BUILDING WEB APPLICATIONS IN R WITH SHINY

Welcome to the course!

Movie browser

Y-axis:
Audience Score

X-axis:
Critics Score

Color by:
MPAA Rating

Alpha:
0 0.5 1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Size:
0 2 5
0 1 2 3 4 5

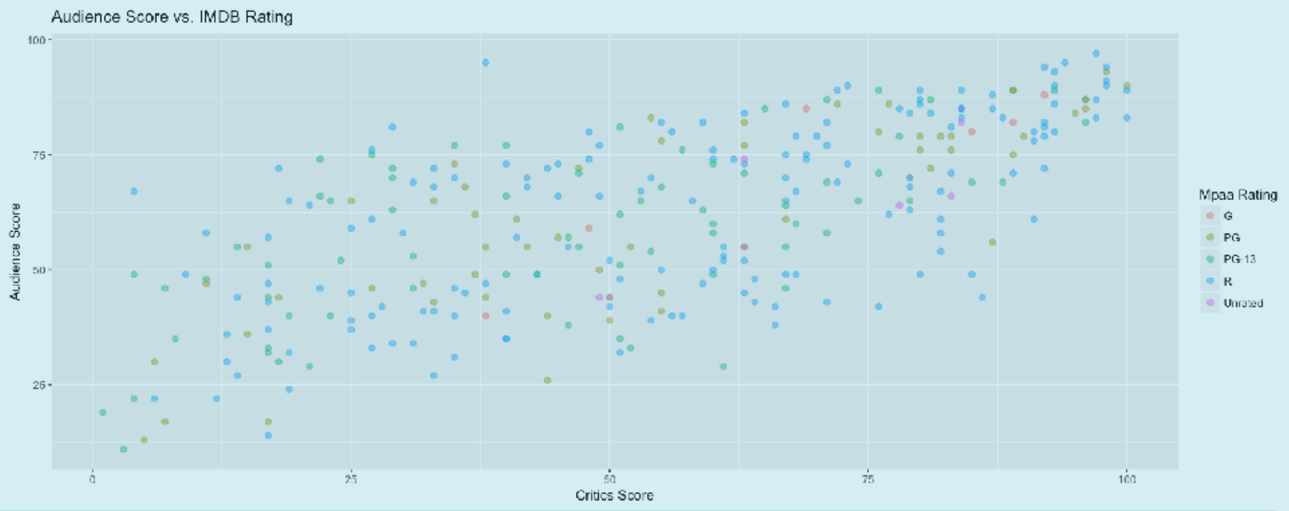
☑ Show data table

Plot title
Audience Score vs. IMDB Rating

Select movie type(s):
☐ Documentary
☑ Feature Film
☐ TV Movie

Sample size:
300

Write CSV



There are 300 Feature Film movies in this dataset.

Show 10 entries Search:

title	title_type	genre	runtime	mpaa_rating	studio	thtr_rel_date
The Godfather, Part II	Feature Film	Mystery & Suspense	202	R	Paramount Pictures	1974-12-20T05:00:00Z
Titanic	Feature Film	Drama	194	PG-13	Paramount Pictures	1997-12-19T05:00:00Z
Meet Joe Black	Feature Film	Drama	178	PG-13	Universal Pictures	1998-11-13T05:00:00Z
The Postman	Feature Film	Action & Adventure	177	R	Warner Home Video	1997-12-25T05:00:00Z
The English Patient	Feature Film	Drama	162	R	Miramax Films	1996-11-15T05:00:00Z
Harry Potter and the Chamber of Secrets	Feature Film	Science Fiction & Fantasy	161	PG	Warner Bros. Pictures	2002-11-15T05:00:00Z

Background

- You are familiar with R as a programming language.
- You are familiar with the Tidyverse, specifically **ggplot2** and **dplyr.**

Help

www.rstudio.com/resources/cheatsheets/

shiny.rstudio.com/

Tips

- Always run the entire script, not just up to the point where you're developing code.
- Sometimes the best way to see what's wrong is to run the app and review the error.
- Watch out for commas!

Anatomy of a Shiny 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

shinyApp()
Creates the Shiny app object

Data

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

Revisit

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



server <- function(input, output) {}



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

Data used for this app

Let’s practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

User interface

Anatomy of a Shiny app

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


server <- function(input, output) {}


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

User interface
- Inputs defined and laid out
- Outputs laid out

Server function
- Outputs calculated
- Any other calculations needed for outputs are performed

server

```
ggplot(data = movies, aes_string(x = input$x, y = input$y)) +
  geom_point()
```

