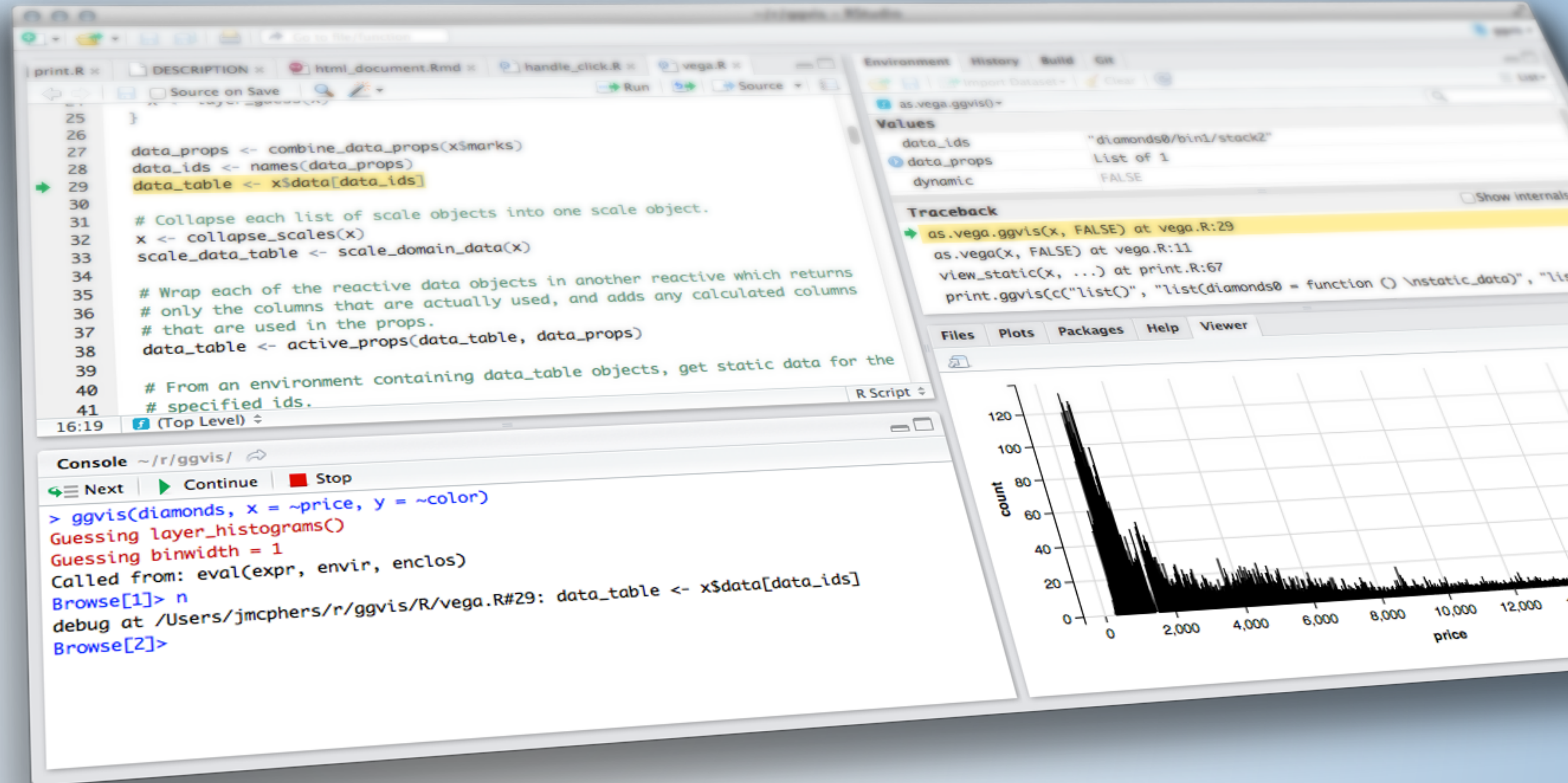




A FAST INTRODUCTION

TO SHINY

Shiny from  Studio™

