

COURSE OVERVIEW

& INTRODUCTION TO SHINY

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Shows R code in a script named `print.R`. Lines 29 and 30 are highlighted in yellow: `data_table <- x$data[data_ids]`.
- Console:** Shows the execution of `ggvis(diamonds, x = ~price, y = ~color)`, which triggers a debugger. The stack trace shows the call originated from `vega.R#29: data_table <- x$data[data_ids]`.
- Environment:** Shows the environment for the `as.vega.ggviz()` function, including variables `data_ids`, `data_props`, and `dynamic`.
- Plots:** A histogram titled "diamonds" showing the distribution of price. The x-axis is labeled "price" and ranges from 0 to 12,000. The y-axis is labeled "count" and ranges from 0 to 120.

Shiny from



OUTLINE

- ▶ Course Overview
 - ▶ Instructor
 - ▶ Course Schedule
- ▶ Intro to GitHub
- ▶ R Projects
- ▶ Shiny High level view
- ▶ Anatomy of a Shiny app
 - ▶ User interface
 - ▶ Server function
 - ▶ Running the app
- ▶ File Structure

Course Overview

HELLO

my name is

**GEOFFREY
ARNOLD**

Geoffrey.Lloyd.Arnold@gmail.com

ABOUT ME

- ▶ Chief Data & Analytics Officer
 - ▶ Allegheny County
- ▶ MSPPM 2015
- ▶ Heinz College

COURSE SCHEDULE

Class 1 - 7/8 - Course Overview & Introduction to GitHub & Shiny

Class 2 - 7/15 - Reactive Programming & User Interfaces

Class 3 - 7/22 - Reactive Programming Pt. 2 & Dashboards

Class 4 - 7/29 - Interactive Visualizations & Advanced Reactivity

Class 5 - 8/5 – Modules & Bookmarking

Class 6 - 8/12 - Connecting to Databases & API's

Class 7* - Leaflet & LeafletProxy

Intro to Github



*“Experience with version control is fast
becoming a requirement for all data
scientists”*

—Rebecca Vickery

WHAT IS GITHUB?

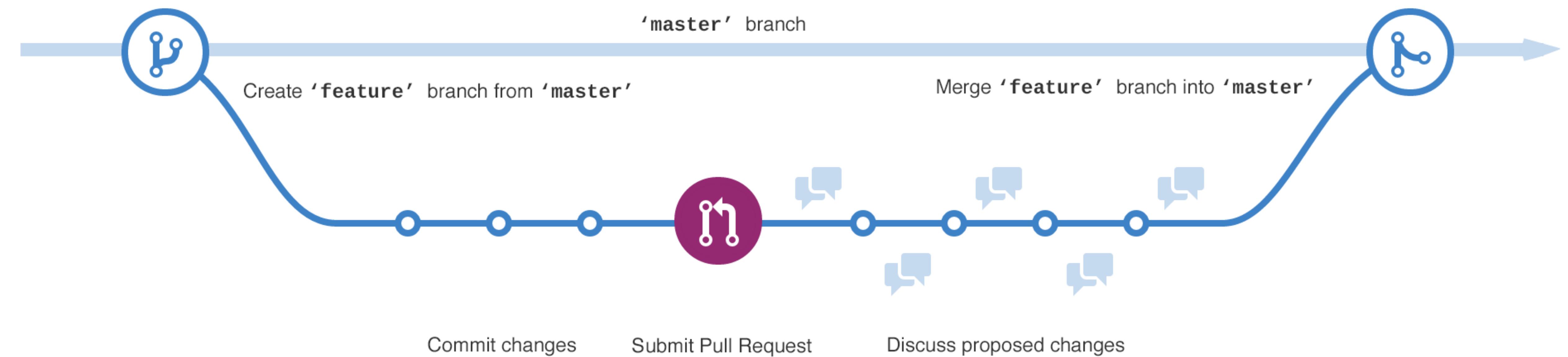
- Git is a Version Control Software
- Github stores the files for your project in a remote location and checks the differences as you change your code
- This allows you to roll back to previous versions of your project if you need to go back
- Makes sharing and collaboration much easier using the Github website

OTHER VERSION CONTROL

There are other kinds of version control software, GitLab also uses git.



We will be using Github in this course, but if in a future life, you might use these others.



Github and Projects in RStudio

Projects in RStudio

“R Projects are great.”

–Geoffrey Arnold

R PROJECTS

- ▶ So what is all this?
 - ▶ Avoid messy environment
 - ▶ Keep custom functions in check
 - ▶ Don't lose your work just because you want to do something else
- ▶ Info: <https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects>

HOW DO PROJECTS WORK?

- ▶ R typically saves your environment information in a default location (typically your Documents folder)
- ▶ When you create a project it gets its own .RData file for the project in the Directory/folder you created
- ▶ This is also the default working directory for your project, so no need to put all of the folders its in
 - ▶ Simply load objects by name if they're in the project folder



EXERCISE

- ▶ Create a “New Project”
 - ▶ Select “New Directory”
 - ▶ Select “New Project”
 - ▶ Make sure the “Create a git repository” is selected
 - ▶ Give the project any name you want
 - ▶ Click “Create Project”
 - ▶ Look at the “Git” tab

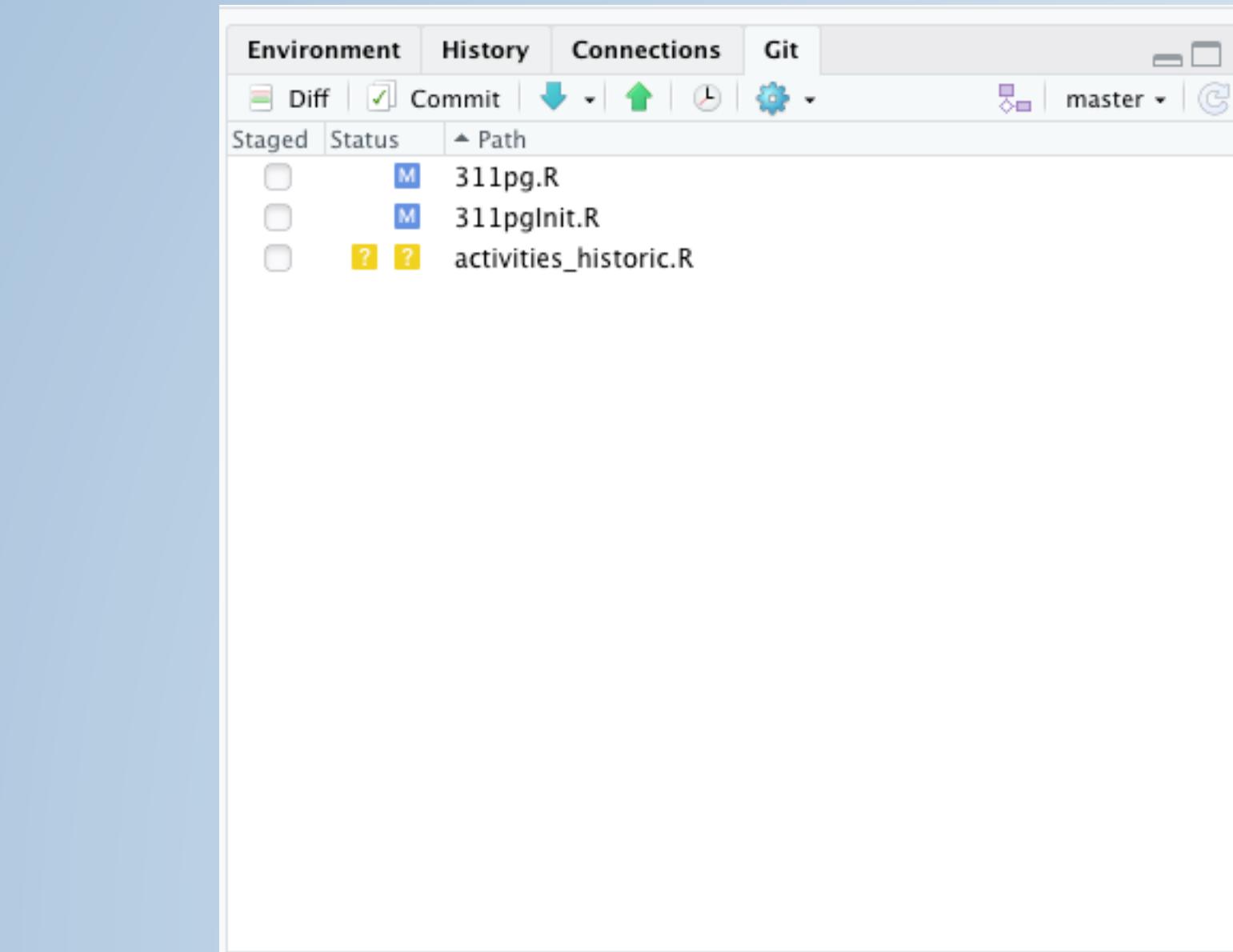
5m 00s



EXERCISE

- ▶ Click “File”
 - ▶ Click “New File”
 - ▶ Select “Shiny Web App”
 - ▶ Give the application a name and click “Create”
 - ▶ Click the “Git” tab
 - ▶ What’s changed?

5m 00s



RStudio: Review Changes

Changes History master Stage Revert Ignore Pull Push

Staged	Status	Path
<input type="checkbox"/>	M	311pg.R
<input type="checkbox"/>	M	311pgInit.R
<input type="checkbox"/>	?	activities_historic.R

Commit message

Show Staged Unstaged Context 5 line Ignore Whitespace

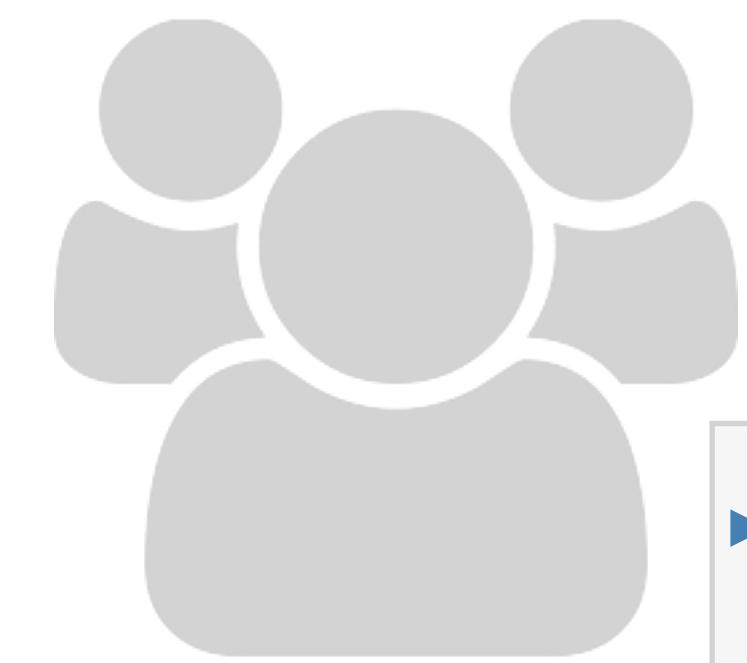
```
@@ -0,0 +1,36 @@
1 require(RPostgreSQL)
2 require(jsonlite)
3 require(dplyr)
4
5 # PG Credentials
6 pg <- fromJSON("pg_creds.json")
7 api <- fromJSON('qalert.json')$key
8
9 # Connection
10 conn <- dbConnect(drv = "PostgreSQL", host = "pg.city.pittsburgh.pa.us", dbname = "postgres", port = 5432, user = pg$pgUser,
11 password = pg$pgPW)
12 sql <- 'SELECT DISTINCT(req.id)
13 FROM qalert.requests req
14 LEFT JOIN qalert.activity act
15     ON req.id = act."requestId"
16
17 WHERE req.id IS NOT NULL AND act.requestId IS NULL'
18
19 # Write new activities
20 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
21 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
22
23 delete <- paste(unlist(unique(activity$id)), collapse = ", ") # Include Id's for new requests & non-master requests
24 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
25 del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
26
27 RPostgreSQL::dbDisconnect(conn)
28 No newline at end of file
29
30 delete <- paste(unlist(unique(activities$id)), collapse = ", ") # Include Id's for new requests & non-master requests
31 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
32 del <- RPostgreSQL::dbSendQuery(conn, sql)
33
34 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activities, row.names = FALSE, append = TRUE)
35 Sys.sleep(3)
36
37 dbDisconnect(conn)
38 No newline at end of file
```