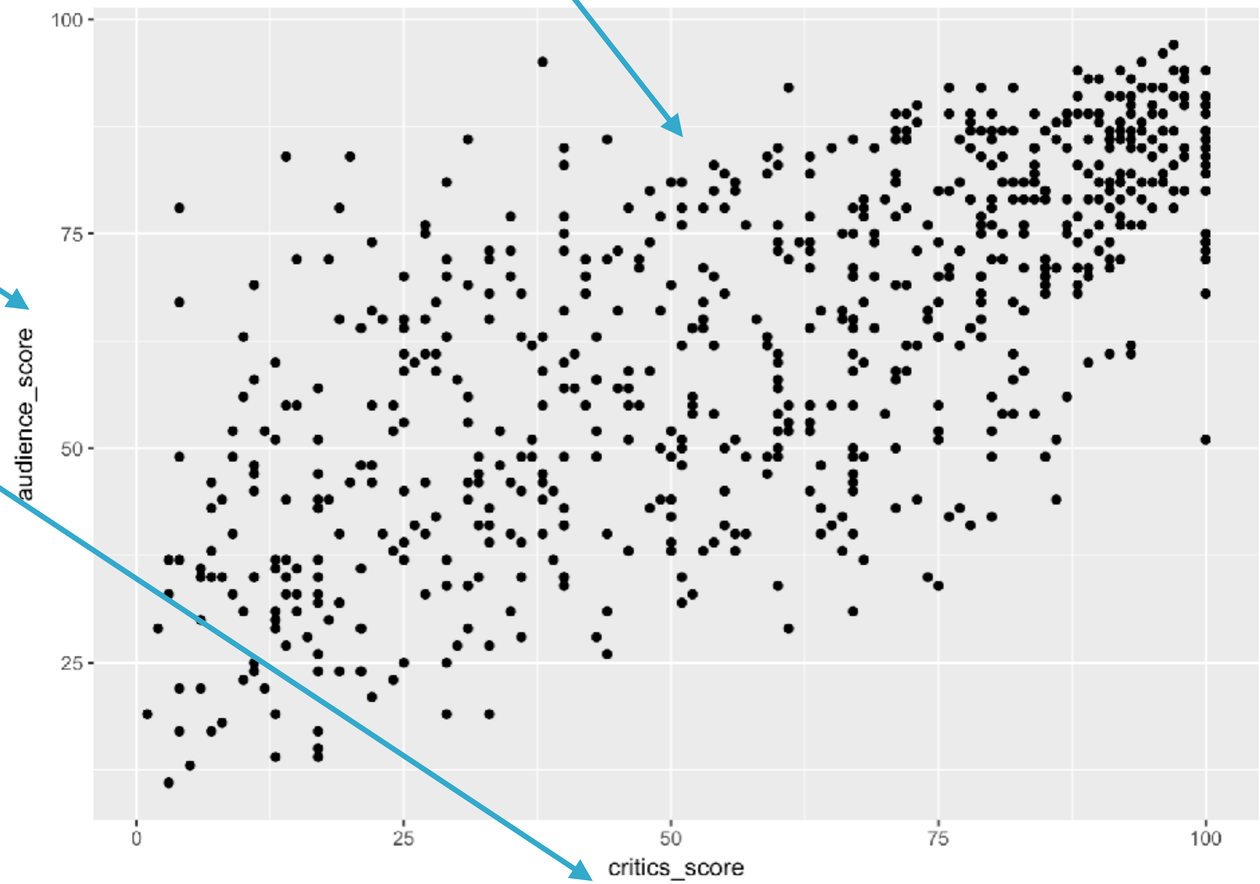
ui

Y-axis:

audience_score

X-axis:

