



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

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

Shiny :: CHEAT SHEET

Basics

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

Building an App

Complete the template by adding arguments to fluidPage(), and a body to the server function.

Inputs

call edit values from the user

[www.rstudio.com/resources/cheatsheets/](http://www.rstudio.com/resources/cheatsheets/)

Shinyapp(ui = ui, server = server)

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

server - a function with instructions on how to build and rebuild the R objects displayed in the UI

shinyapp - combines ui and server into an app. Wrap with runApp() if calling from a sourced script, or inside a function.

SHARE YOUR APP

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 <https://shinyapps.io>

2. Click the Publish icon in the RStudio IDE or run: `rsconnect::deployApp()` ("path to directory")

Build or purchase your own Shiny Server at [www.rstudio.com/products/shiny-server/](https://www.rstudio.com/products/shiny-server/)

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

renderDataframe(outputId, expr, options = list(callback, escape, env, quoted, ...))

renderImage(expr, env, quoted, deviceFile)

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

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

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

renderText(expr, env, quoted, func)

renderUI(expr, env, quoted, func)

renderTable(outputId, expr, options = list(callback, escape, env, quoted, ...))

renderImage(outputId, width, height, click, dblclick, hover, hoverDelay, id, idAttr, hoverDelayType, brush, clickId, hoverId)

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

verbatimTextOutput(outputId)

tableOutput(outputId)

textOutput(outputId, contains, inline)

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

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

textInput(inputId, label, value)

passwordInput(inputId, label, value)

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

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

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

submitButton(inputId, label)

submitButton(inputId, label, value)

textInput(inputId, label, value)

Shiny from R Studio

Get Started Gallery Articles Reference Deploy Help Contribute

[shiny.rstudio.com/](http://shiny.rstudio.com/)

Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

# 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()
```

## User interface

controls the layout and appearance of app

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

## Server function

contains instructions needed to build app

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

## 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





BUILDING WEB APPLICATIONS IN R WITH SHINY

# Let's practice!



BUILDING WEB APPLICATIONS IN R WITH SHINY

# User interface

# Anatomy of a Shiny app

