- fluidPage()


server <- function(input, output) {}


shinyApp(ui = ui, server = server)
```

User interface
controls the layout and
appearance of app

Server function
contains instructions
needed to build app

User interface

```r
# Define UI for application that plots features of movies
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create fluid page layout

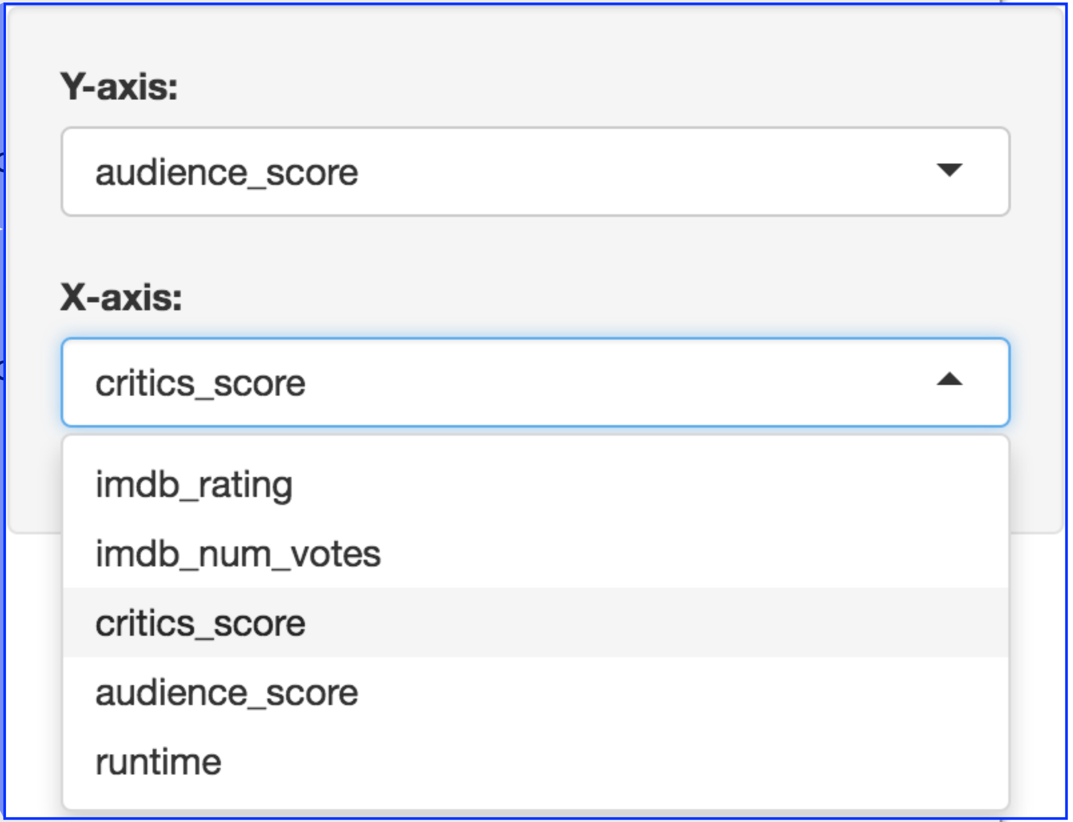
```r
# Define UI for application that plots features of movies
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a layout with a sidebar and main area

```r
# Define UI for application that plots features of movies
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a sidebar panel containing **input** controls that can in turn be passed to `sidebarLayout`

```
# Define UI for application that plots features of movies
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "c
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "c
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```



Y-axis:

audience_score

X-axis:

critics_score

```
# Define UI for application that plots features of movies
ui <- fluidPage(

  # Sidebar layout with a input and output definitions
  sidebarLayout(
    # Inputs: Select variables to plot
    sidebarPanel(
      # Select variable for y-axis
      selectInput(inputId = "y", label = "Y-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes", "critics_score", "audience_score", "runtime"),
                  selected = "critics_score")
    ),

    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "scatterplot")
    )
  )
)
```

Create a main panel containing **output** elements that get created in the server function can in turn be passed to `sidebarLayout`

Server function

```
# Define server function required to create the s
server <- function(input, output) {

  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```

Contains instructions needed to build app

```
# Define server function required to create the scatterplot
server <- function(input, output) {

  # Create the scatterplot object the plotOutput
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x,
      geom_point()
  })
}
```

Renders a **reactive** plot that is suitable for assigning to an output slot

```
# Define server function required to create the scatterplot
server <- function(input, output) {

  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```

Good ol' ggplot2 code, with **input**s from UI

Running the app