Show Staged Unstaged Context 5 line Ignore Whitespace Stage All Discard All

```
@@ -79,8 +79,8 @@ activity <- since$activity %>%
79 79 delete <- paste(unlist(unique(since$activity$id)), collapse = ", ") # Include Id's for new requests & non-master requests
80 80 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
81 81 del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
82 82
83 83 # Write new activities
84 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
85 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
86
87 delete <- paste(unlist(unique(activities$id)), collapse = ", ") # Include Id's for new requests & non-master requests
88 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
89 del <- RPostgreSQL::dbSendQuery(conn, sql)
90
91 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activities, row.names = FALSE, append = TRUE)
92 Sys.sleep(3)
93
94 dbDisconnect(conn)
95 No newline at end of file
```

WHAT'S THE COMMAND LINE

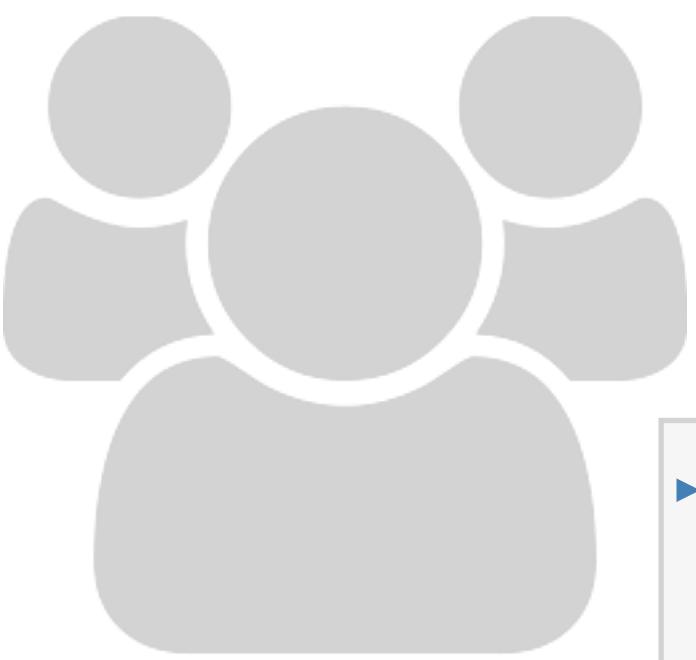
- ▶ Command or terminal git commands are how git was first used.
- ▶ You don't need to know how to do these things as either the “Git” tab in RStudio or the Github desktop program provides a GUI (gooey user interface) for you.
- ▶ However, knowing the hard way to do something never hurt anybody.



OPTIONAL EXERCISE

- ▶ <https://learngitbranching.js.org/>
- ▶ Complete Introduction Sequences
 - ▶ Git Commits
 - ▶ Branching Git
 - ▶ Merging in Git

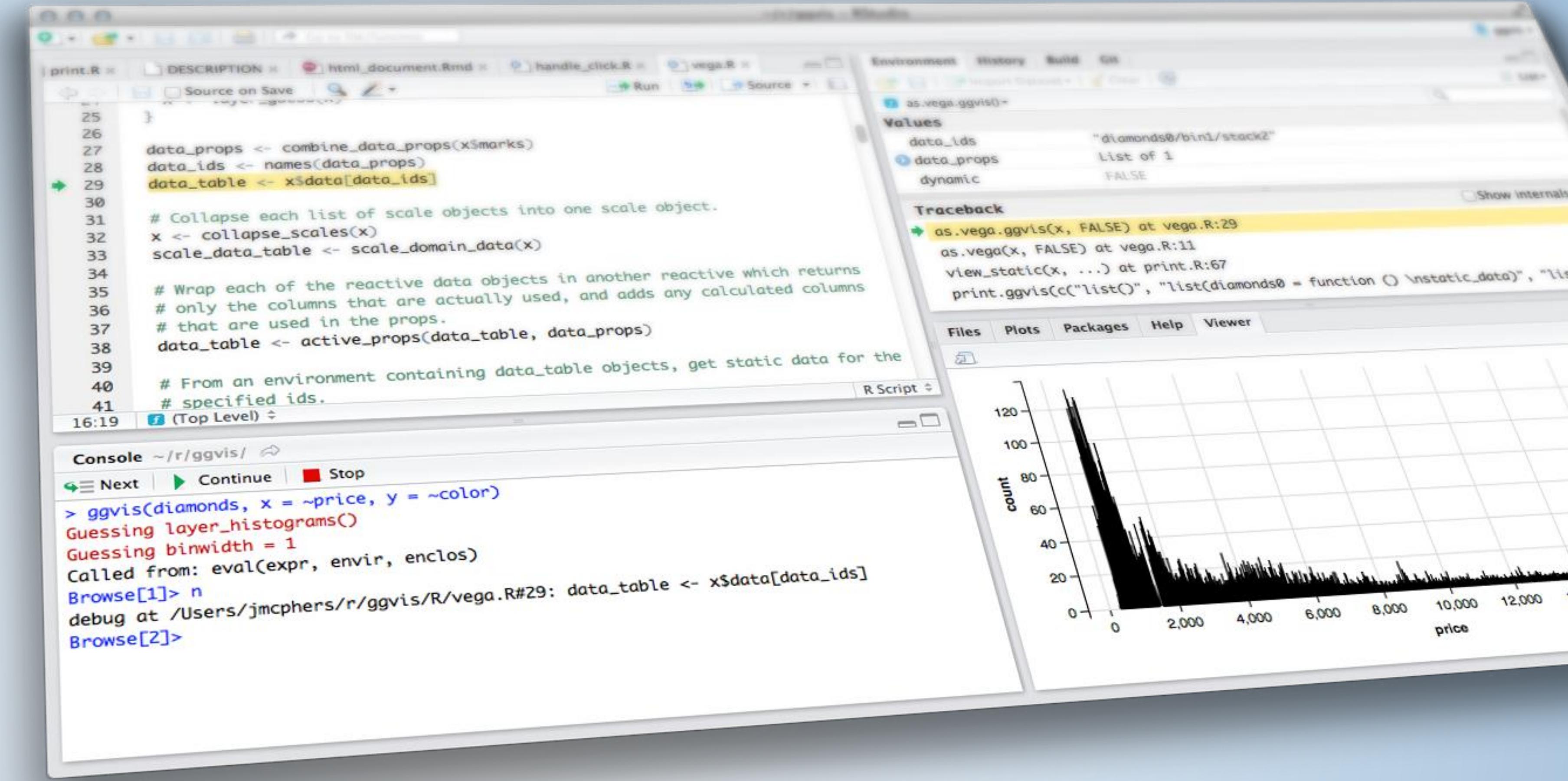
5m 00s



OPTIONAL EXERCISE

- ▶ Go to “Global Options”
 - ▶ Click “Git/SVN”
 - ▶ Ensure “Enable version control interface for Studio projects” is selected
 - ▶ Click “Create SSH Key...”
 - ▶ Click “Create”
 - ▶ Click “View public key” and copy key
 - ▶ Go to <https://github.com/settings/keys>
 - ▶ Click “New SSH key”
 - ▶ Paste key in text box and give your key a name
 - ▶ Click “Add SSH Key”
 - ▶ If you have two factor authorization turned on for GitHub (people with previous GitHub accounts may have this turned on) you will need your Personal Access Token to login later
 - ▶ Everyone else, your GitHub login and password will be important when logging in later.

CLASS BREAK



The screenshot shows the RStudio interface with the following components:

- Code Editor:** The `print.R` script is open, showing R code related to data manipulation and visualization.
- Console:** The command `> ggvis(diamonds, x = ~price, y = ~color)` is being evaluated, with a debug session active at line 29.
- Environment:** The `as.vega.ggvis()` function is highlighted in the environment pane, showing its arguments and return value.
- Viewer:** A histogram of diamond prices is displayed, showing the count of diamonds versus price.