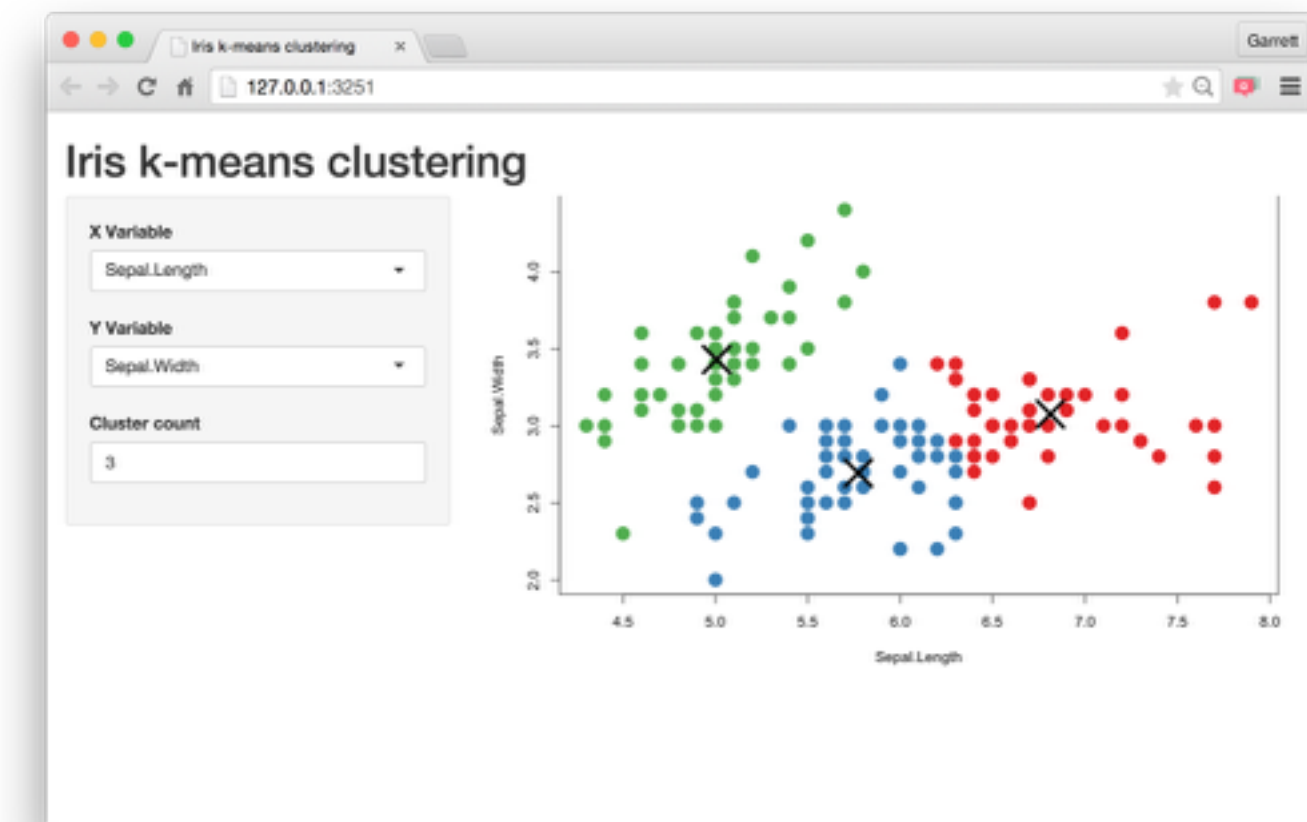
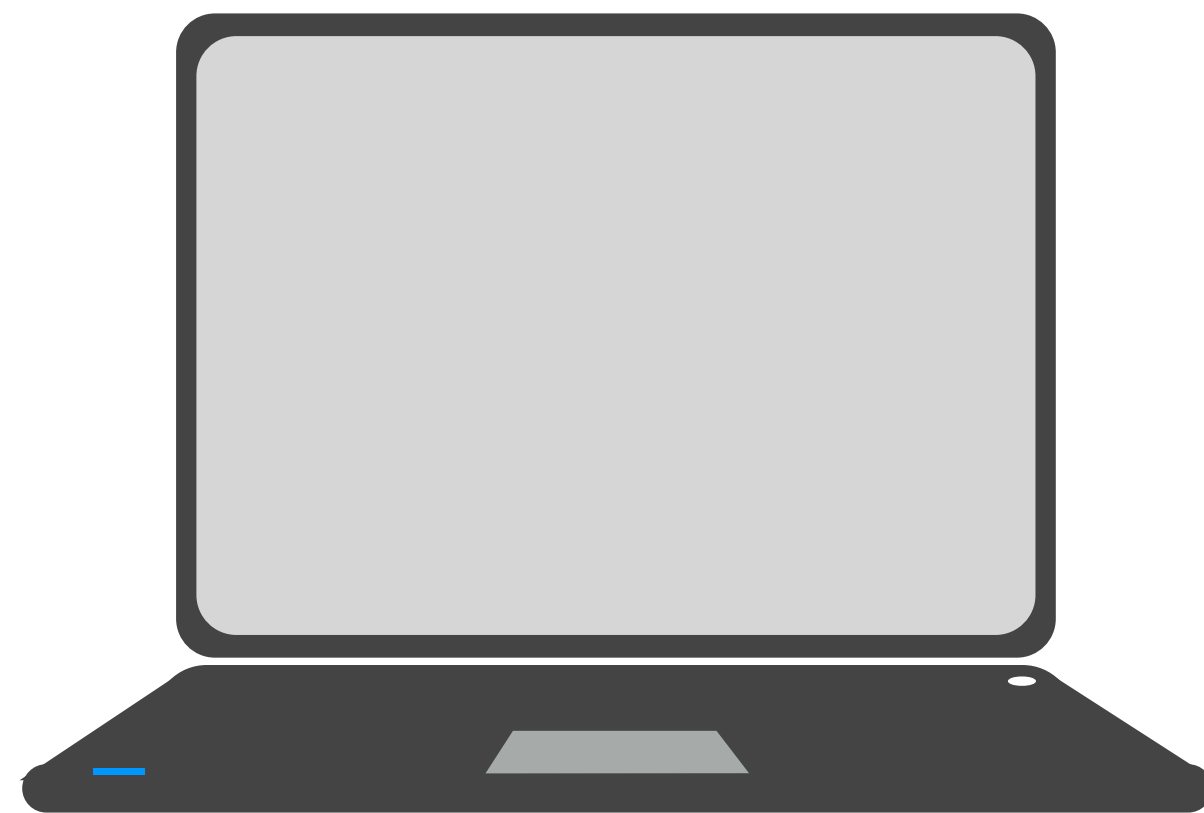
OUTLINE