critics_score

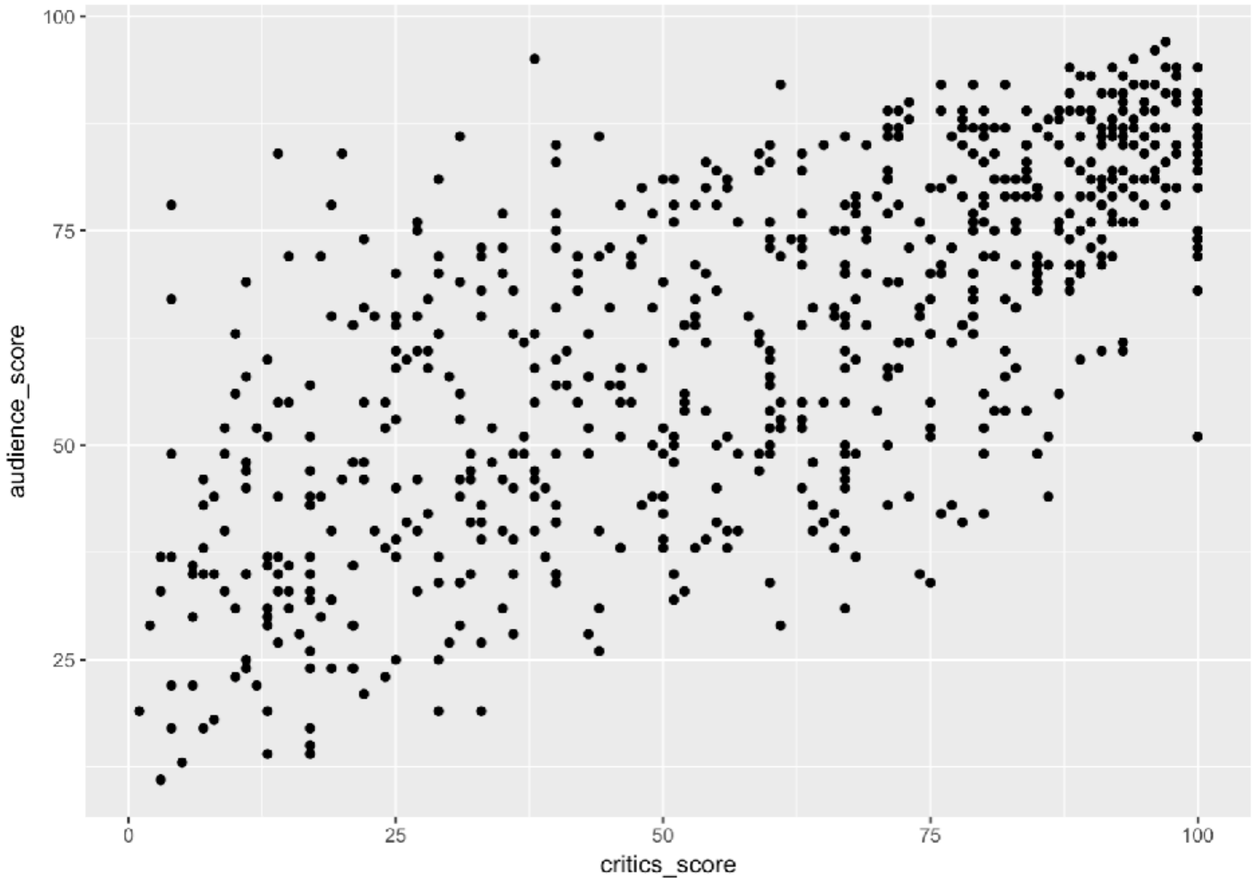


Y-axis:

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")
    )
  )
)
```

```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

sidebarPanel

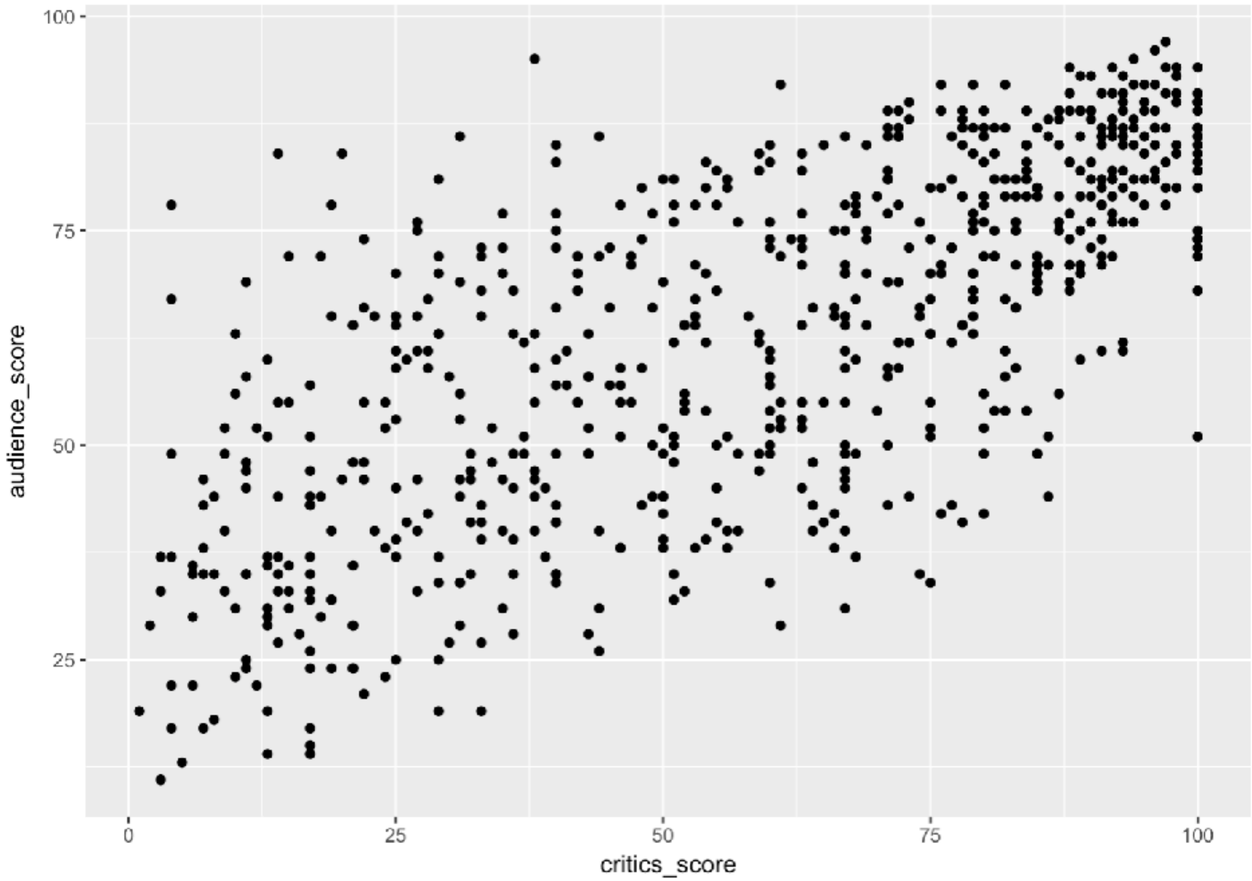
Y-axis:

audience_score

X-axis:

critics_score

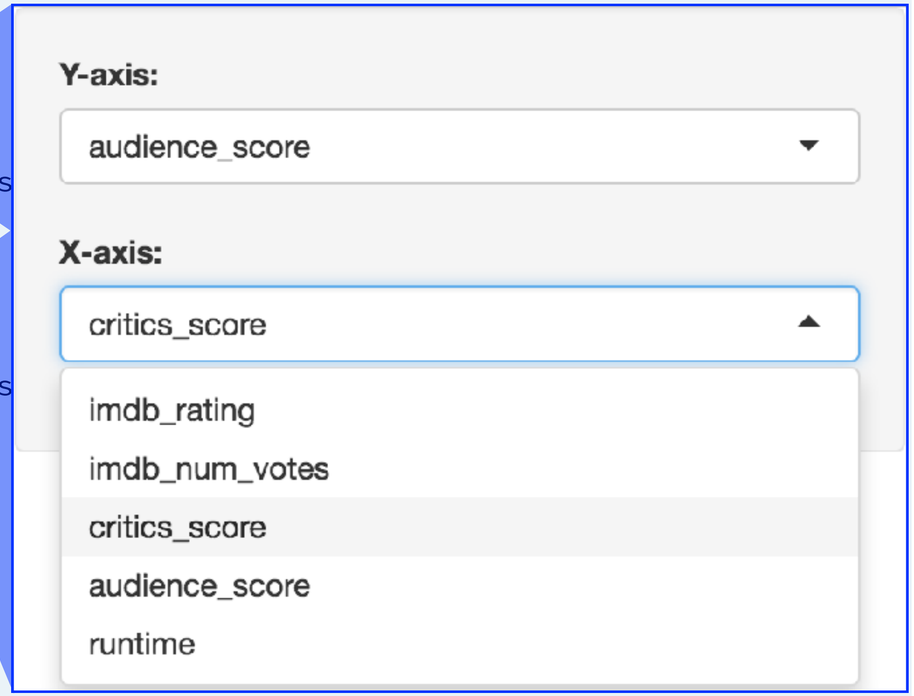
mainPanel



```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

```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
                  selected = "audience_score"),
      # Select variable for x-axis
      selectInput(inputId = "x", label = "X-axis:",
                  choices = c("imdb_rating", "imdb_num_votes
                  selected = "critics_score")
    ),