```
library(shiny)
```

```
library("movies.Rdata")
```

```
ui <- fluidPage()
```

## User interface

- Inputs defined and laid out
- Outputs laid out

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

## Server function

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

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

**server**

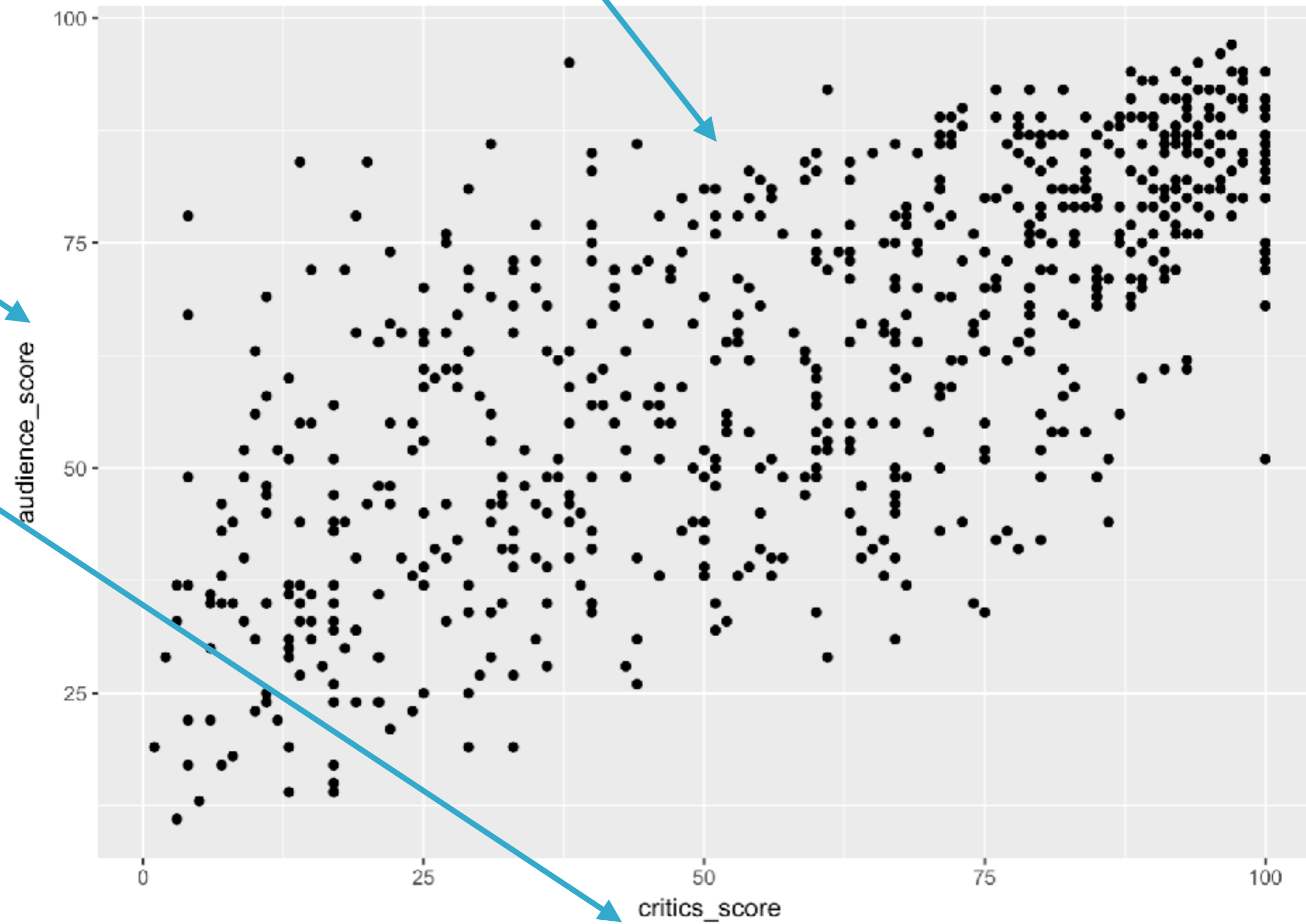
```
ggplot(data = movies, aes_string(x = input$x, y = input$y)) +  
  geom_point()
```

**ui****Y-axis:**

audience\_score

**X-axis:**

critics\_score

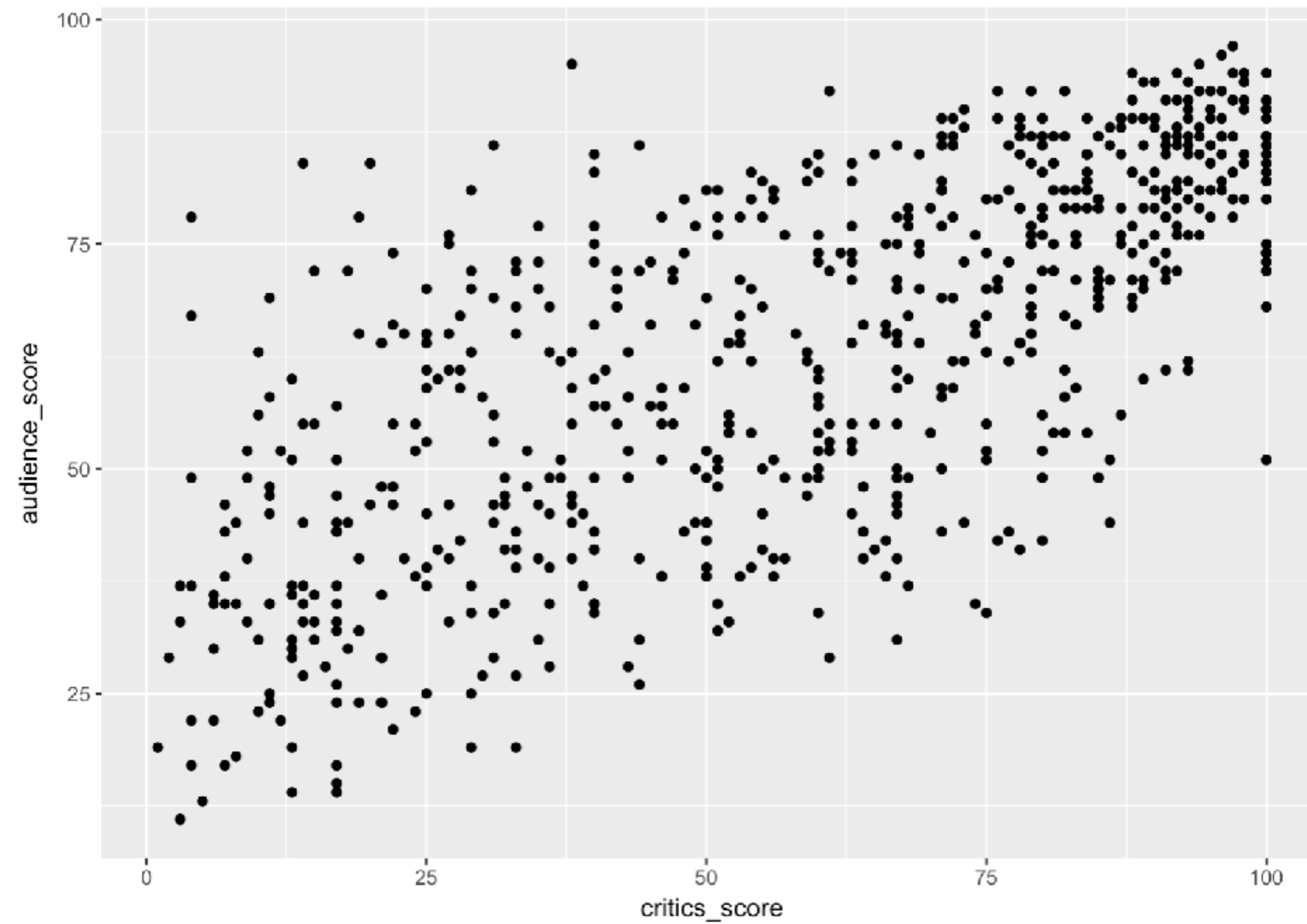


**Y-axis:**

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")
    )
  )
)
```



```
# 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

## sidebarPanel

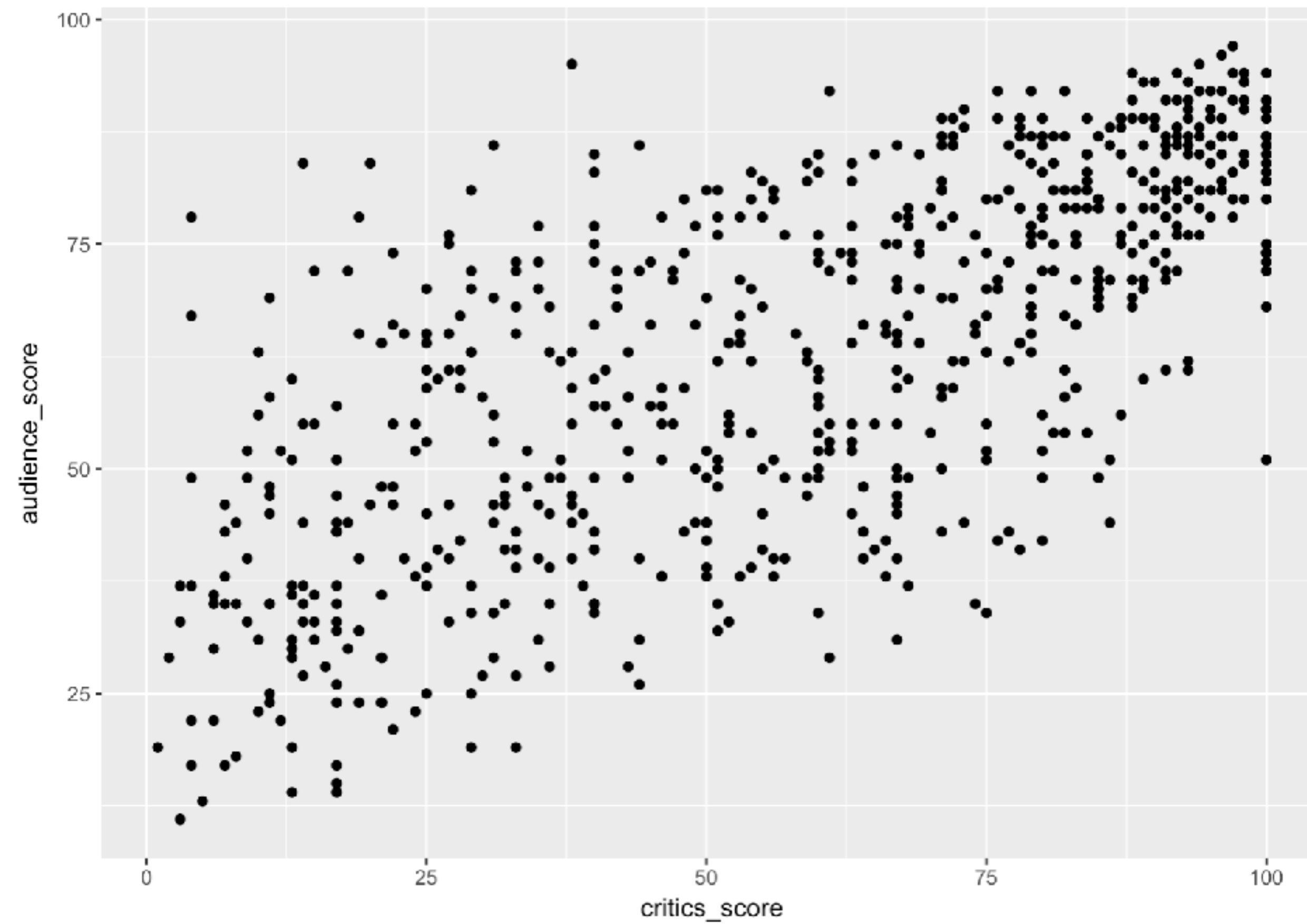
Y-axis:

audience\_score ▼

X-axis:

critics\_score ▼

## mainPanel