- ▶ High level view
- ▶ Anatomy of a Shiny app
 - ▶ User interface
 - ▶ Server function
 - ▶ Running the app
- ▶ File Structure
- ▶ Sharing your app

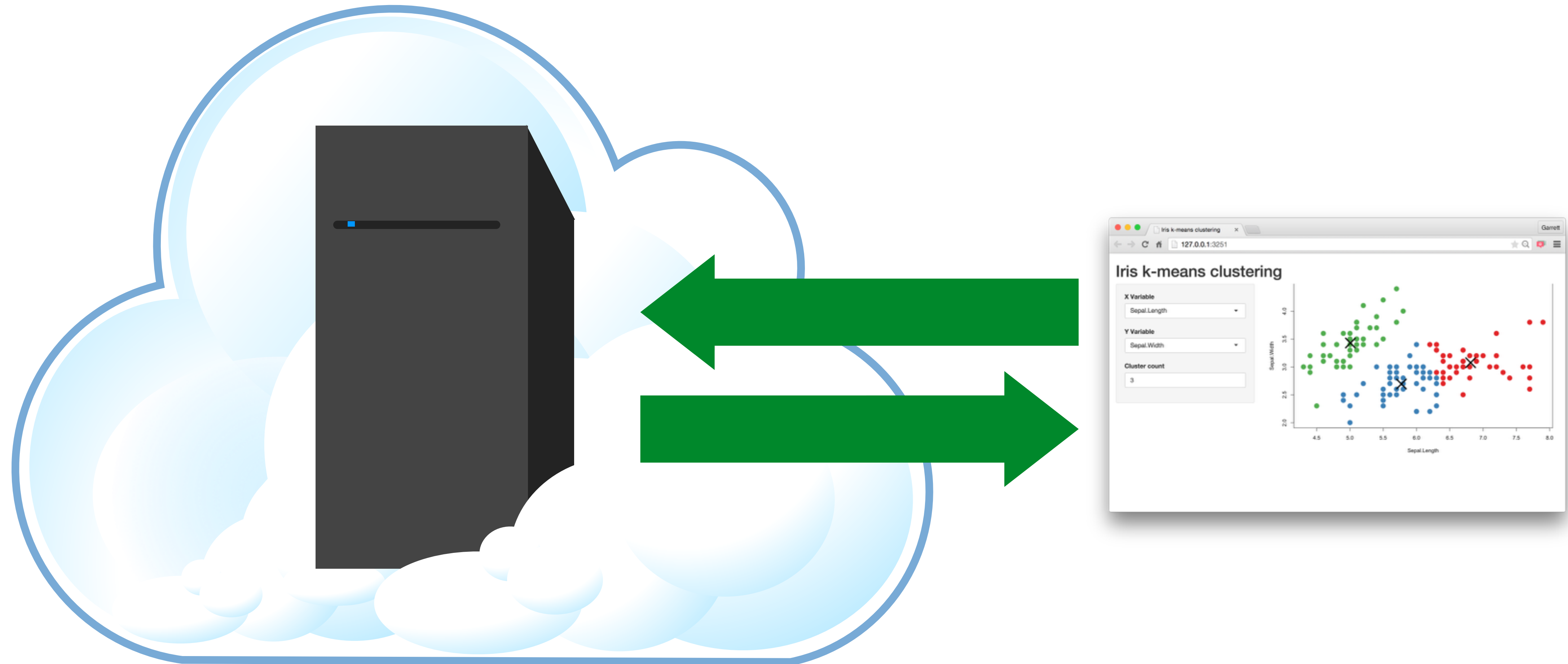
High level

view

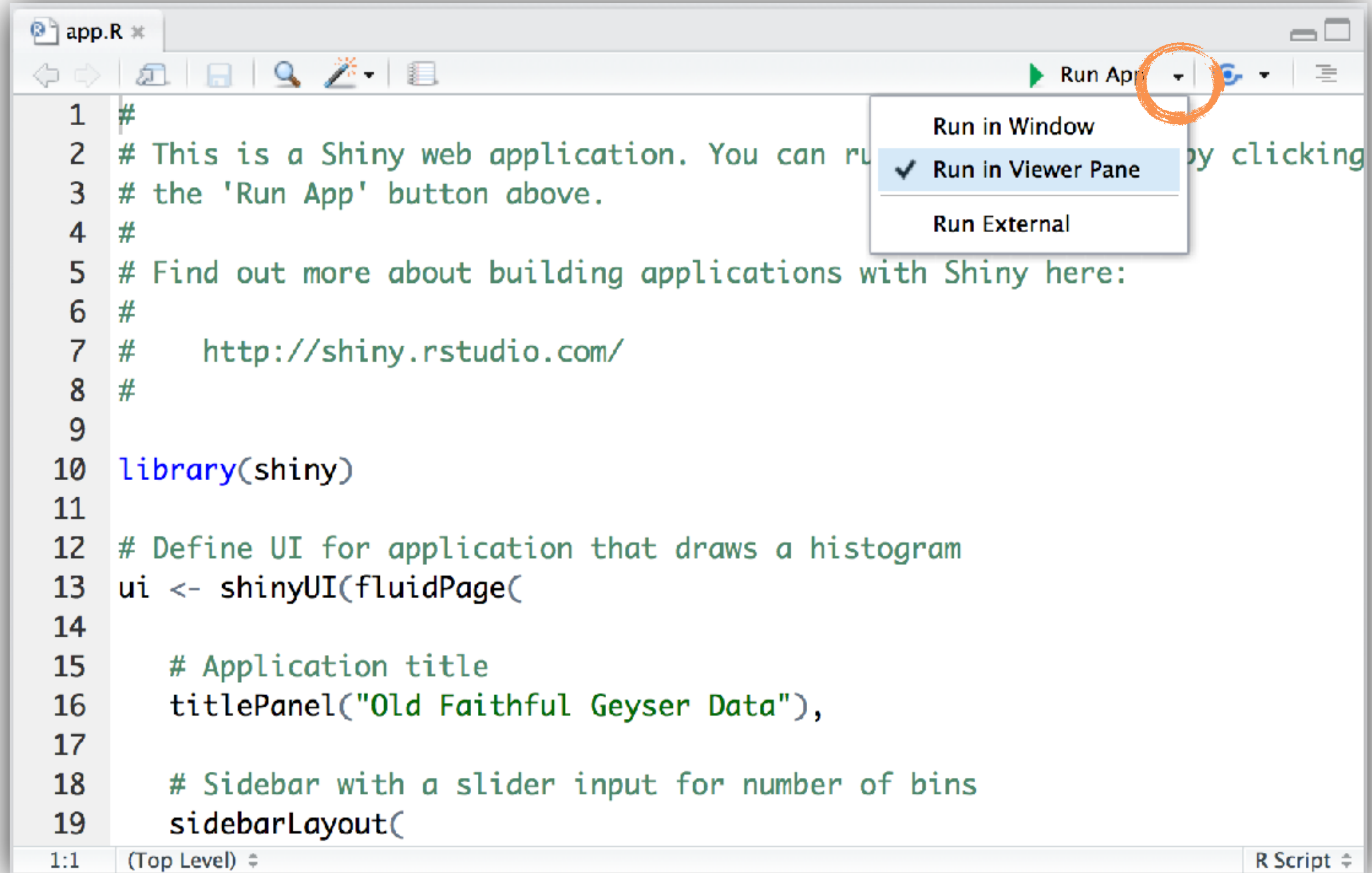
Every Shiny app is maintained by a computer running R