Shiny from

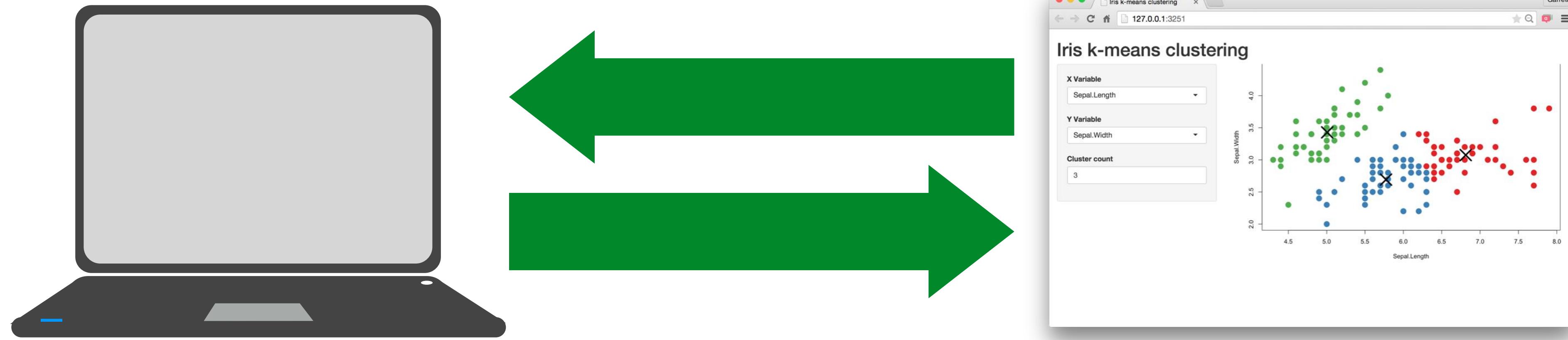


Shiny Examples

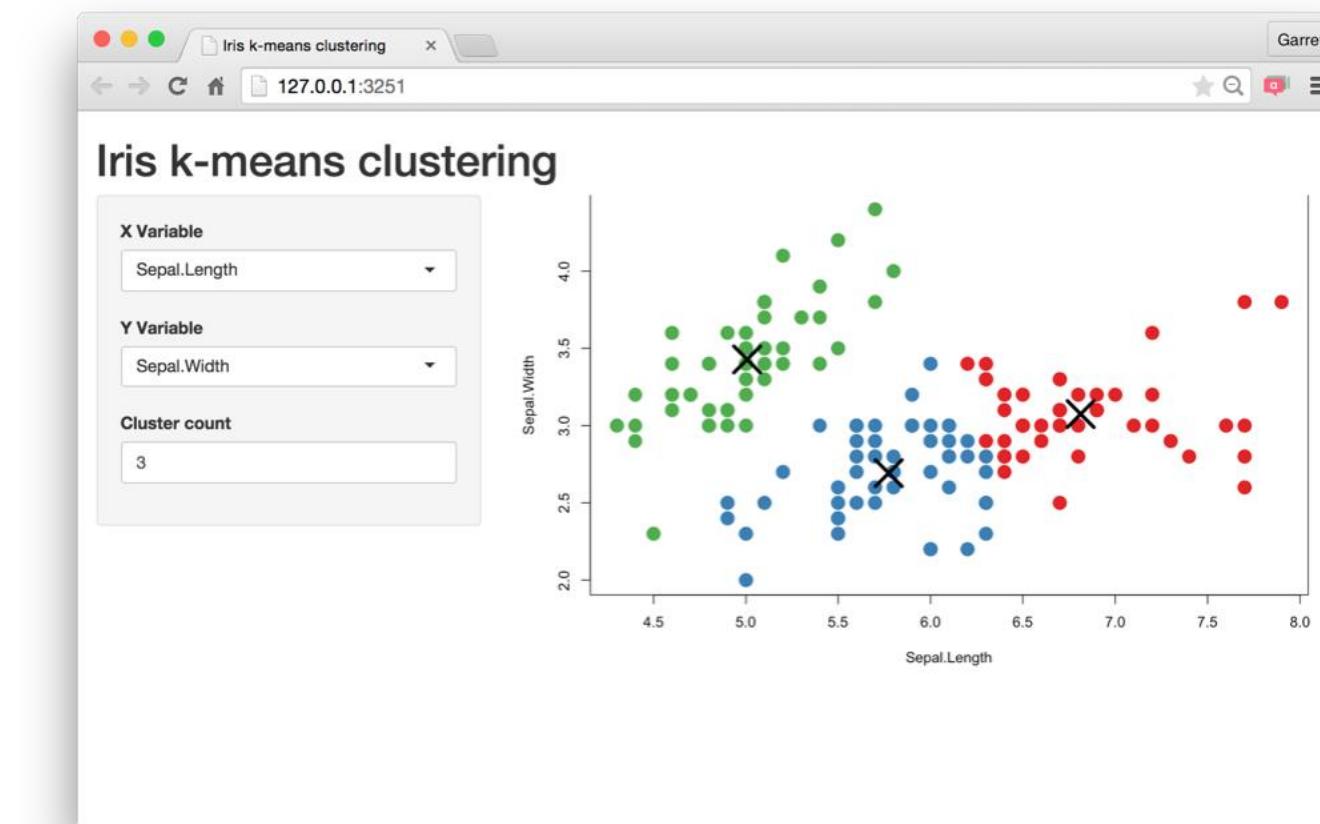
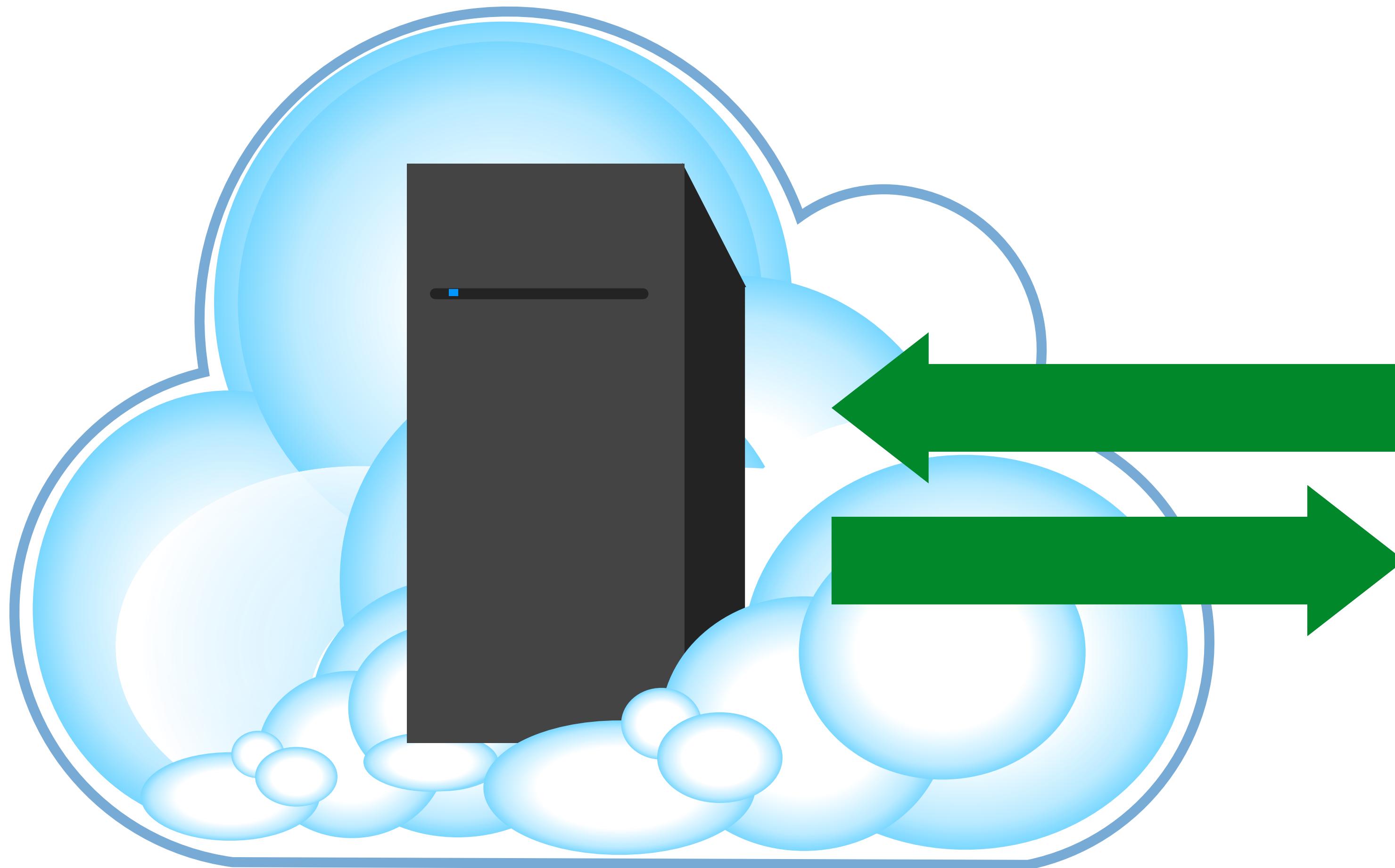
- [Port Authority Bus Tracker](#)
- [British Columbia CCISS Tool](#)
- [Commute Explorer](#)
- [Covid-19 Tracker](#)

Shiny High level
view

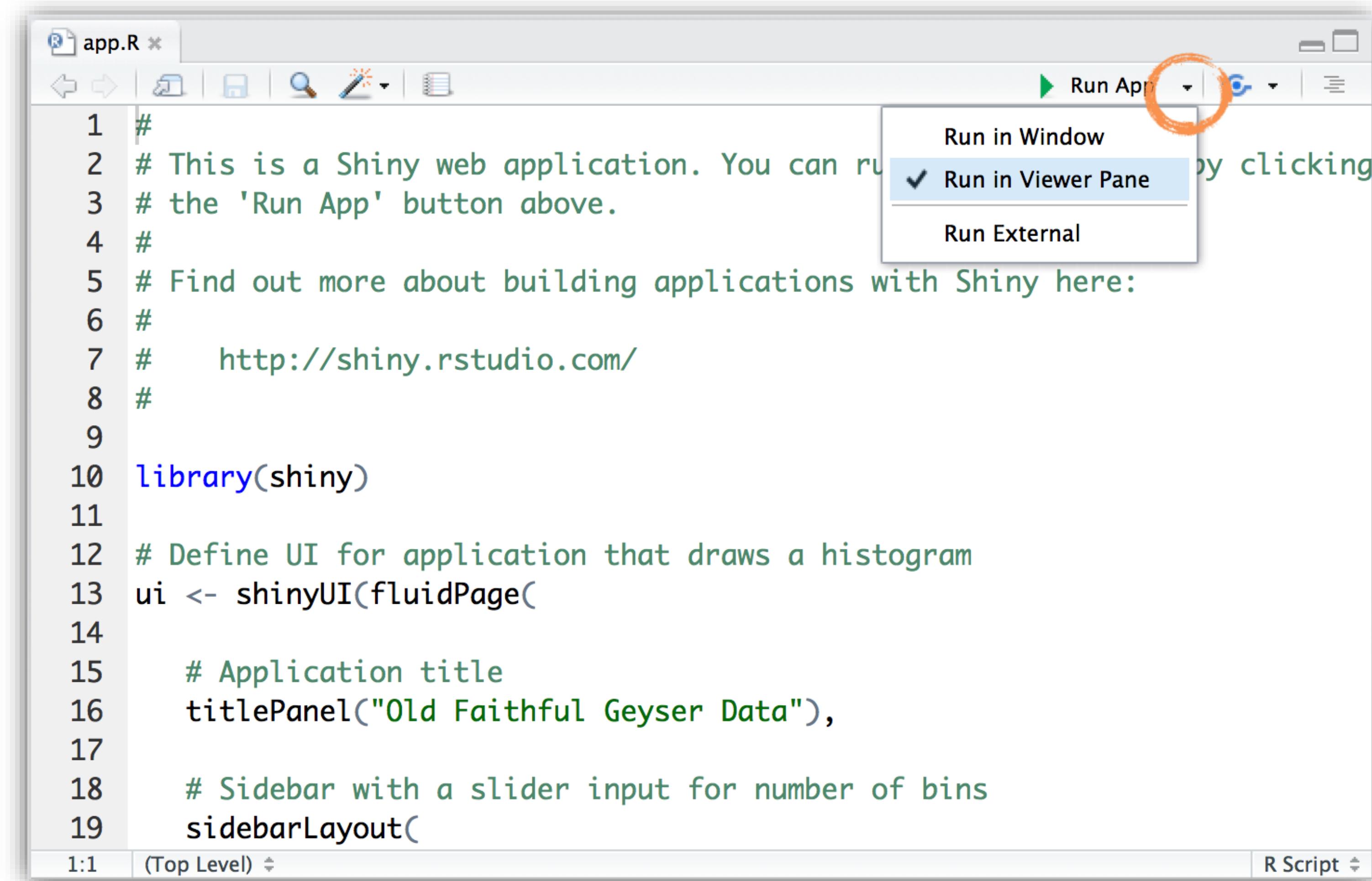
Every Shiny app is maintained by a computer running R