```
# 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

```
# 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",  
                              "audience_score"),  
                  selected = "audience_score"),
```

```
      # Select variable for x-axis
```

```
      selectInput(inputId = "x", label = "X-axis:",  
                  choices = c("imdb_rating", "imdb_num_votes",  
                              "critics_score"),  
                  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")
    )
  )
)
```



```
# 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

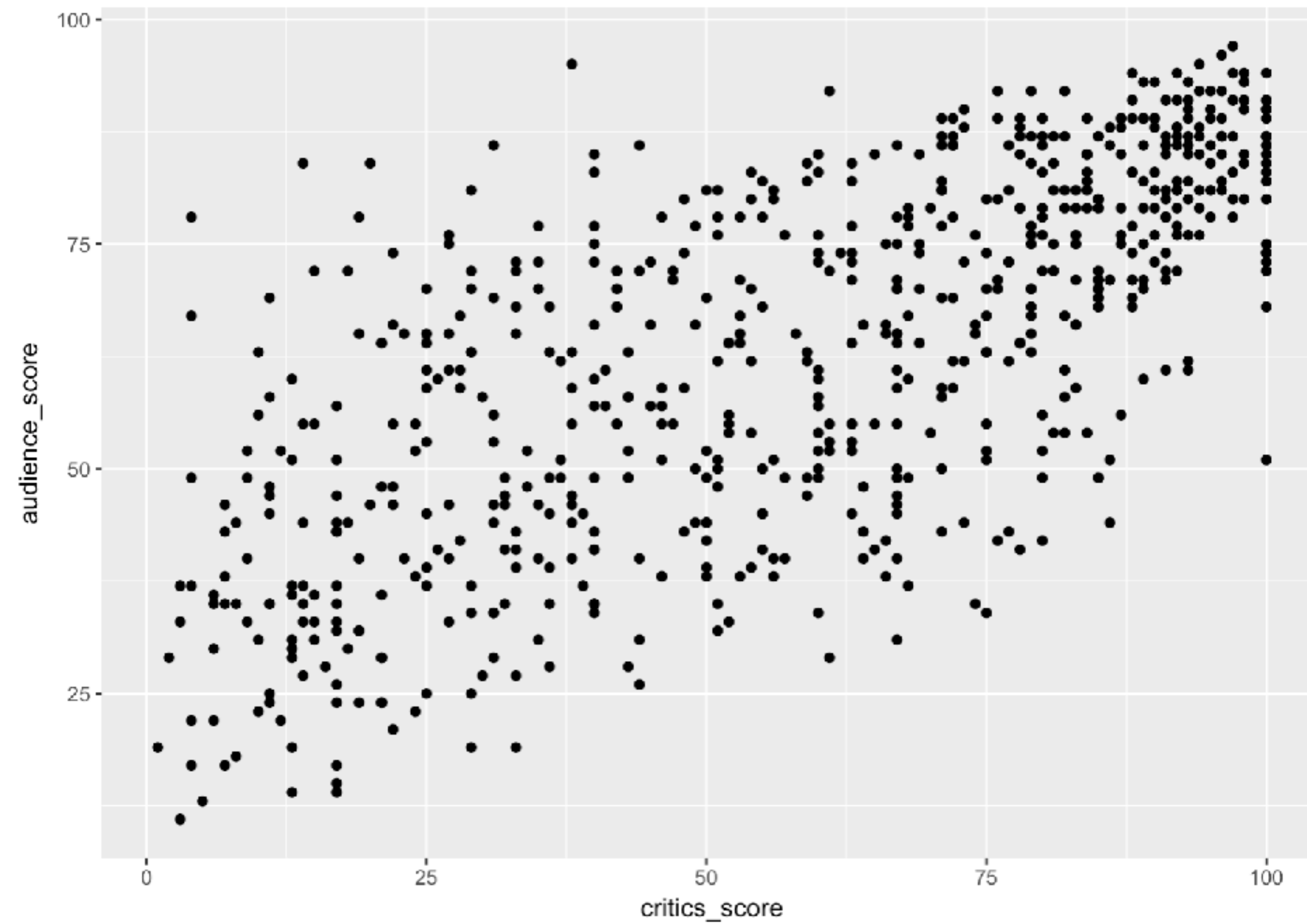
# Server function

**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 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
```

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

Specifies how the plot  
output should be updated

```
# Define server function required to create the scatterplot
```