Every Shiny app is maintained by a computer running R

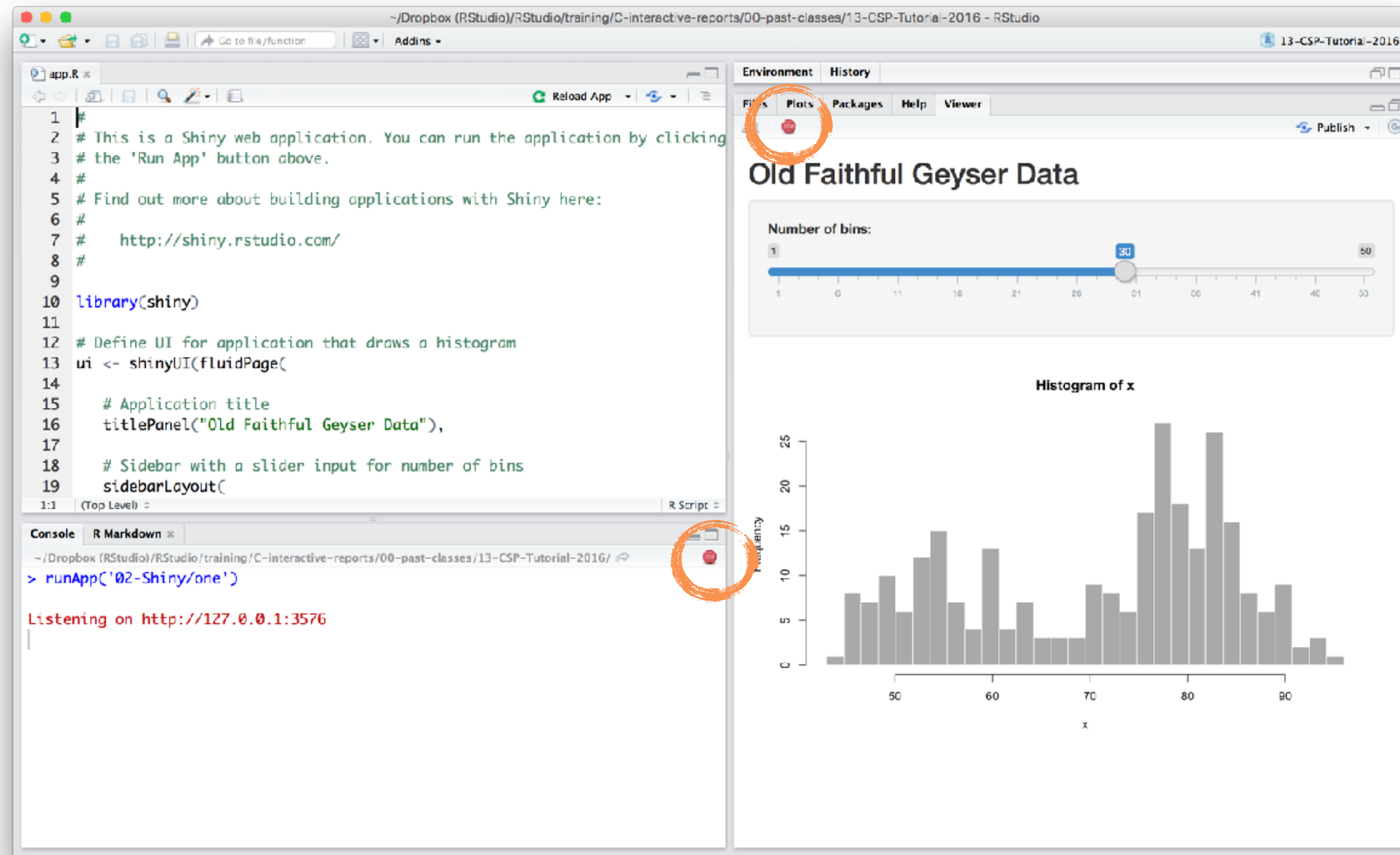


Change display

A screenshot of the RStudio interface. The top toolbar shows the 'Run App' button (a green play icon) circled in orange. A dropdown menu is open, showing three options: 'Run in Window', 'Run in Viewer Pane' (which is selected and has a checkmark), and 'Run External'. The background shows an R script editor with a Shiny application template. The script includes comments about running the application and a URL, followed by R code for loading the 'shiny' library and defining the user interface (UI) for a histogram application titled 'Old Faithful Geyser Data'.

```
1 #
2 # This is a Shiny web application. You can run this 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 left pane displays the R script code for the application, which includes comments and the `library(shiny)` and `ui <- shinyUI(fluidPage(` lines. The right