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



```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")
    )
  )
)
```

```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 main panel containing **output** elements that get created in the server function

BUILDING WEB APPLICATIONS IN R WITH SHINY

Let's practice!

BUILDING WEB APPLICATIONS IN R WITH SHINY

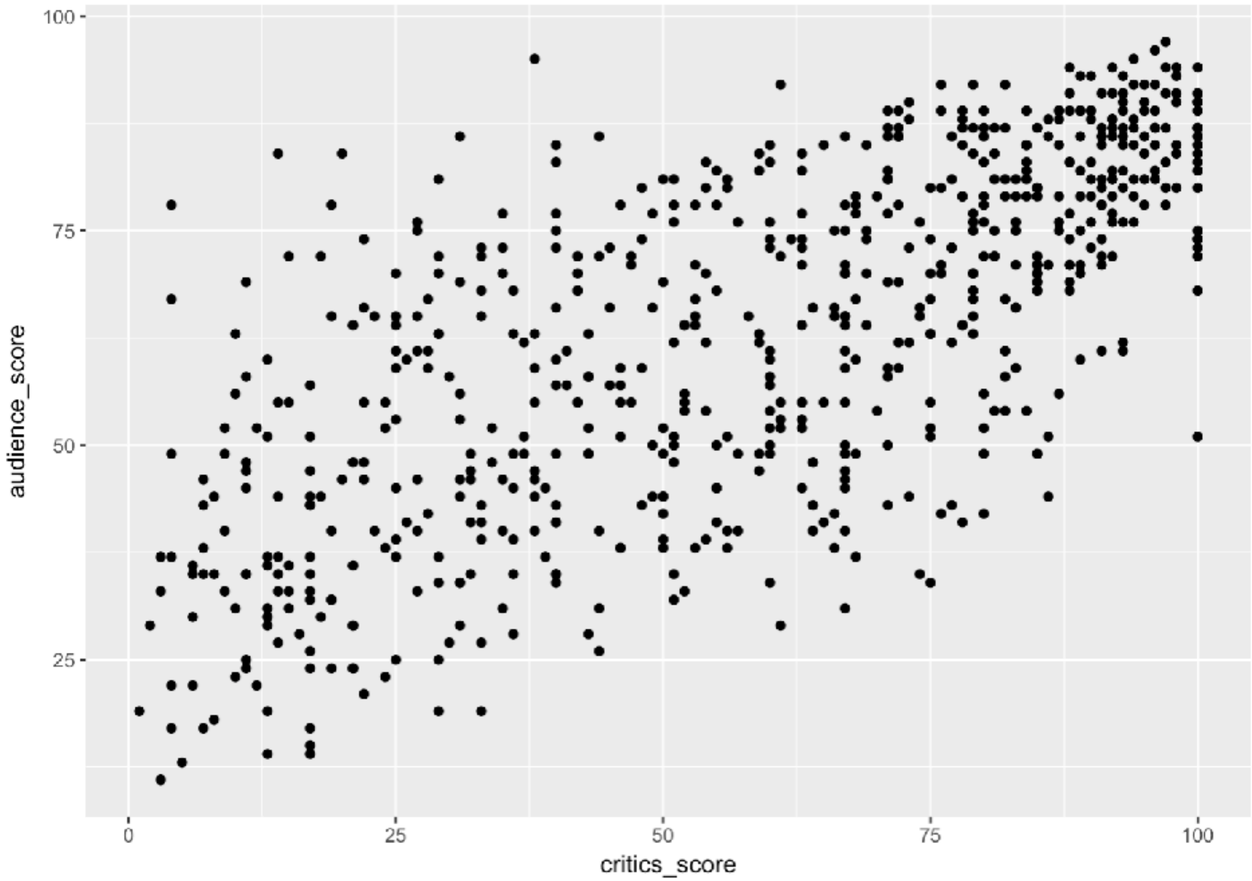
Server function

Y-axis:

audience_score

X-axis:

critics_score



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

  # Create 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 scatterp
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 functi
  output$scatterplot <- renderPlot({
    ggplot(data = movies, aes_string(x = input$x, y = i
      geom_point()
  })
}
```

Specifies how the plot output should be updated

```r
# 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

```r
ui <- fluidPage(
. . .
      # 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")
. . .
)
```

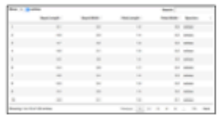
```r
server <- function(input, output) {

  # 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()
  })
}
```

Rules of server functions

1. Save objects to display to `output$xx`
2. Build objects to display with `render*()`
3. Use input values with `input$xx`

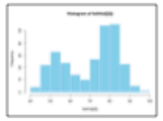
Outputs - render*() and *Output() functions work together to add R output to the UI

works with



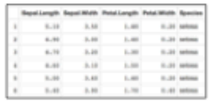
DT::**renderDataTable**(expr, options, callback, escape, env, quoted) — **dataTableOutput**(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile) — **imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)



renderPlot(expr, width, height, res, …, env, quoted, func) — **plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)

renderPrint(expr, env, quoted, func, width) — **verbatimTextOutput**(outputId)

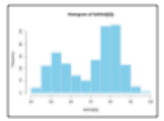


renderTable(expr,…, env, quoted, func) — **tableOutput**(outputId)

foo

renderText(expr, env, quoted, func) — **textOutput**(outputId, container, inline)

renderUI(expr, env, quoted, func) — & **uiOutput**(outputId, inline, container, …) **htmlOutput**(outputId, inline, container, …)



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

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

```
ui <- fluidPage(
. . .
    # Output: Show scatterplot
    mainPanel(
      plotOutput(outputId = "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()
  })
}
```

Reactivity


input$x
expression()
output$y
Create your
own reactive values
*Input()
Update
Render
reactive output
render*()

Putting all the pieces together