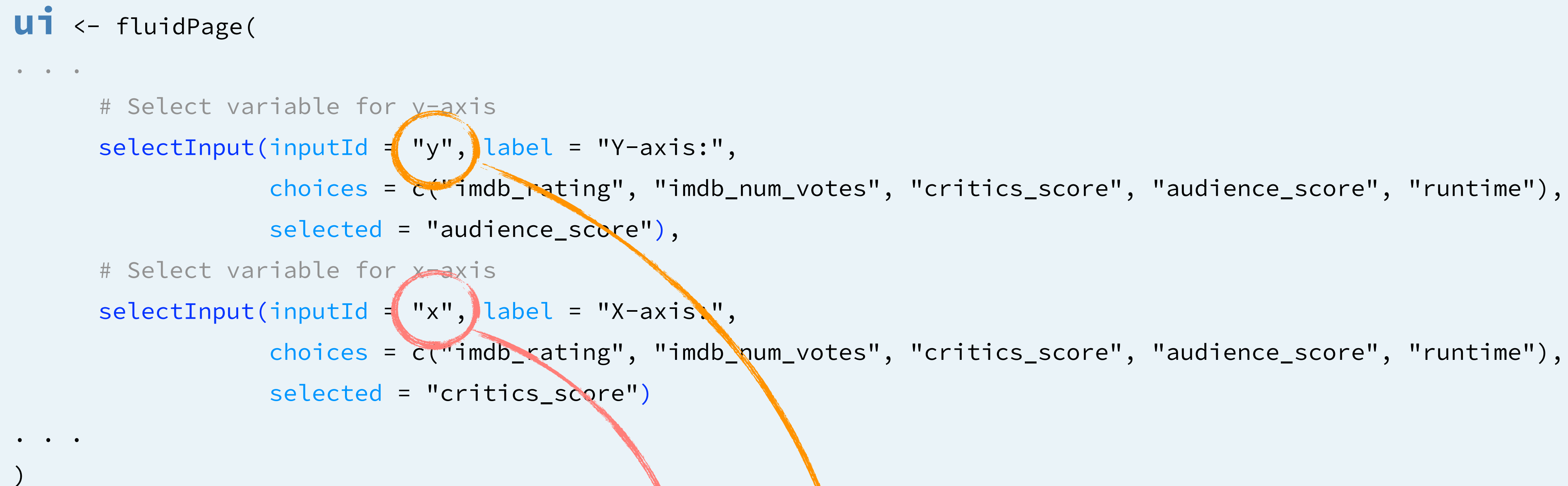
```
server <- function(input, output) {
```

```
# 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

```
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")  
  . . .  
)
```



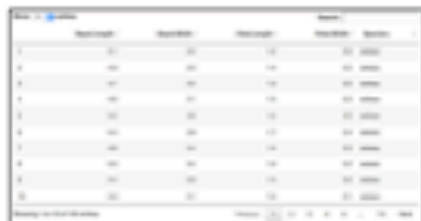
```
server <- function(input, output) {  
  
  # Create the scatterplot object the plotOutput function is expecting  
  output$scatterplot <- renderPlot({  
    ggplot(data = movies, aes_string(x = input$x, y = input$y)) +  
      geom_point()  
  })  
}
```

# 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

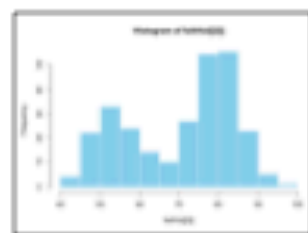


**DT::renderDataTable**(expr, options, callback, escape, env, quoted) **dataTableOutput**(outputId, icon, ...)



**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)

```
'data.frame': 3 obs. of 2 variables:
 $ Sepal.Length: num 5.1 4.9 4.7
 $ Sepal.Width : num 3.5 3 3.2
```

**renderPrint**(expr, env, quoted, func, width)

**verbatimTextOutput**(outputId)

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	5.2	3.7	1.5	0.2	setosa
5	5.0	3.4	1.4	0.2	setosa
6	5.4	3.9	1.7	0.2	setosa

**renderTable**(expr,..., env, quoted, func)

**tableOutput**(outputId)

foo

**renderText**(expr, env, quoted, func)

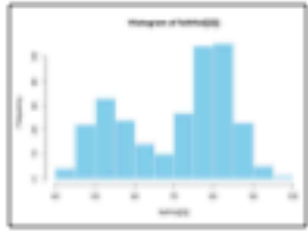
**textOutput**(outputId, container, inline)



**renderUI**(expr, env, quoted, func)

&

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



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

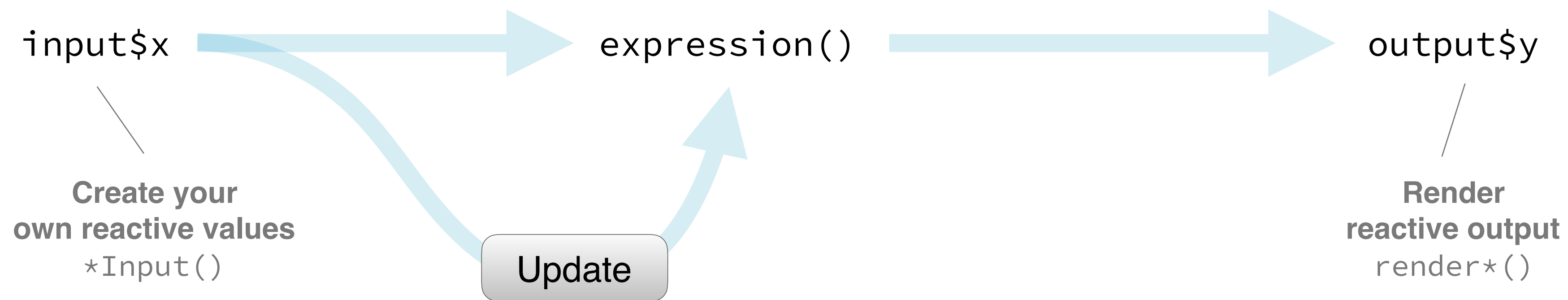
**plotOutput**(outputId, width, height, click,  
dblclick, hover, hoverDelay, inline,  
hoverDelayType, brush, clickId, hoverId)