pane shows the application output, titled "Old Faithful Geyser Data", featuring a slider for "Number of bins" and a histogram of the data. Two orange circles highlight the red stop buttons: one in the top right of the application viewer and another in the bottom right of the R script editor. The console at the bottom shows the command `> runApp('02-Shiny/one')` and the message "Listening on http://127.0.0.1:3576".

```
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 )
```

Environment History
Files Plots Packages Help Viewer
Publish

Old Faithful Geyser Data

Number of bins: 1 30 50

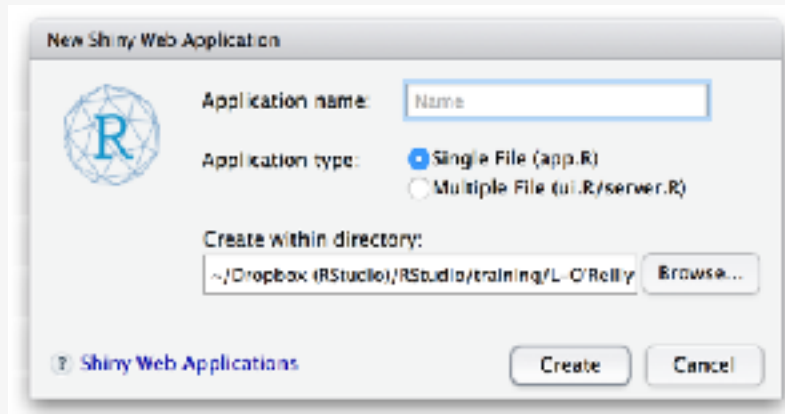
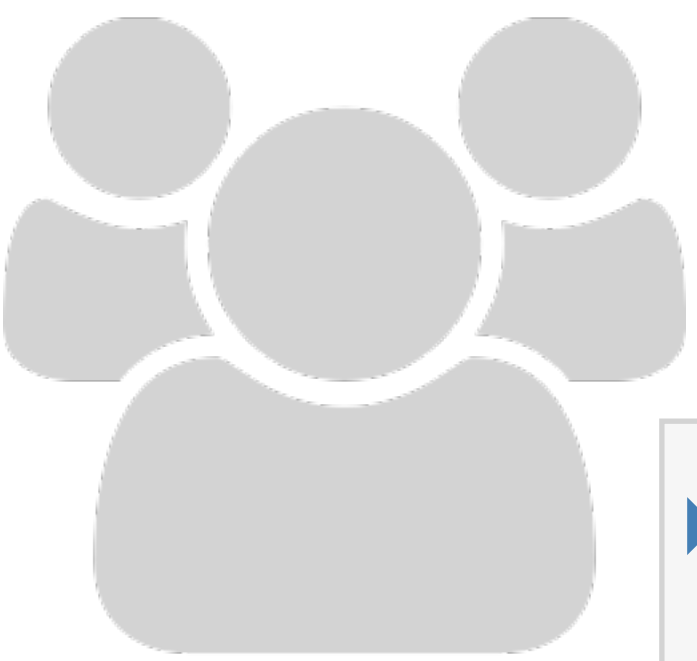
Histogram of x

Frequency

x

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

EXERCISE



 **Run App**



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

3_m 00_s

Anatomy of a Shiny app

WHAT'S IN AN 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)
```