Every Shiny app is maintained by a computer running R



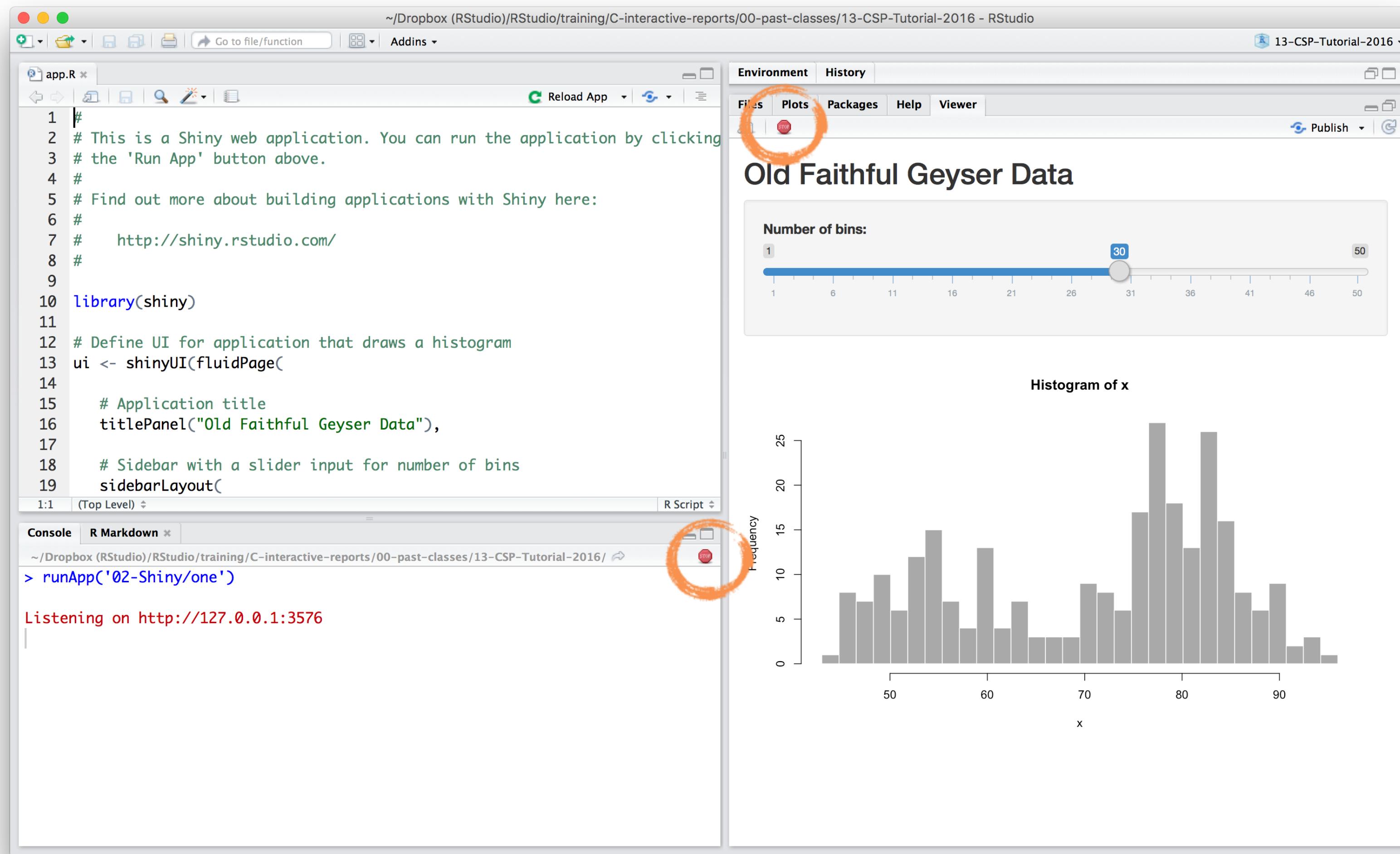
Change display



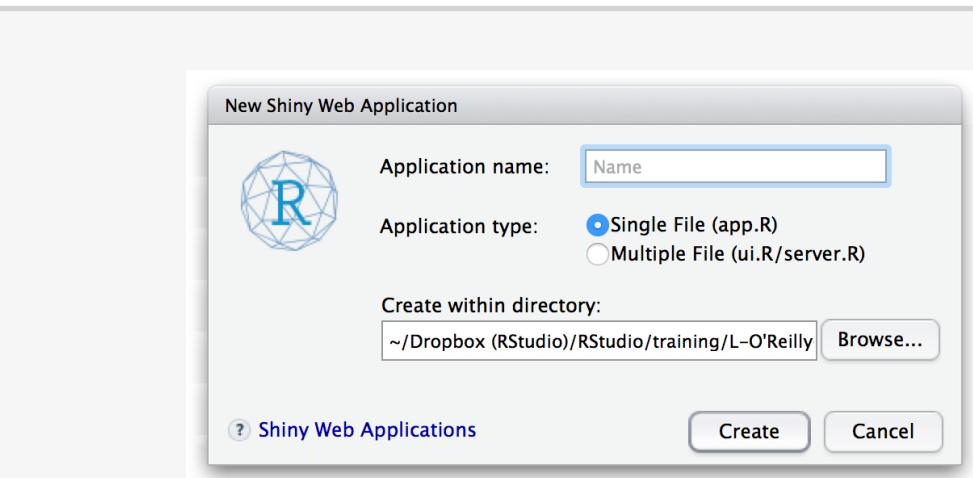
The screenshot shows the RStudio interface with an R script file named "app.R" open. The code is a basic Shiny application template. A context menu is displayed over the "Run App" button in the toolbar, with the "Run in Viewer Pane" option selected. The "Run in Window" option is also visible.

```
1 #  
2 # This is a Shiny web application. You can run  
3 # the 'Run App' button above.  
4 #  
5 # Find out more about building applications with Shiny here:  
6 #  
7 #     http://shiny.rstudio.com/  
8 #  
9 #  
10 library(shiny)  
11  
12 # Define UI for application that draws a histogram  
13 ui <- shinyUI(fluidPage(  
14  
15     # Application title  
16     titlePanel("Old Faithful Geyser Data"),  
17  
18     # Sidebar with a slider input for number of bins  
19     sidebarLayout(
```

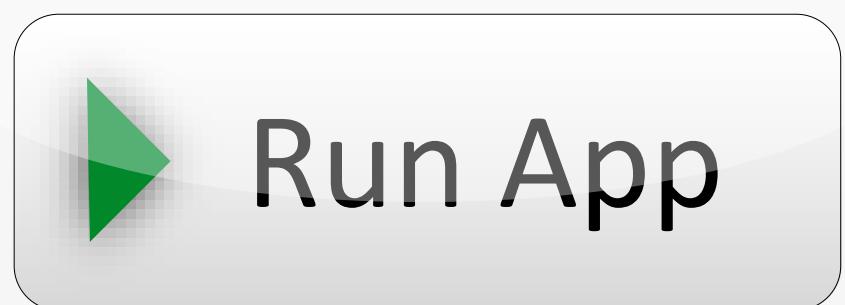
Close an app



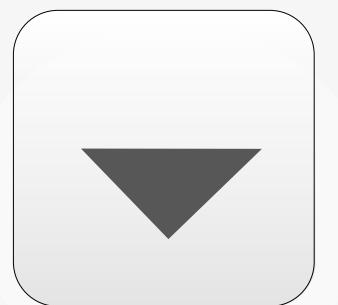
EXERCISE



Open a new Shiny app with
File ➔ New File ➔ Shiny Web App...



Launch the app by opening app.R and
clicking **Run App**



Close app by clicking the stop sign icon

Select view mode in the drop down
menu next to Run App

3m 00s

Anatomy of a Shiny app

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

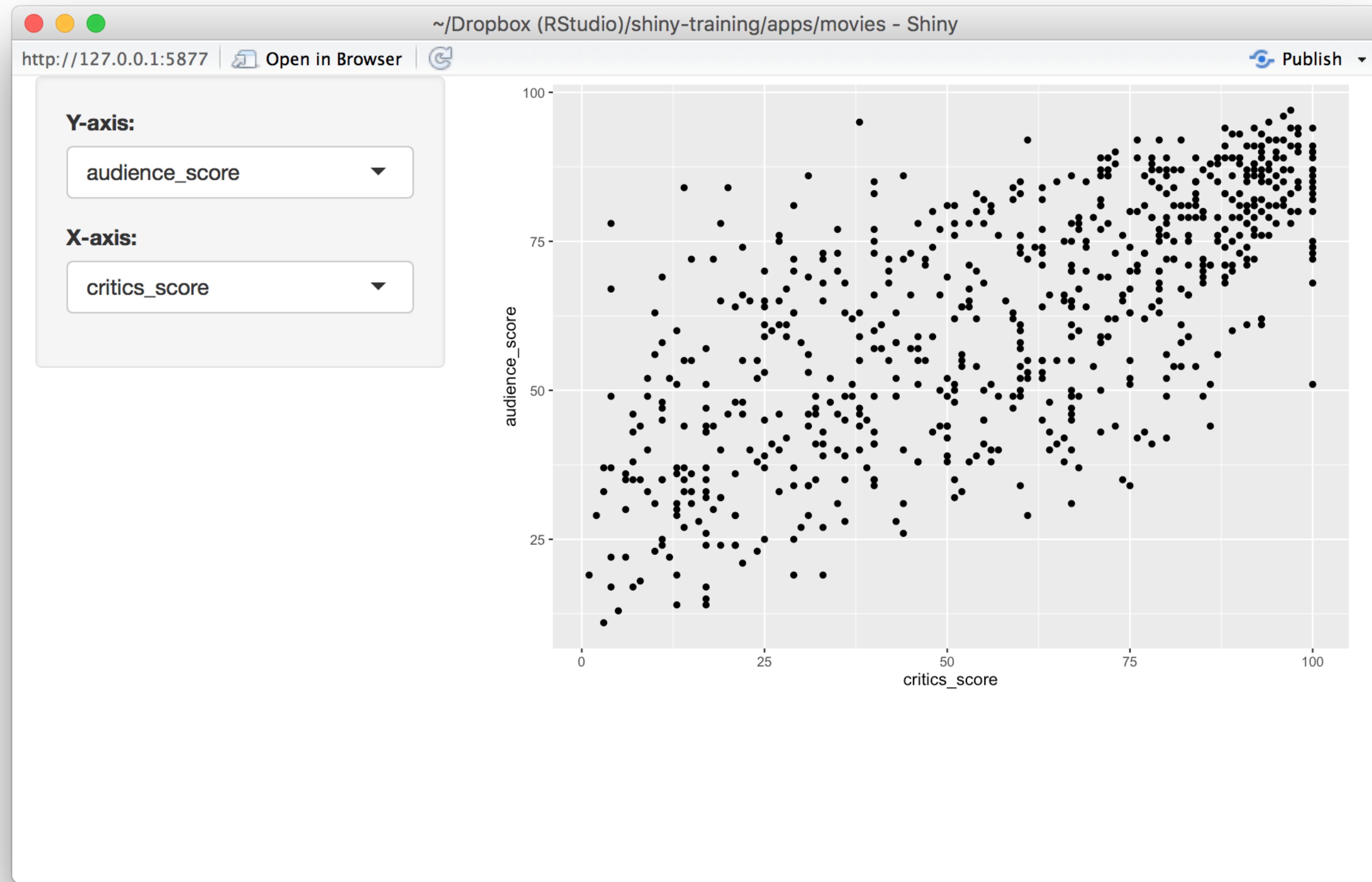


Let's build a simple movie browser app!