```r
# Run the application
shinyApp(ui = ui, server = server)
```

APP 1: PLOT MY DATA

- Example of what it can be used for
- *Plot changes depending on the type of variable*
- *Quick way to look at the data, QC, plan analysis*
- *Not p-hacking*

APP 2: PLOT YOUR DATA

- *Upload CSV*
  - *Could be the one in the folder, or any on your computer*
- Explore!

APP 3: PLOT AND SEE YOUR DATA

- Grow the app bit by bit
- UI changed to tabs
- Try it out:

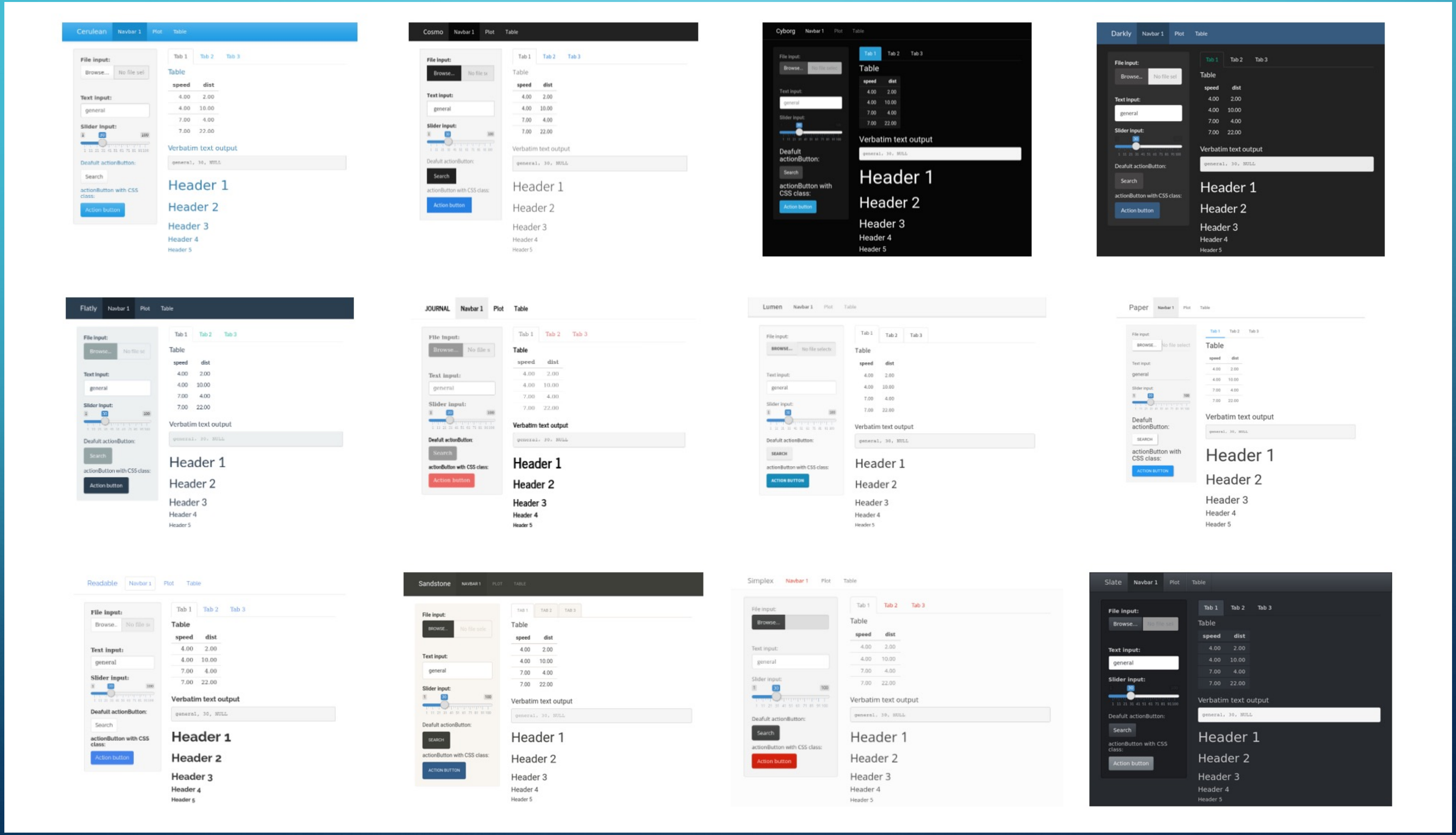
  Use renderTable() and tableOutput() to display the dataset

  Solution found in: *app3_plotAndSeeYourData_soltn*

APP 3: PLOT AND SEE YOUR DATA