```
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



# Putting all the pieces together

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



BUILDING WEB APPLICATIONS IN R WITH SHINY

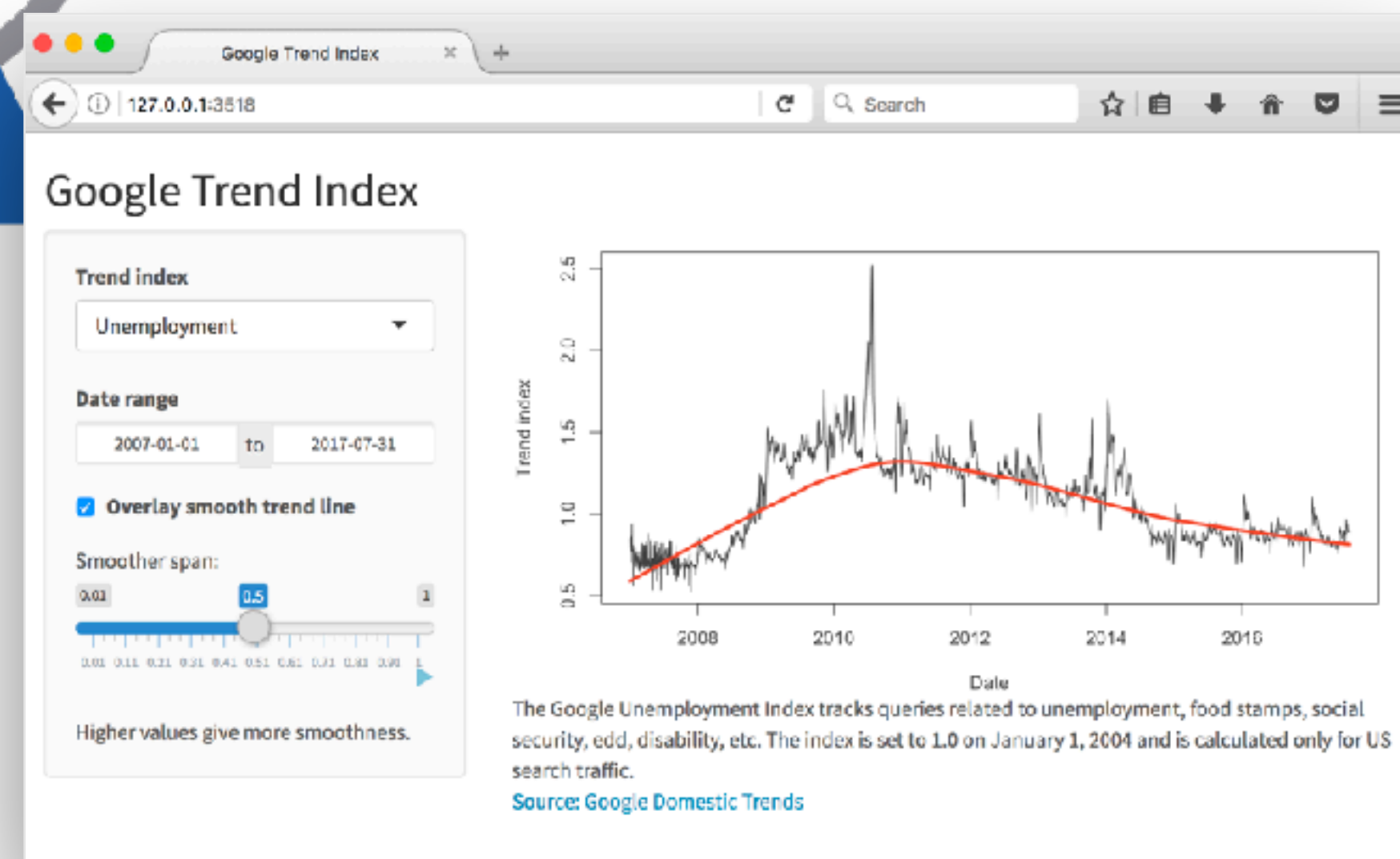
# Let's practice!