```
# Create the Shiny app object
shinyApp(ui = ui, server = server)
```

BUILDING WEB APPLICATIONS IN R WITH SHINY

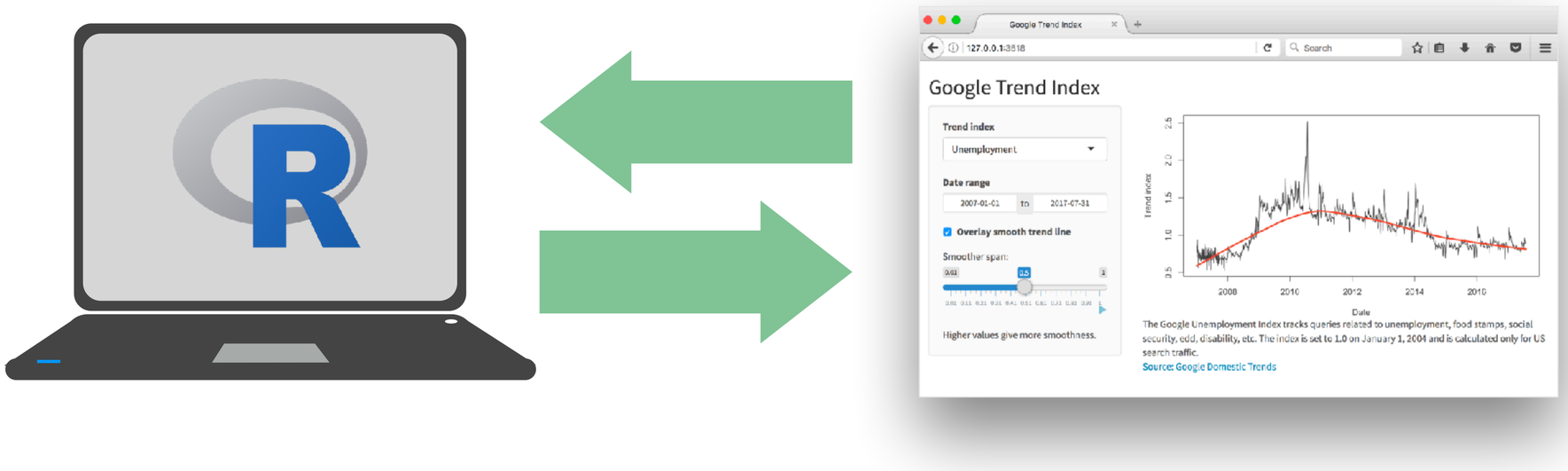
Let's practice!

Recap

Every Shiny app has a webpage that the user visits, and behind this webpage there is a computer that serves this webpage by running R.

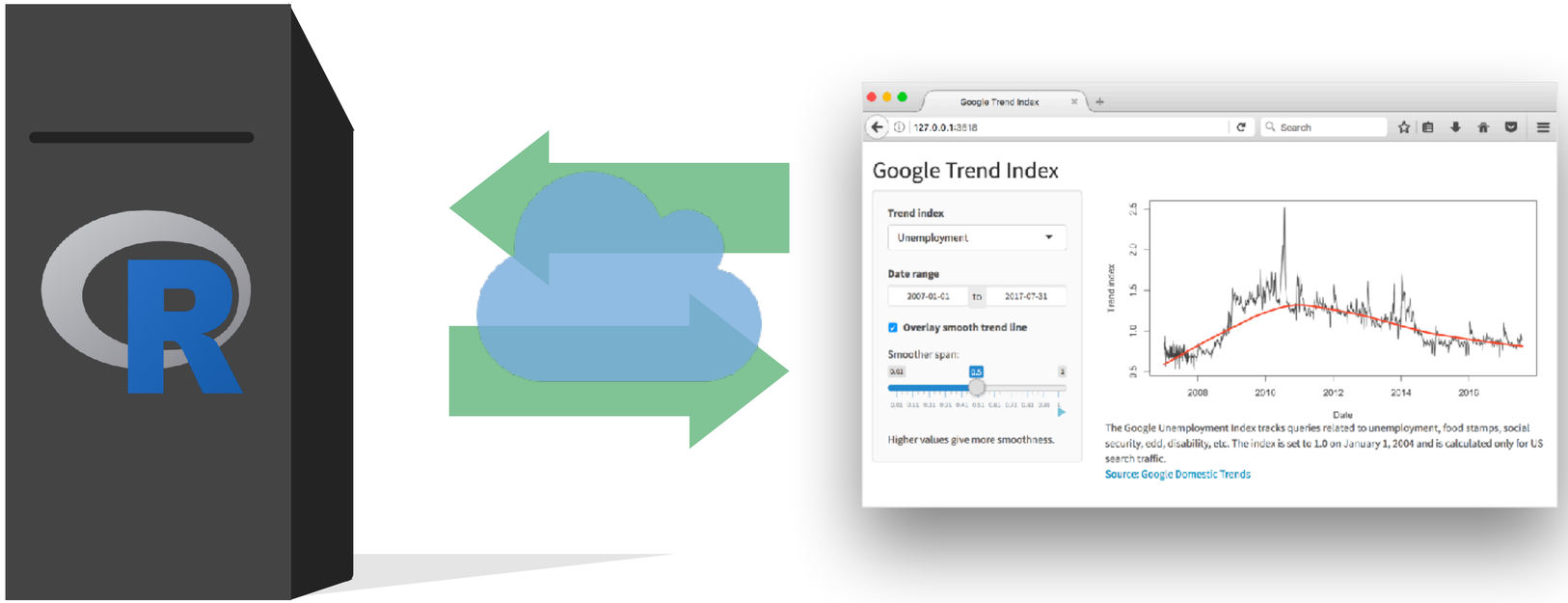
When running your app locally,

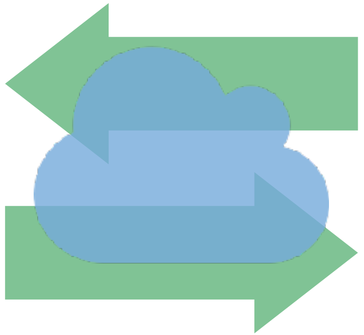
the computer serving your app is your computer.

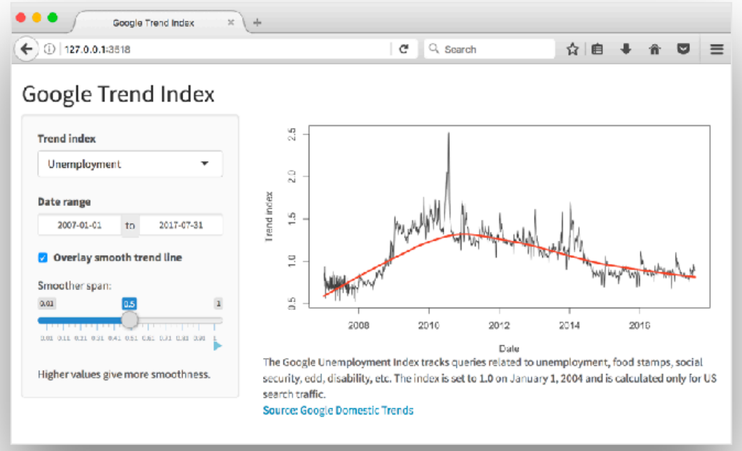