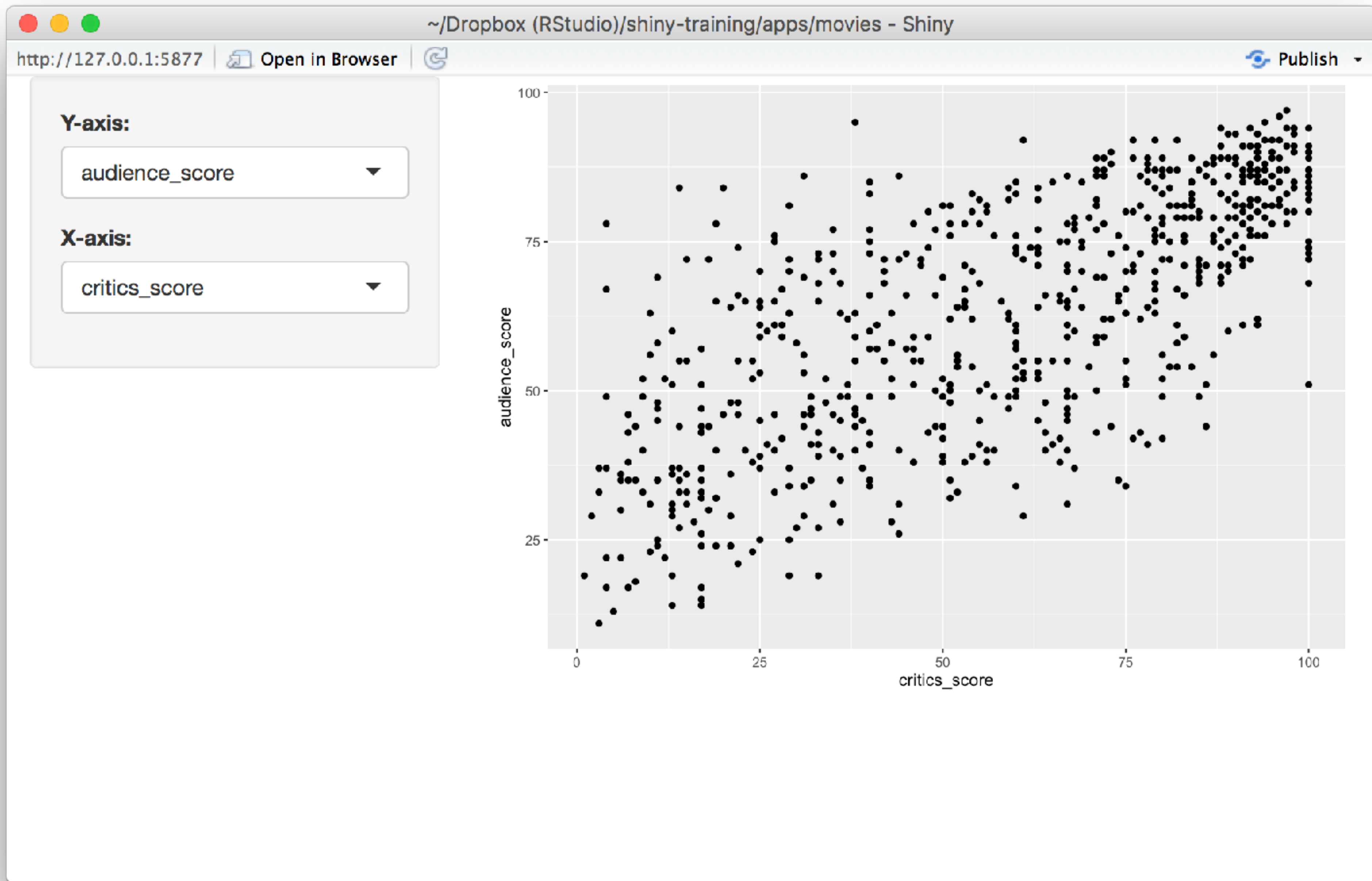


Let's build a simple movie browser app!