```
# UI
tabPanel("Full Data - Normal",
         tableOutput("data_raw")),


# Server
output$data_raw <- renderTable(dat())
```

APP 4: PLAY WITH THEMES IN CEDAR



DEBUGGING

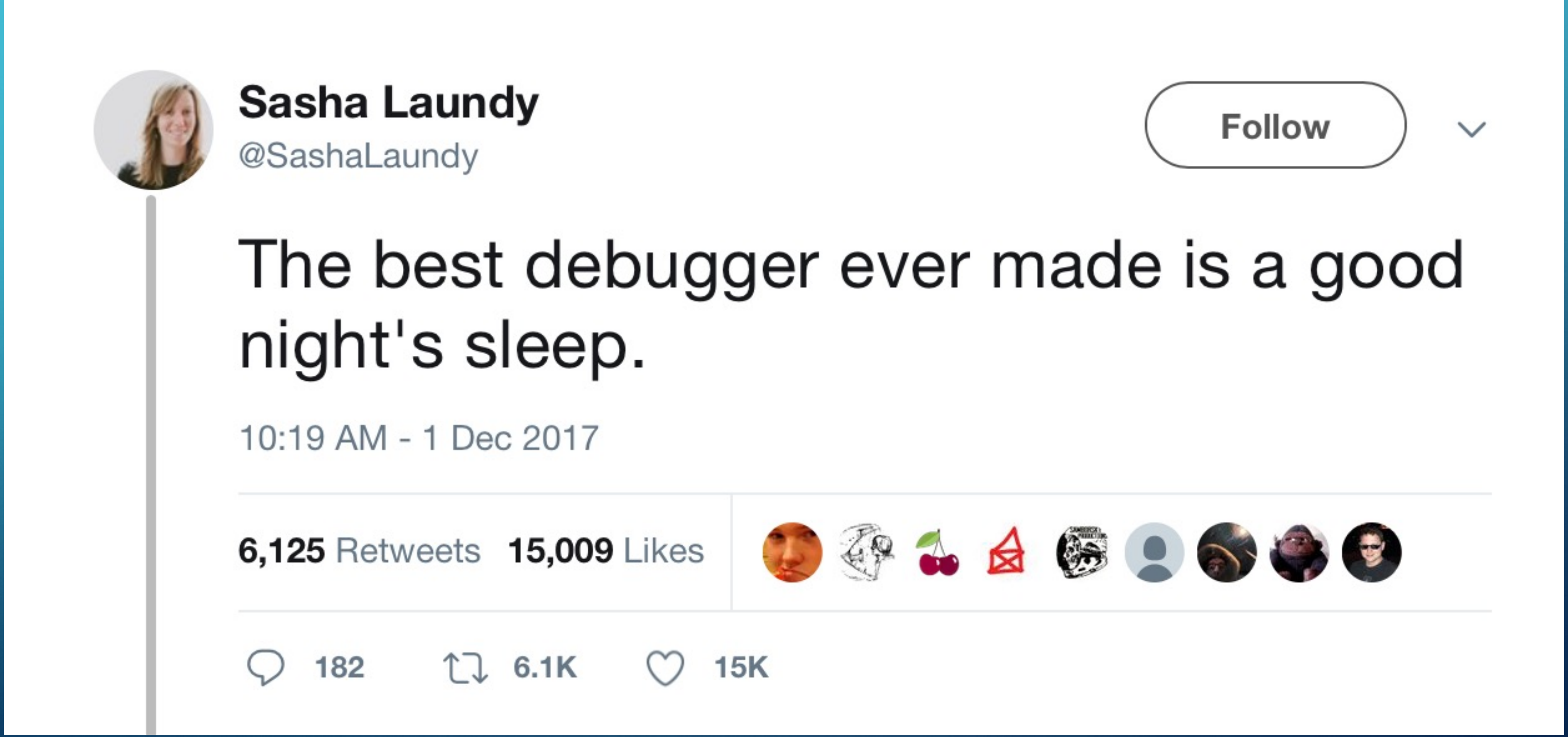
Debugging Shiny is a pain

- Many parentheses, ids, and parts
- Sometimes things don't show up
- Check that ids match

Tools:

- Use renderText() with print()
- Go inside the app using the red button in Rstudio or browse
- Version control…

DEBUGGING



Sasha Laundy
@SashaLaundy

The best debugger ever made is a good night's sleep.

10:19 AM - 1 Dec 2017

6,125 Retweets 15,009 Likes

DEBUGGING

SHINY RECAP

- Reactive programming using R -> start from the output
- Start small, build by parts
- It really is doable!

TAKE YOUR APP TO THE NEXT LEVEL

- Separate UI and server into files (add global.r file)
- Use modules as functions
  - Use the same code to plot different datasets
  - Same app, different dataset (CEDAR and PINE)
- Only plot when a button is pressed
- Cache plots that take a bit to compute
- Take snapshots of current state
- Download files
- Publish app through Rstudio or in private server
- Flexdashboards for more advanced formatting

THANKS!

Full slides available online