When your app is deployed,

the computer serving your app is a web server.



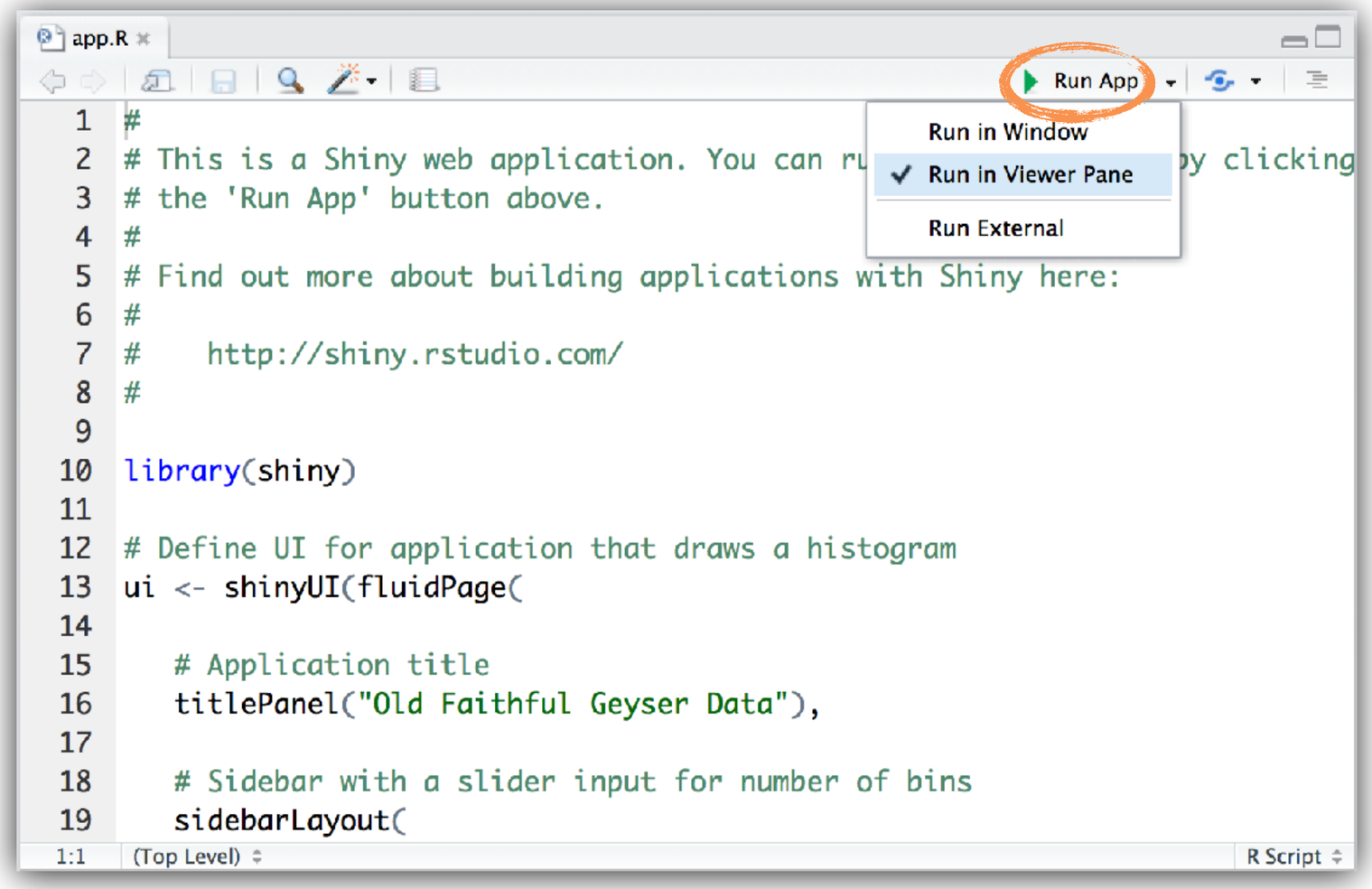




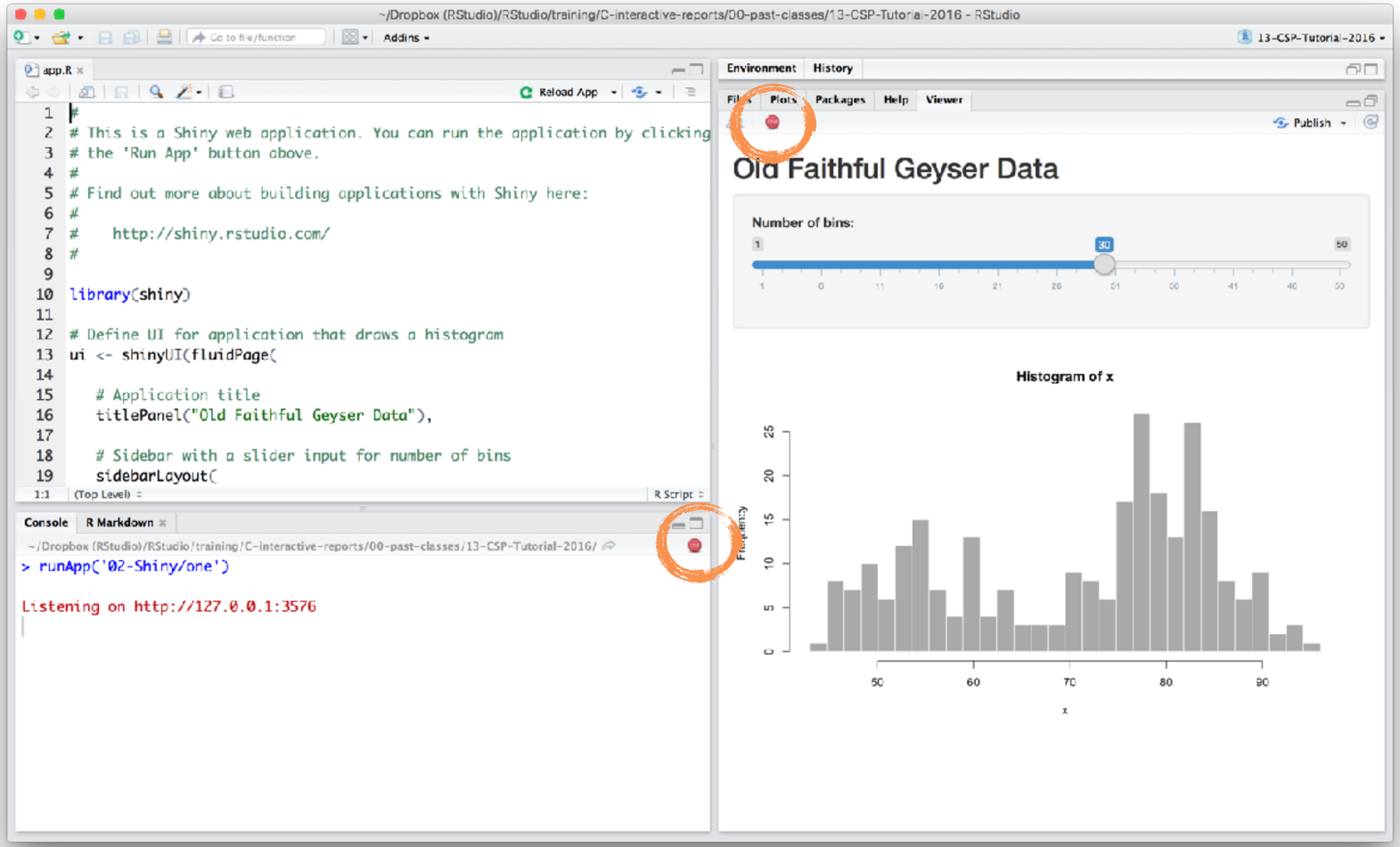
Server instructions

User interface

Change display



Close an app


~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/00-past-classes/13-CSP-Tutorial-2016 - RStudio
app.R
Reload App
#
This is a Shiny web application. You can run the application by clicking
the 'Run App' button above.
#
Find out more about building applications with Shiny here:
#
http://shiny.rstudio.com/
#
library(shiny)
# Define UI for application that draws a histogram
ui <- shinyUI(fluidPage(
# Application title
titlePanel("Old Faithful Geyser Data"),
Sidebar with a slider input for number of bins
sidebarLayout(
Console
R Markdown
~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/00-past-classes/13-CSP-Tutorial-2016/
> runApp('02-Shiny/one')
Listening on http://127.0.0.1:3576
Environment
History
Plots
Packages
Help
Viewer
Publish
Old Faithful Geyser Data
Number of bins:
Histogram of x
Frequency
x

On to the next chapter!