movies.Rdata

Data from IMDB and Rotten Tomatoes on random sample of 651 movies released in the US between 1970 and 2014

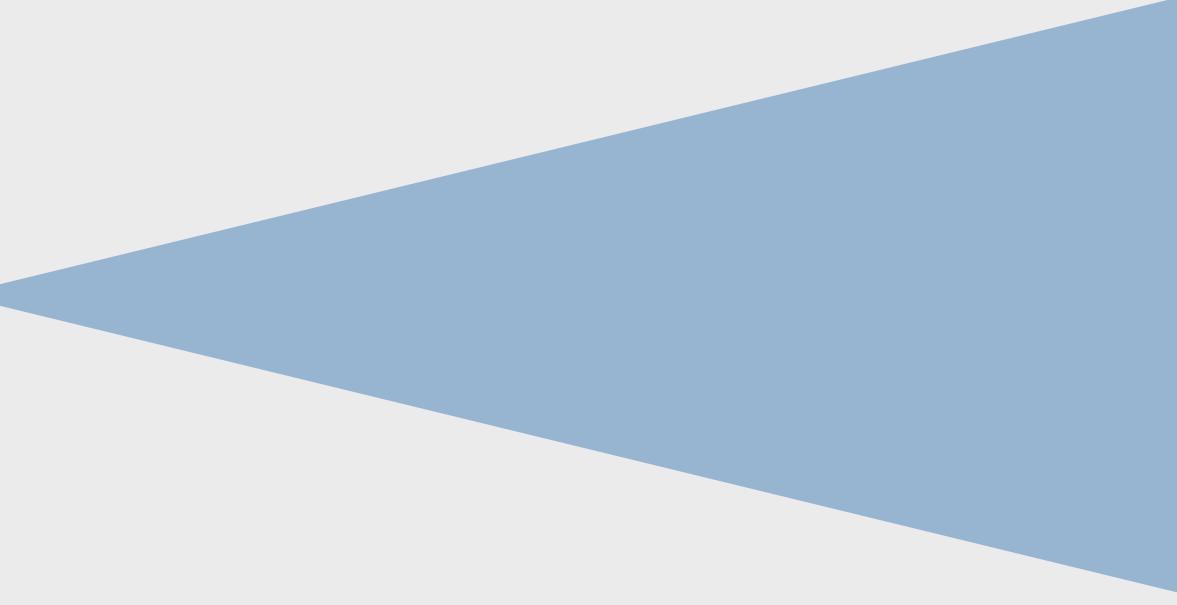


APP TEMPLATE

```
library(shiny)  
library(ggplot2)  
load("movies.Rdata")  
ui <- fluidPage()
```

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

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



Dataset used for this app

User interface

```
# 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")
    )
  )
)
```

```
# 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

```
# 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

```
# 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", "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")
  )
)
```

Y-axis:

audience_score

X-axis:

critics_score

imdb_rating

imdb_num_votes

critics_score

audience_score

runtime

```
# 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 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()
  })
}
```

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

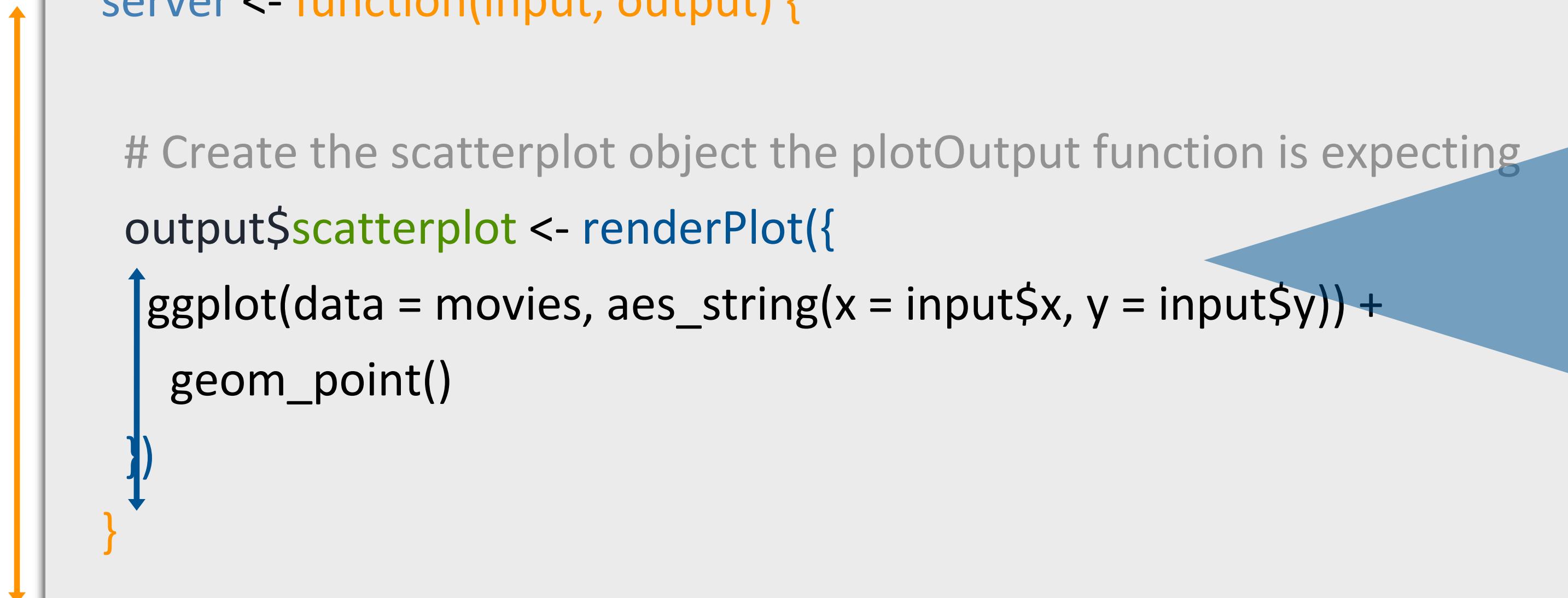
  # Create the scatterplot object the plotOutput function is expecting
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
      geom_point()
  })
}
```



Contains instructions
needed to build app

```
# 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()
  })
}
```



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 **inputs** from UI

Running the app

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

DEMO



Putting it all together...

movies_01.R



EXERCISE

- ▶ Add new select menu to color the points by
 - ▶ `inputId = "z"`
 - ▶ `label = "Color by:"`
 - ▶ `choices = c("title_type", "genre", "mpaa_rating", "critics_rating", "audience_rating")`
 - ▶ `selected = "mpaa_rating"`
- ▶ Use this variable in the aesthetics of the `ggplot` function as the `color` argument to color the points by
- ▶ Run the app in the Viewer Pane
- ▶ Compare your code / output with the person sitting next to / nearby you

5m 00s

A large, light gray checkmark icon inside a circle, indicating a correct answer or solution.

SOLUTION

Solution to the previous exercise

`movies_02.R`

INPUTS

Interactive Web Apps with shiny Cheat Sheet
learn more at shiny.rstudio.com

R Studio
Basics

A Shiny app is a web page (`UI`) connected to a computer running a live R session (`Server`).
Users can manipulate the UI, which will cause the server to update the UI displays (by running R code).
Begin writing a new app with this template. Preview the app by running the code at the command line.

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {
  shinyApp(ui = ui, server = server)
}
```

- `ui` needs R functions that assemble an HTML user interface for your app.
- `server` is a function with instructions on how to build and return the outputs defined in the UI.
- `shinyApp` combines `ui` and `server` into a functioning app. Wrap with `runApp()` if calling from a sourced script or inside a function.

Share your app

1. Create a free or professional account at <http://shinyapps.io>
2. Click the **Publish** icon in the RStudio IDE (if >0.99) or run `rsconnect::deployApp(<path to directory>)`

Build or purchase your own Shiny Server at www.rstudio.com/products/shiny-server/

RSStudio® is a trademark of RStudio, Inc. • CC-BY-RStudio • info@rstudio.com • 844-448-3222 • studio.com

More cheat sheets at <http://www.rstudio.com/resources/cheatsheets/>

Learn more at shiny.rstudio.com/tutorial/shiny-0.12.0/ • (updated: 01/26)

Action `actionButton(inputId, label, icon, ...)`

Link `actionLink(inputId, label, icon, ...)`

Choice 1
 Choice 2
 Choice 3

Check me

checkboxGroupInput(inputId, label, choices, selected, inline)

checkboxInput(inputId, label, value)

dateInput(inputId, label, value, min, max, format, startview, weekstart, language)

dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)

fileInput(inputId, label, multiple, accept)

numericInput(inputId, label, value, min, max, step)

.....

Choice A
 Choice B
 Choice C

selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also selectizeInput())

radioButtons(inputId, label, choices, selected, inline)

sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)

submitButton(text, icon)
 (Prevents reactions across entire app)

textInput(inputId, label, value)





EXERCISE

- ▶ Add new input variable to control the alpha level of the points
 - ▶ This should be a sliderInput
 - ▶ See shiny.rstudio.com/reference/shiny/latest/ for help
 - ▶ Values should range from 0 to 1
 - ▶ Set a default value that looks good
- ▶ Use this variable in the geom of the ggplot function as the alpha argument
- ▶ Run the app in a new window
- ▶ Compare your code / output with the person sitting next to / nearby you

5m 00s

A large, light gray checkmark icon inside a circle, indicating a correct answer or solution.

SOLUTION

Solution to the previous exercise

`movies_03.R`

OUTPUTS

Interactive Web Apps with shiny Cheat Sheet
learn more at shiny.rstudio.com

R Studio

Basics

A Shiny app is a web page (UI) connected to a computer running an R session (Server).

Users can manipulate the UI, which will cause the server to update the UI displays (by running R code).

App template

Begin writing a new app with this template. Preview the app by running the code at the command line.

```
library(shiny)
ui <- fluidPage(
  numericInput("n", "Number of observations", value = 25),
  plotOutput("hist")
)
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
shinyApp(ui = ui, server = server)
```

• ui - nested R functions that assemble an HTML user interface for your app

• server - a function that takes the R objects displayed in the UI and manipulates them to produce new ones

• shinyApp - combines ui and server into a functioning app. Wrap with runApp() if calling from a source script or inside a function

Shinyapps

The easiest way to share your app is to host it on shinyapps.io, a cloud based service from RStudio.

1. Create a free or professional account at <http://shinyapps.io>
2. Click the Publish icon in the RStudio IDE (>0.99) or run: rconnect::deployApp("~/path to directory")

Build or purchase your own Shiny Server at www.rstudio.com/products/shiny-server/

RSudio® is a trademark of RStudio, Inc. • CC-BY RStudio • info@rstudio.com • 844-448-3212 • rstudio.com • More cheat sheets at <http://rstudio.com/resources/cheatsheets/> • Learn more at shiny.rstudio.com/tutorial • shiny 0.12.0 • Updated: 01/16

Building an App - Complete the template by adding arguments to fluidPage() and a body to the server function.