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

Contains instructions
needed to build app

```
# 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
```

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

```
}
```

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

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

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

Good ol' ggplot2 code,
with **inputs** from UI

Running the app

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

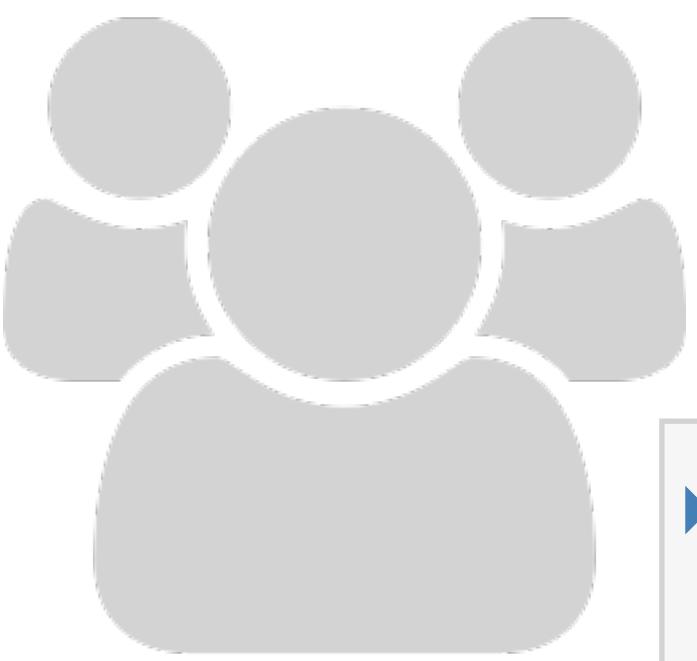


DEMO

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

5_m 00_s

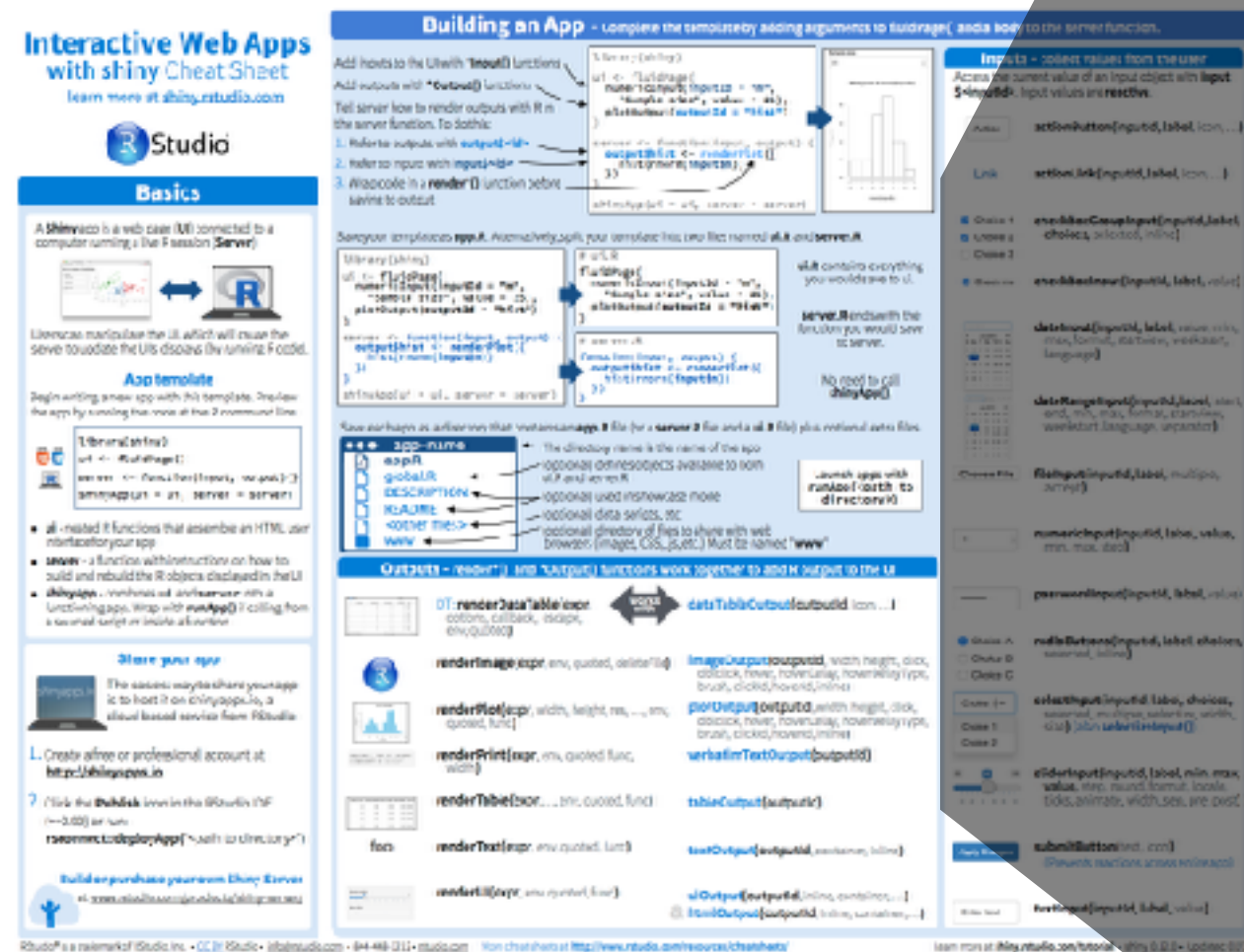


SOLUTION

Solution to the previous exercise

`movies_02.R`

INPUTS



Action

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

Link

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

- ☒ Choice 1
- ☒ Choice 2
- ☐ Choice 3

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

- ☒ Check me

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)