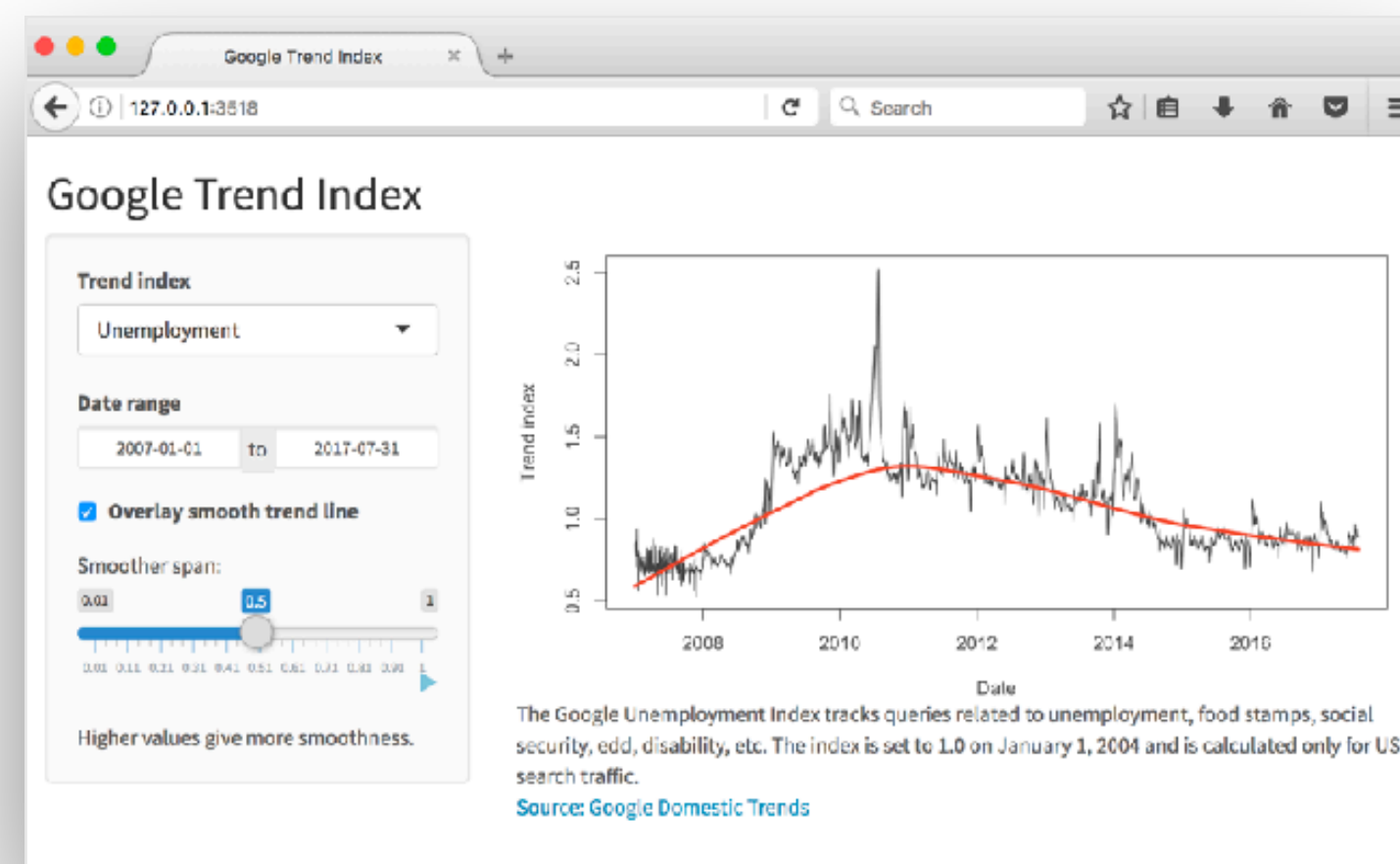
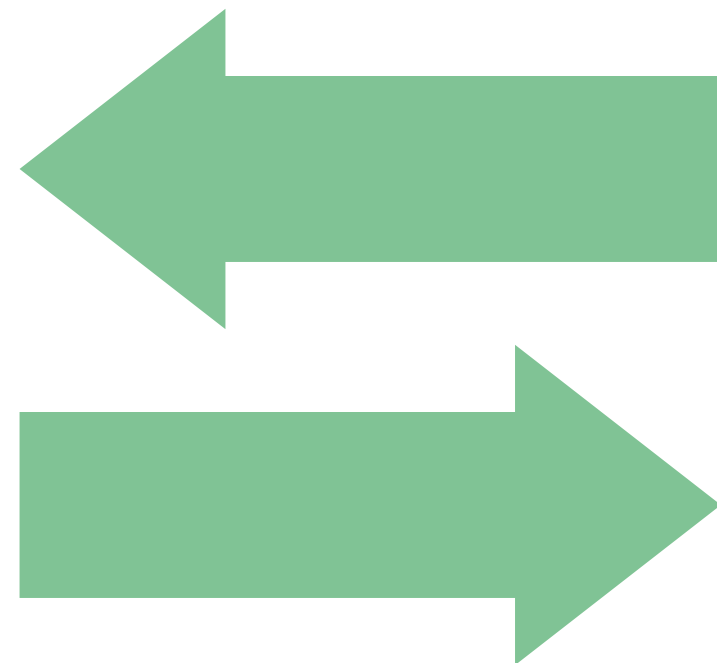
BUILDING WEB APPLICATIONS IN R WITH SHINY

# Recap

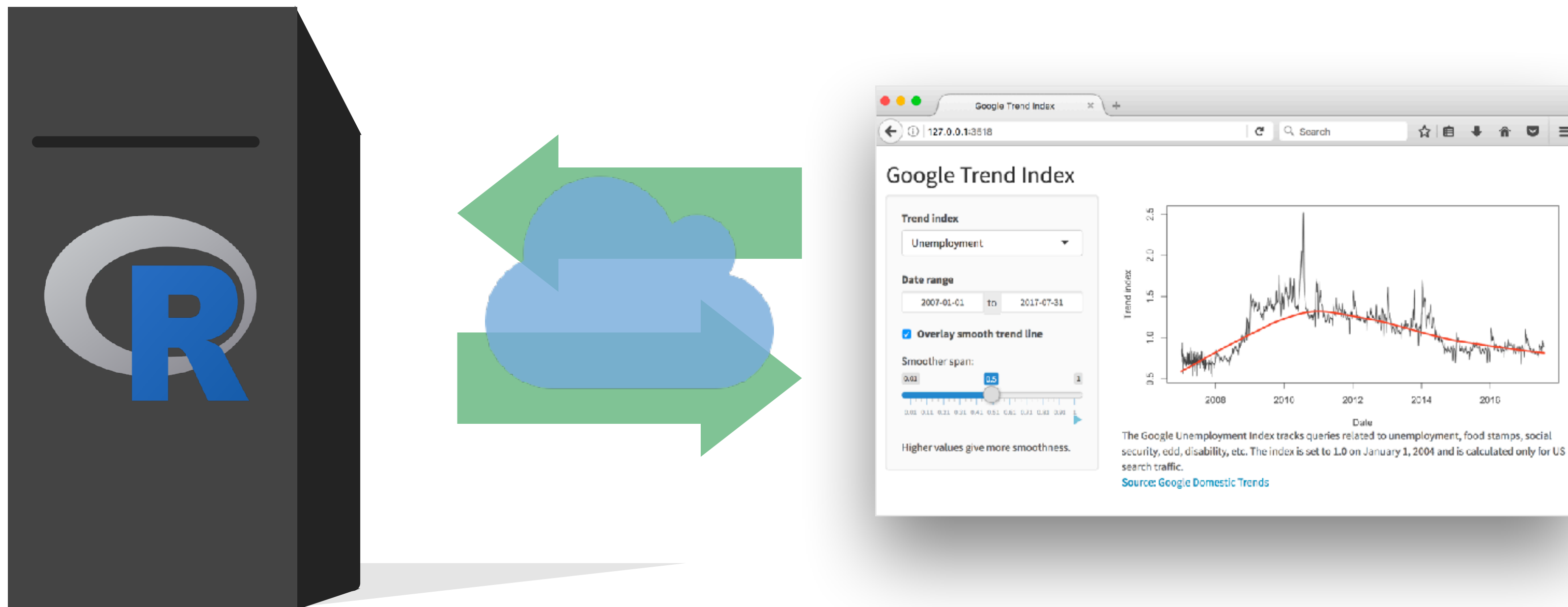
Every Shiny app has a webpage that the user visits,  
and behind this webpage there is a computer  
that serves this webpage by running R.