Add inputs to the UI with `input()` functions.
Add outputs with `output()` functions.
Tell server how to handle outputs with R in the server function. To do this:

1. Refer to outputs with `output$<id>`
2. Refer to inputs with `input$<id>`
3. Wrap code in a `render()` function before saving to output

Save your template as `app.R`. Alternatively, split your template into two files named `ui.R` and `server.R`.

ui.R

```
library(shiny)
ui <- fluidPage(
  numericInput("n", "Number of observations", value = 25),
  plotOutput("hist")
)
```

server.R

```
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}
```

shinyApp(ui = ui, server = server)

Inputs - collect values from the user

Access the current value of an input object with `$input$<id>`. Input values are reactive.

Action

- actionButton(inputId, label, icon, ...)
- actionLink(inputId, label, icon, ...)

Link

- choice 1
- choice 2
- choice 3

checkboxInput(inputId, label, value)

checkboxGroupInput(inputId, label, choices, selected, inline)

dateInput(inputId, label, value, min, max, format, startview, weekstart, language)

dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)

fileInput(inputId, label, multiple, accept)

numericInput(inputId, label, value, min, max, step)

passwordInput(inputId, label, value)

radioButtons(inputId, label, choices, selected, inline)

selectInput(inputId, label, choices, selected, multiple, selectize, width, size) (also `selectizeInput()`)

selectizeInput(inputId, label, choices, selected, multiple, selectize, width, size)

sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, app, post)

submitButton(inputId, label)

textInput(inputId, label, value)

uiOutput(outputId, inline, container, ...)

htmlOutput(outputId, inline, container, ...)

Choose file

Launch apps with runApp(spath to directory?)

data.table, Frame: 3 obs. of 2 variables

\\$ Sepal.Length: num 5.1 4.9 4.7

\\$ Sepal.Width : num 3.5 3.2 3.2

Outputs - render() and output() functions work together to add R output to the UI

• `DT::renderDataTable(expr, options, callback, escape, env, quoted)`

• `dataTableOutput(outputId, icon, ...)`

• `renderImage(expr, env, quoted, deleteFile)`

• `imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)`

• `renderPlot(expr, width, height, res, ..., env, quoted, func)`

• `plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)`

• `renderPrint(expr, env, quoted, func, width)`

• `printOutput(outputId, width, height, env, quoted)`

• `renderTable(expr, ..., env, quoted, func)`

• `tableOutput(outputId)`

• `renderText(expr, env, quoted, func)`

• `textOutput(outputId, container, inline)`

• `renderUI(expr, env, quoted, func)`

• `uiOutput(outputId, inline, container, ...)`

• `htmlOutput(outputId, inline, container, ...)`

• `chooseColorInput(inputId, label, ...)`

• `chooseDateInput(inputId, label, ...)`

• `chooseFileInput(inputId, label, ...)`

• `chooseInput(inputId, label, ...)`

• `chooseNumberInput(inputId, label, ...)`

• `chooseRadioInput(inputId, label, ...)`

• `chooseSelectizeInput(inputId, label, ...)`

• `chooseSliderInput(inputId, label, ...)`

• `chooseSubmitInput(inputId, label, ...)`

• `chooseTextInput(inputId, label, ...)`

• `dateRangeInput(inputId, label, ...)`

• `fileInput(inputId, label, ...)`

• `passwordInput(inputId, label, ...)`

• `radioButtons(inputId, label, ...)`

• `selectInput(inputId, label, ...)`

• `selectizeInput(inputId, label, ...)`

• `sliderInput(inputId, label, ...)`

• `textInput(inputId, label, ...)`

• `uiOutput(outputId, ...)`

• `htmlOutput(outputId, ...)`

• `chooseColorOutput(outputId, ...)`

• `chooseDateOutput(outputId, ...)`

• `chooseFileOutput(outputId, ...)`

• `chooseInputOutput(outputId, ...)`

• `chooseNumberOutput(outputId, ...)`

• `chooseRadioOutput(outputId, ...)`

• `chooseSelectizeOutput(outputId, ...)`

• `chooseSliderOutput(outputId, ...)`

• `chooseSubmitOutput(outputId, ...)`

• `chooseTextOutput(outputId, ...)`

• `dateRangeOutput(outputId, ...)`

• `fileOutput(outputId, ...)`

• `passwordOutput(outputId, ...)`

• `radioButtonOutput(outputId, ...)`

• `selectInputOutput(outputId, ...)`

• `selectizeOutput(outputId, ...)`

• `sliderOutput(outputId, ...)`

• `textOutput(outputId, ...)`

• `uiOutputOutput(outputId, ...)`

• `htmlOutputOutput(outputId, ...)`

• `chooseColorOutputOutput(outputId, ...)`

• `chooseDateOutputOutput(outputId, ...)`

• `chooseFileOutputOutput(outputId, ...)`

• `chooseInputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutput(outputId, ...)`

• `chooseRadioOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutput(outputId, ...)`

• `chooseSliderOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutput(outputId, ...)`

• `chooseTextOutputOutput(outputId, ...)`

• `dateRangeOutputOutput(outputId, ...)`

• `fileOutputOutput(outputId, ...)`

• `passwordOutputOutput(outputId, ...)`

• `radioButtonOutputOutput(outputId, ...)`

• `selectInputOutputOutput(outputId, ...)`

• `selectizeOutputOutput(outputId, ...)`

• `sliderOutputOutput(outputId, ...)`

• `textOutputOutput(outputId, ...)`

• `uiOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `sliderOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `textOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `uiOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `htmlOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseColorOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseDateOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseFileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseNumberOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseRadioOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSelectizeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSliderOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseSubmitOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `chooseTextOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `dateRangeOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `fileOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `passwordOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `radioButtonOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectInputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutputOutput(outputId, ...)`

• `selectizeOutputOutputOutput`



EXERCISE

- ▶ Add a checkbox input to decide whether the data plotted should be shown in a data table
 - ▶ This should be a checkboxInput (see shiny.rstudio.com/reference/shiny/latest/ for help)
- ▶ Create a new output item using DT::renderDataTable, an if statement to check if the box is checked, and DT::datatable
 - ▶ Show first seven columns of movies data, show 10 rows at a time, and hide row names, e.g.
 - ▶ data = movies[, 1:7]
 - ▶ options = list(pageLength = 10)
 - ▶ rownames = FALSE
- ▶ Add a dataTableOutput to the main panel
- ▶ Run the app in a new Window, check and uncheck the box to test functionality
- ▶ Compare your code / output with the person sitting next to / nearby you
- ▶ **Optional:** If you finish early, move on to the next exercise

5m 00s

A large, light gray checkmark icon inside a circle, indicating a correct answer or solution.

SOLUTION

Solution to the previous exercise

`movies_04.R`



EXERCISE

Optional: If you finish the previous exercise early

- ▶ Add a title to your app with titlePanel, which goes before the sidebarLayout
- ▶ Prettify the variable names shown as input choices. *Hint:*
 - ▶ `choices = c("IMDB rating" = "imdb_rating", ...)`
- ▶ Prettify the axis and legend labels of your plot. *Hint:* You might use
 - ▶ `str_replace_all` from the `stringr` package
 - ▶ `toTitleCase` from the `tools` package

5m 00s

A large, light gray checkmark icon inside a circle, indicating a correct answer or solution.

SOLUTION

Solution to the previous exercise

`movies_05.R`

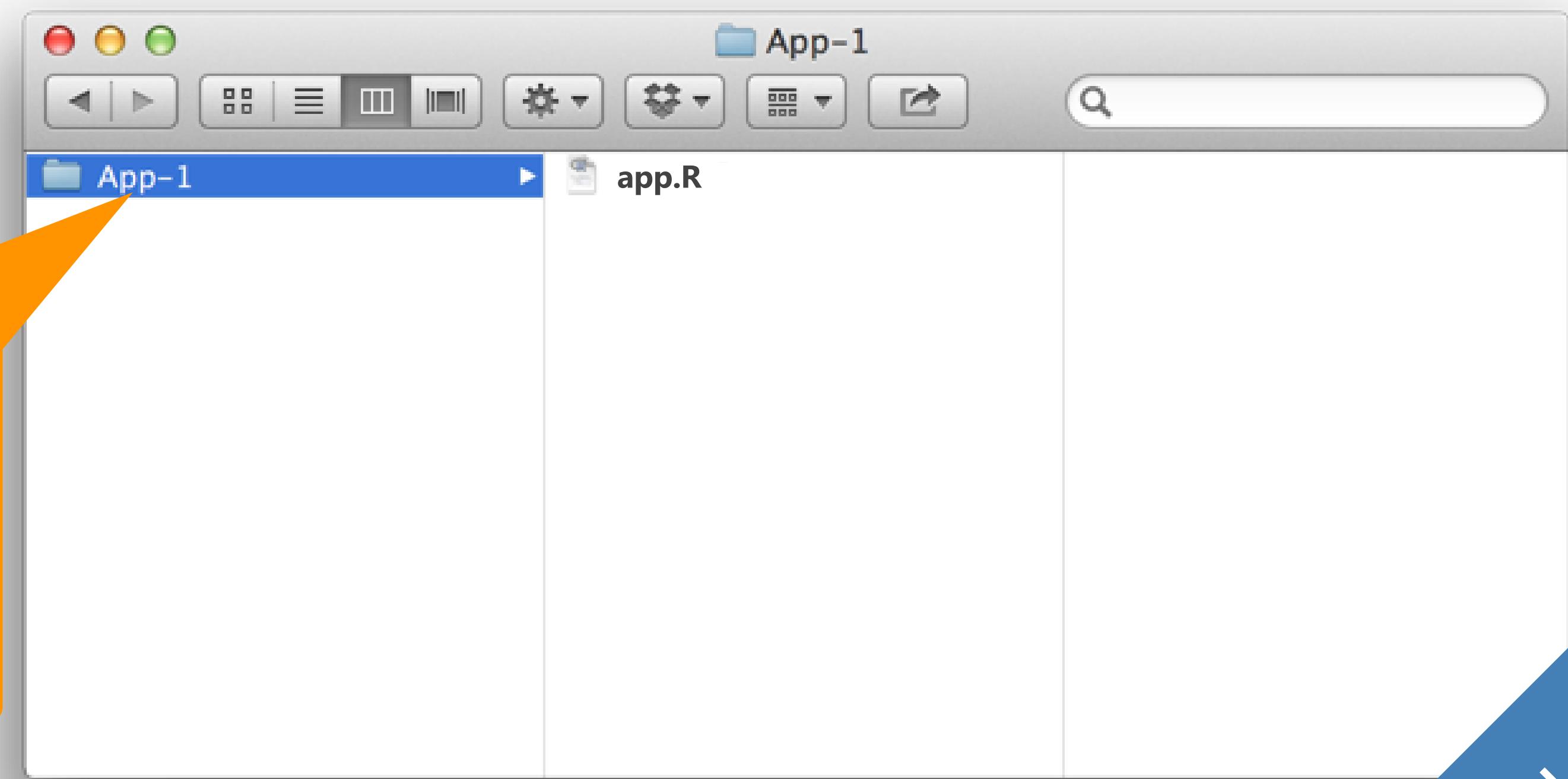
File structure

SAVING YOUR SINGLE FILE APP

One directory with every file the app needs:

- ▶ `app.R` (your script which ends with a call to `shinyApp()`)
- ▶ datasets, images, css, helper scripts, etc.