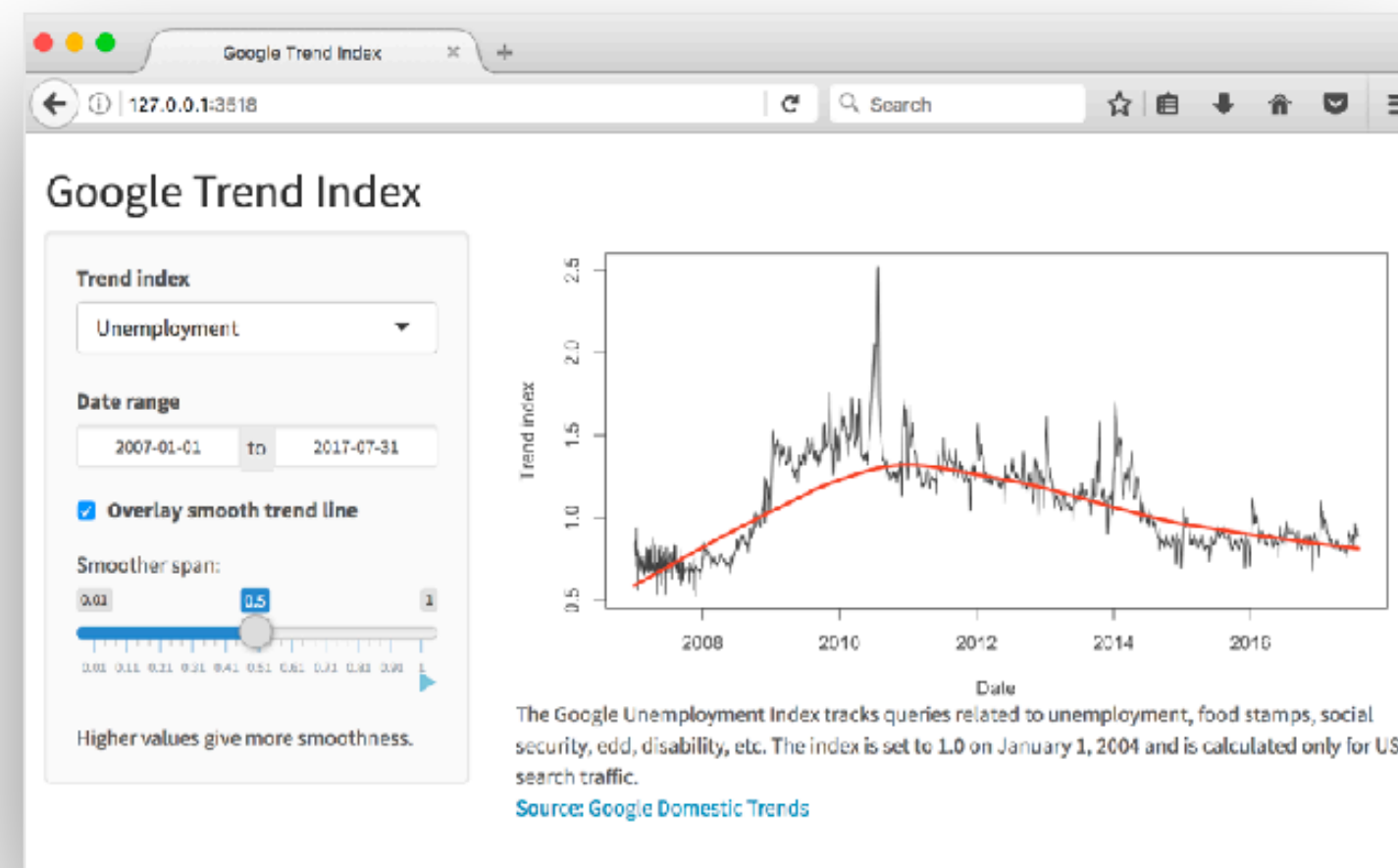
When running your app locally,  
the computer serving your app is your computer.



When your app is deployed,  
the computer serving your app is a web server.