We will focus on
the single file
format throughout
the course

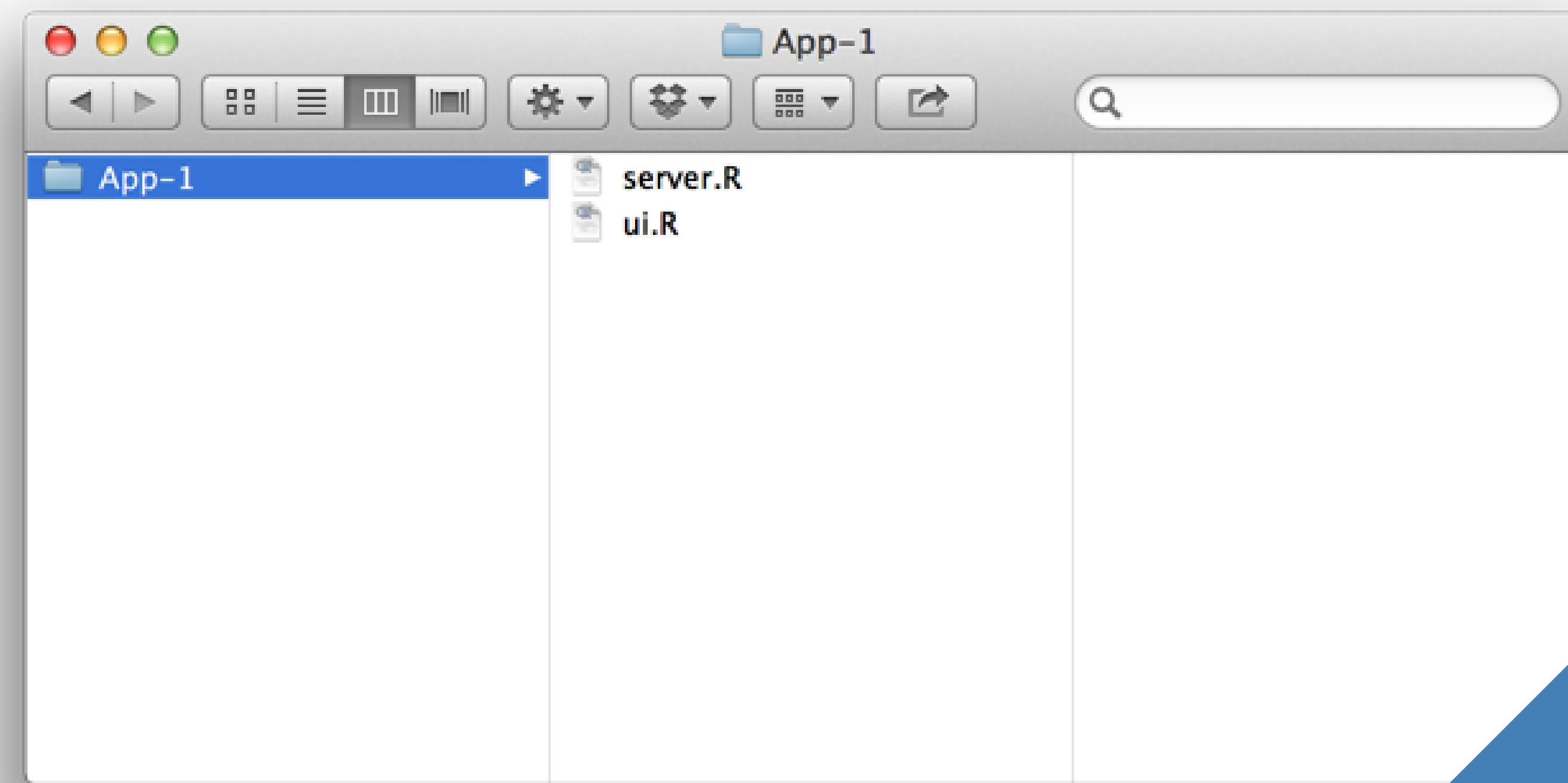


You must use this exact name
(app.R) for deploying the
app to shinyapps.io

SAVING YOUR MULTIPLE FILE APP

One directory with every file the app needs:

- ▶ ui.R and server.R
- ▶ datasets, images, css, helper scripts, etc.



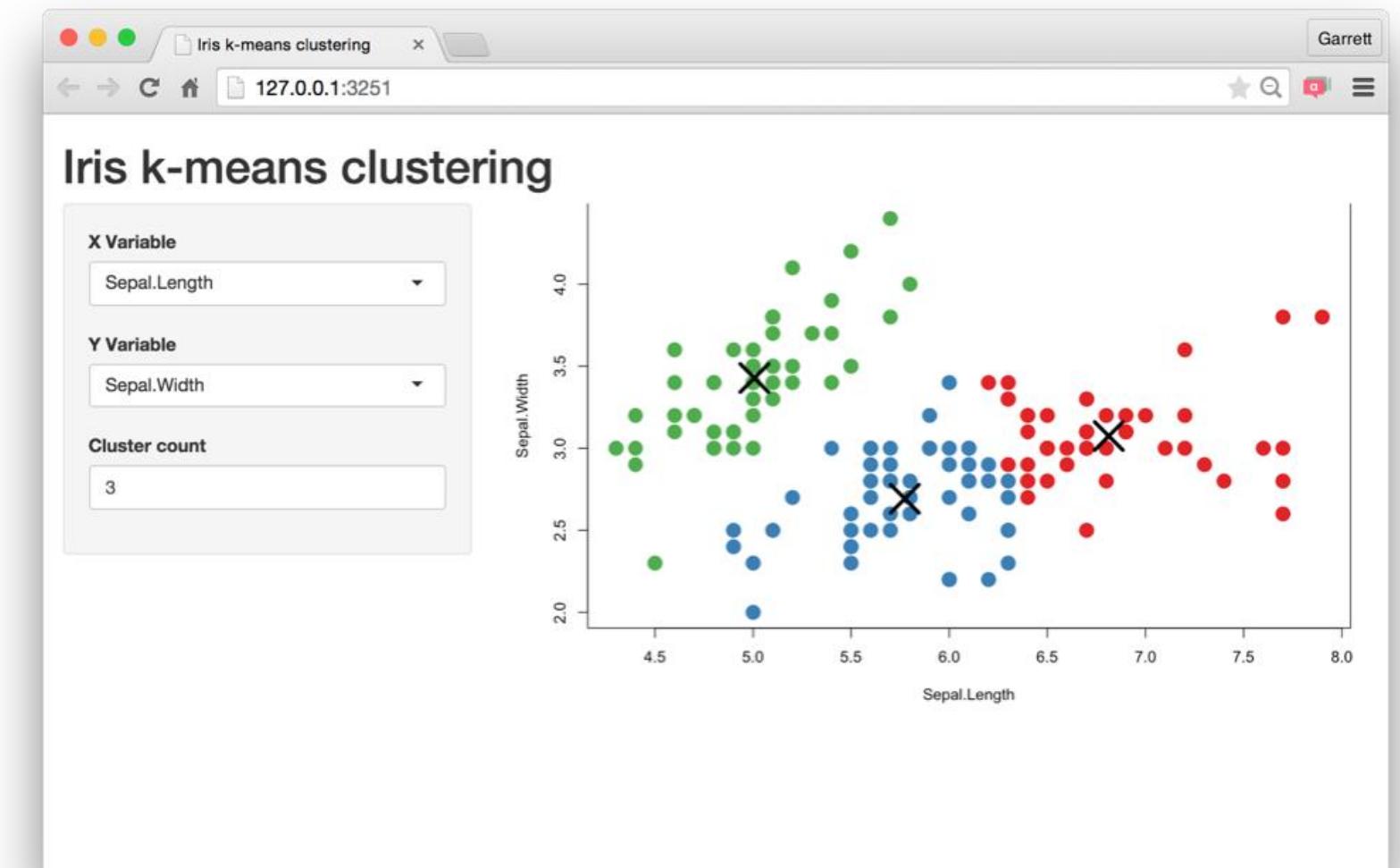
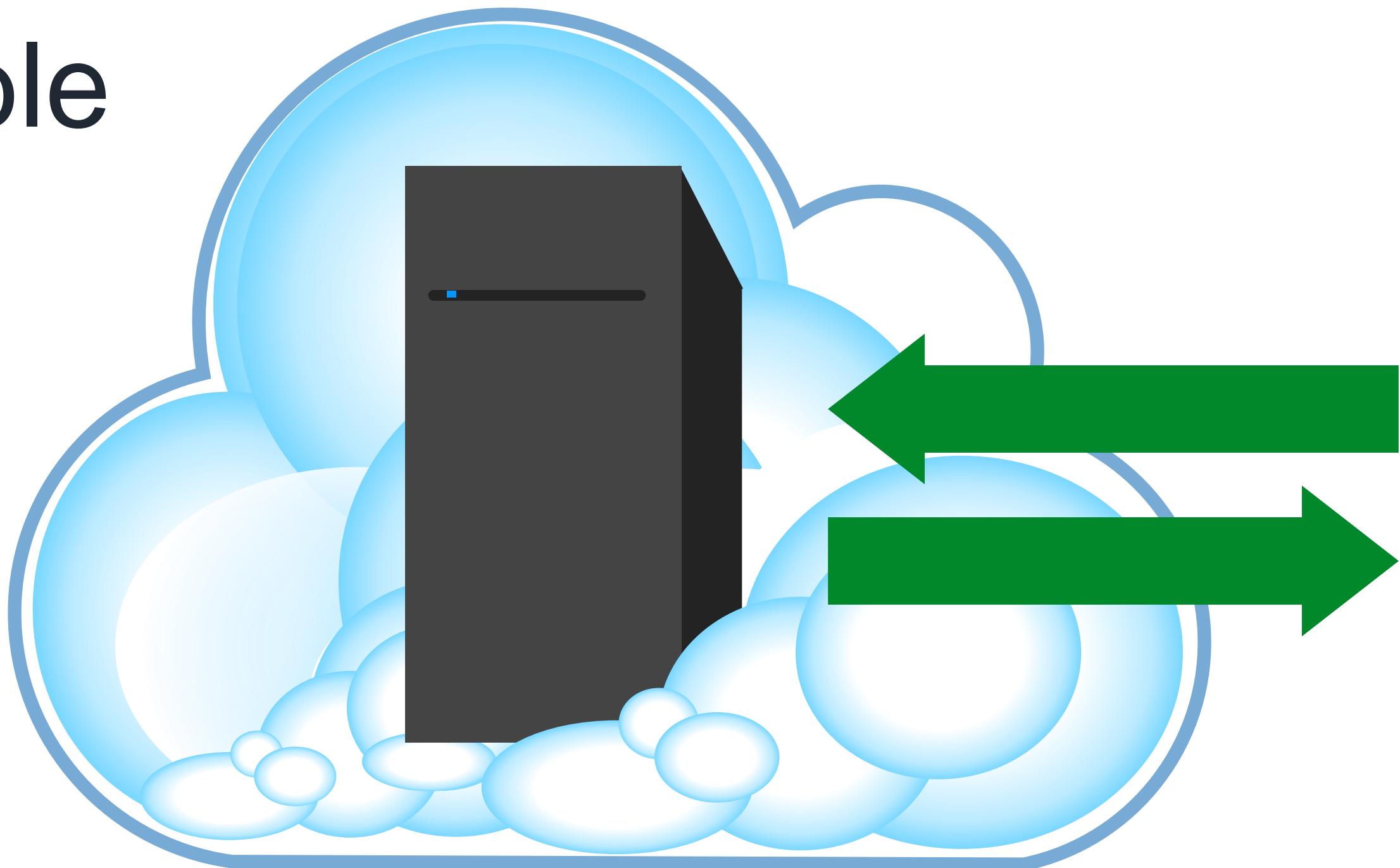
You must use these
exact names