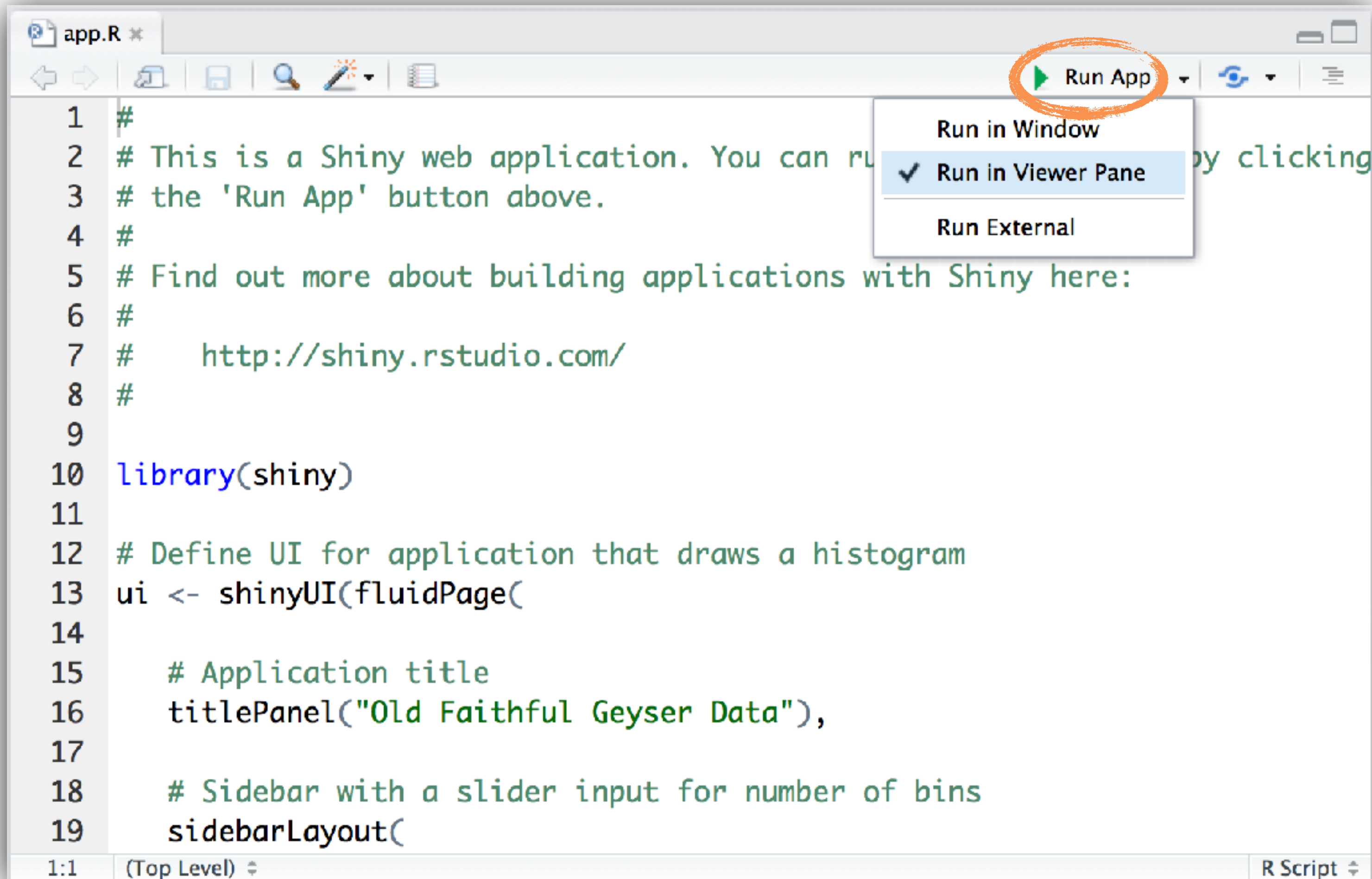


Server instructions



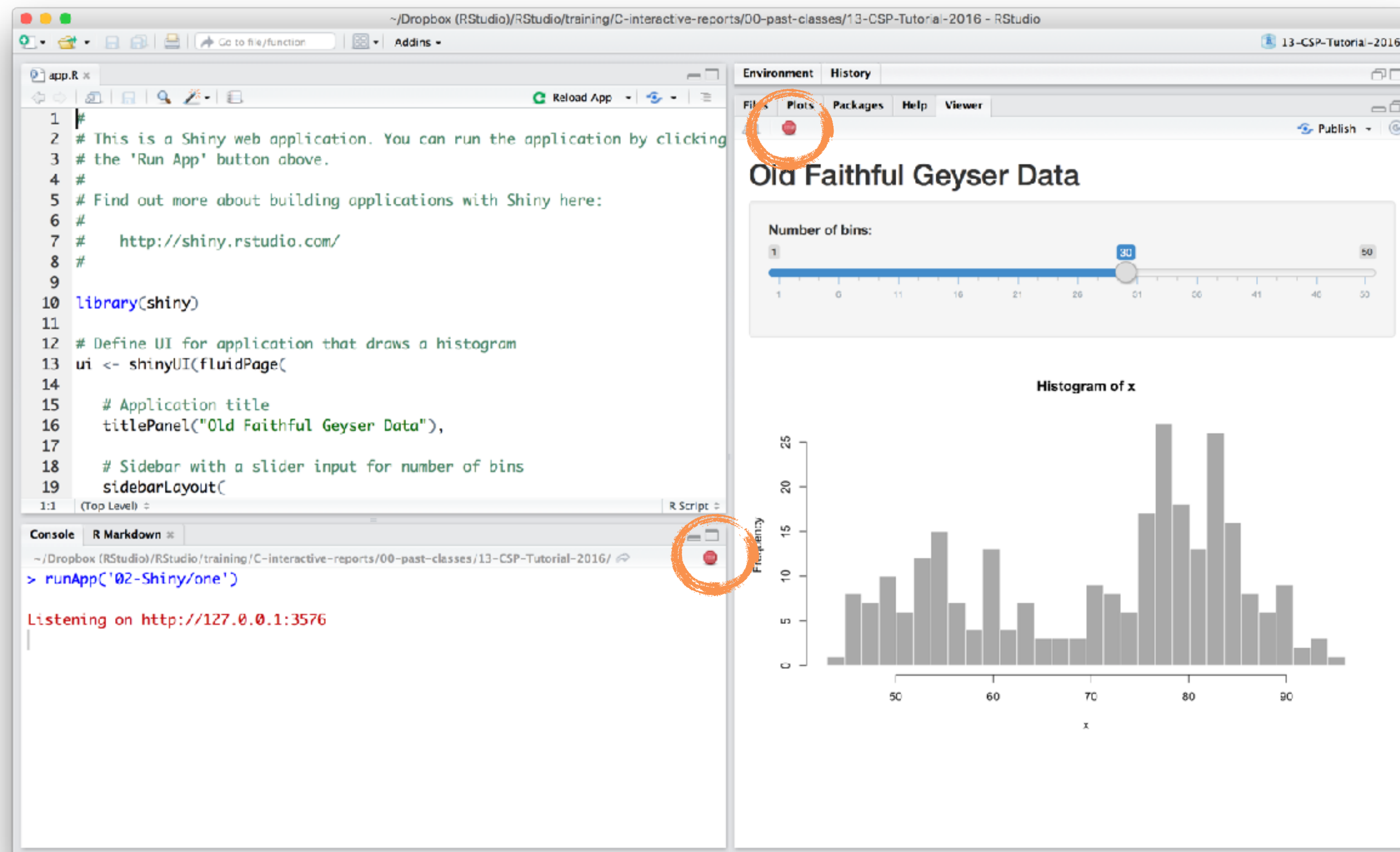
User interface

# Change display



```
1 #
2 # This is a Shiny web application. You can run the application by clicking
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



The screenshot shows the RStudio interface with a Shiny application running. The application is titled "Old Faithful Geyser Data" and displays a histogram of x with a slider for the number of bins. The R console shows the command `runApp()` and the message "Listening on http://127.0.0.1:3576". The "Plots" tab in the top right is highlighted with an orange circle, and the "Run" button in the bottom right is also highlighted with an orange circle.

```
1 #
2 # This is a Shiny web application. You can run the application by clicking
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   # Application title
15   titlePanel("Old Faithful Geyser Data"),
16   # Sidebar with a slider input for number of bins
17   sidebarLayout(
18     #
19     #
20   )
21 )
```

Console

```
> runApp('02-Shiny/one')
```

Listening on http://127.0.0.1:3576

Old Faithful Geyser Data

Number of bins:

1 30 50

Histogram of x

Frequency

x



BUILDING WEB APPLICATIONS IN R WITH SHINY

# On to the next chapter!