Sharing your app

shinyapps.io

A server maintained by RStudio

- ▶ easy to use
- ▶ secure
- ▶ scalable



HASSE-FREE CLOUD HOSTING FOR SHINY

shinyapps.io by Posit

[Home](#) [Features](#) [Pricing](#) [Support](#) [Log In](#)

Share your Shiny Applications Online

Deploy your Shiny applications on the Web in minutes

[Sign Up](#)



Easy To Use



Secure

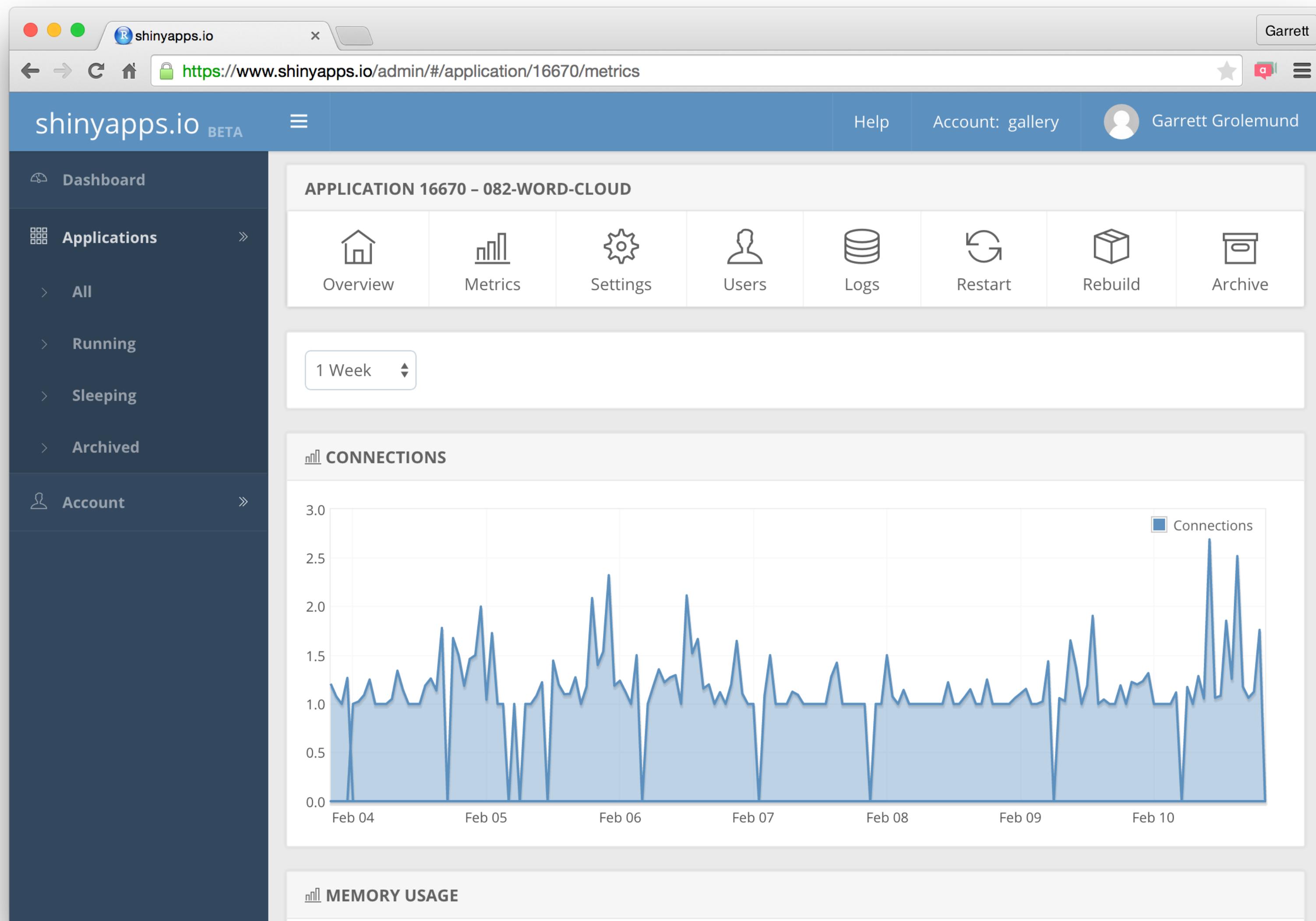


Scalable

Shiny from



WITH BUILT-IN METRICS



MEMBERSHIP PRICING

FREE

\$ 0 /month

New to Shiny? Deploy your applications for FREE.

5 Applications

25 Active Hours

Community Support

[Sign Up Now](#)

STARTER

\$ 13 /month

(or \$145/year)

More applications. More active hours!

25 Applications

100 Active Hours

Premium Email Support

[Sign Up Now](#)

BASIC

\$ 49 /month

(or \$550/year)

Take your users to the next level!

Unlimited Applications

500 Active Hours

Performance Boost

Premium Email Support

[Sign Up Now](#)

STANDARD

\$ 119 /month

(or \$1,330/year)

Password protection? Authenticate your users!

Unlimited Applications

2,000 Active Hours

Authentication

Performance Boost

Premium Email Support

[Sign Up Now](#)

PROFESSIONAL

\$ 349 /month

(or \$3,860/year)

Professional has it all! Personalize your domains.

Unlimited Applications

10,000 Active Hours

Authentication

Account Sharing

Performance Boost

Custom Domains

Premium Email Support

[Sign Up Now](#)

POSIT CONNECT

<https://posit.co/products/enterprise/connect/>



- ✓ **Push-button publish from RStudio**
Shiny apps, R Markdown docs, and more
- ✓ **Self-managed content**
content authors decide permissions
- ✓ **Scheduled reports**
automatically run and email Rmd
- ✓ **Support**
direct priority support

evaluation free trial

Build your own
server

SHINY SERVER

<https://posit.co/products/open-source/shinyserver/>



- ✓ Deploy Shiny apps to the internet
- ✓ Run on-premises
move computation closer to the data
- ✓ Host multiple apps on one server
- ✓ Deploy inside the firewall
- ✓ xcopy deployment





EXERCISE

- ▶ Create a folder called movies in the ShinyApps folder
- ▶ Move any one of the movies app R scripts you worked on into this folder, and rename it as app.R
- ▶ Also move the movies.Rdata file into this folder
- ▶ Run the app
- ▶ Rstudio will take you to a browser or local view where you can interact with the deployed app

3m 00s

SHINYPROXY

<https://shinyproxy.io/>



✓ **Secure access**

LDAP, GoogleAuth, SSL, and more

✓ **Management**

View open apps and generate usage statistics

✓ **Docker**

uses docker to maintain dedicated app instances

✓ **Open Source**

✓ **ACL's**

Configurable to restrict app access

Free & open
source!

More information

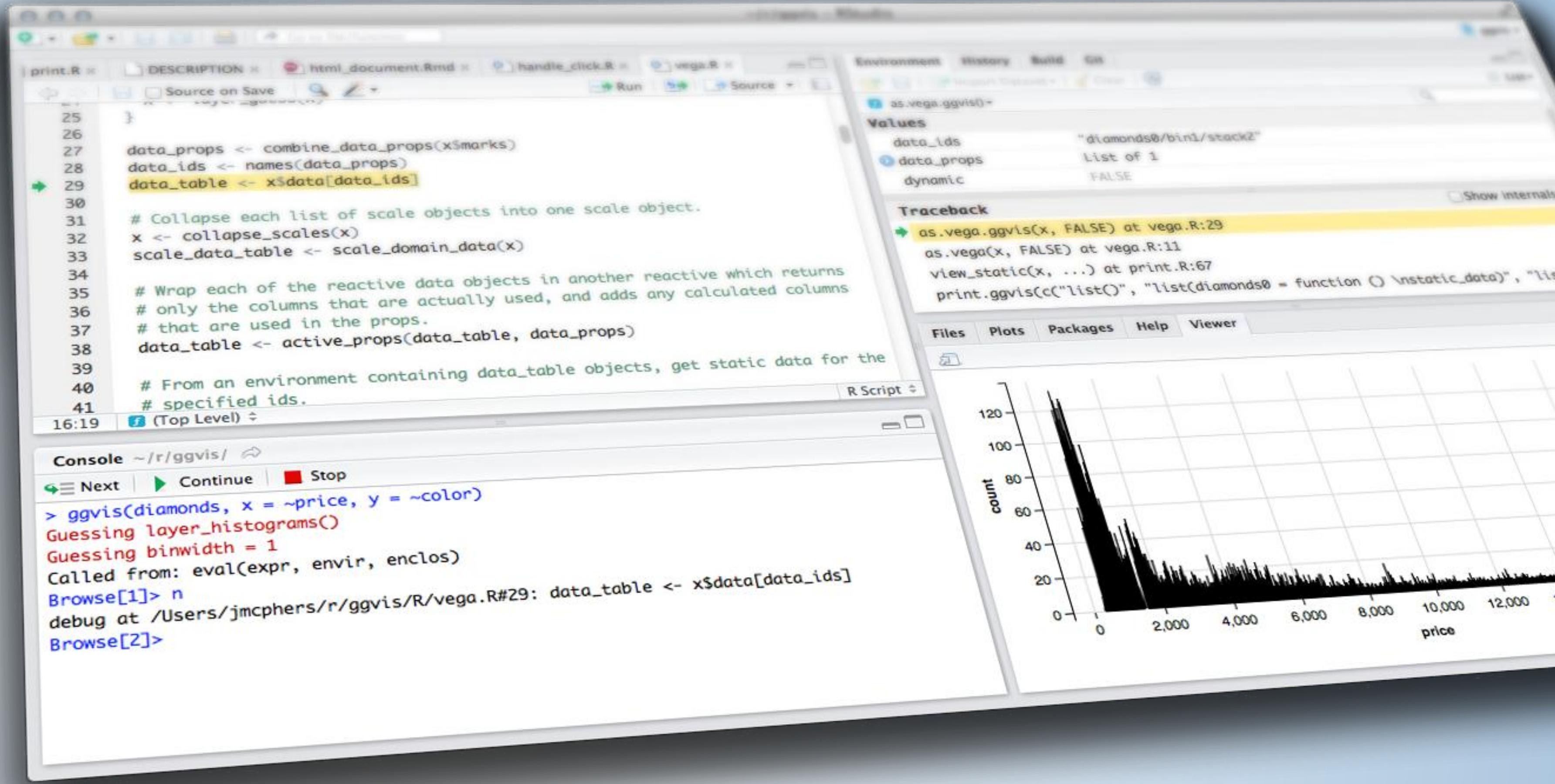
- ▶ Shinyproxy documentation:
<https://shinyproxy.io/documentation/deploying-apps/>
- ▶ Shinyproxy github repo:
<https://github.com/openanalytics/shinyproxy>

How do I deploy apps?



COURSE OVERVIEW

& INTRODUCTION TO SHINY



Shiny from