Choose File

fileInput(inputId, label, multiple, accept)

1

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

.....

passwordInput(inputId, label, value)

- ☒ Choice A
- ☐ Choice B
- ☐ Choice C

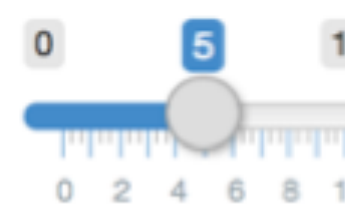
radioButtons(inputId, label, choices, selected, inline)

Choice 1 | ▲

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

Choice 1

Choice 2



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

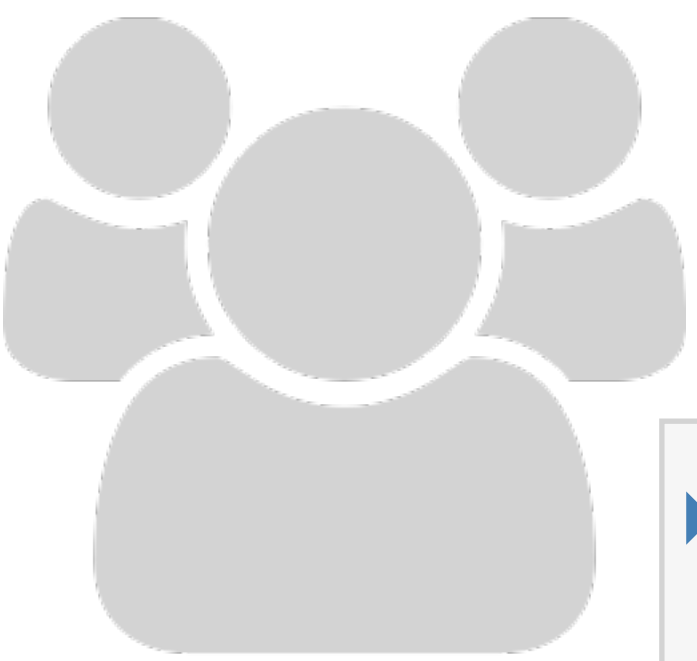
Apply Changes

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

Enter text

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

5_m 00_s



SOLUTION

Solution to the previous exercise

`movies_03.R`

OUTPUTS

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

Studio

Basics

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

Users can manipulate the UI, which will cause the server to update the UIs displayed by the client.

App template

Begin writing a shiny app with this template. Deploy the app by clicking the icon at the bottom right.

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

• all-in-one R functions that assemble an HTML user interface for your app

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

• **server** - a function that assembles an R function with instructions on how to build and rebuild the R objects displayed in the UI

• **shinyApp** - combines ui and server into a single function that can be called from a terminal script or inside a R script

Store your app

If the session is a shiny app, it is stored in the shinyapps.io cloud-based service from RStudio.

1. Create a free or professional account at <https://shinyapps.io>

2. Push the Shiny app to the shinyapps.io cloud-based service from RStudio.

3. Build and push your shiny app to shinyapps.io

4. Build and push your shiny app to shinyapps.io

5. Build and push your shiny app to shinyapps.io

6. Build and push your shiny app to shinyapps.io

7. Build and push your shiny app to shinyapps.io

8. Build and push your shiny app to shinyapps.io

9. Build and push your shiny app to shinyapps.io

10. Build and push your shiny app to shinyapps.io

Building an App - Learn the shiny workflow: how to build an app, how to build an app, how to build an app.

1. Add a new Shiny app to the Shiny workflow.

2. Add a new Shiny app to the Shiny workflow.

3. Add a new Shiny app to the Shiny workflow.

4. Add a new Shiny app to the Shiny workflow.

5. Add a new Shiny app to the Shiny workflow.

6. Add a new Shiny app to the Shiny workflow.

7. Add a new Shiny app to the Shiny workflow.

8. Add a new Shiny app to the Shiny workflow.

9. Add a new Shiny app to the Shiny workflow.

10. Add a new Shiny app to the Shiny workflow.

11. Add a new Shiny app to the Shiny workflow.

12. Add a new Shiny app to the Shiny workflow.

13. Add a new Shiny app to the Shiny workflow.

14. Add a new Shiny app to the Shiny workflow.

15. Add a new Shiny app to the Shiny workflow.

16. Add a new Shiny app to the Shiny workflow.

17. Add a new Shiny app to the Shiny workflow.

18. Add a new Shiny app to the Shiny workflow.

19. Add a new Shiny app to the Shiny workflow.

20. Add a new Shiny app to the Shiny workflow.

21. Add a new Shiny app to the Shiny workflow.

22. Add a new Shiny app to the Shiny workflow.

23. Add a new Shiny app to the Shiny workflow.

24. Add a new Shiny app to the Shiny workflow.

25. Add a new Shiny app to the Shiny workflow.

26. Add a new Shiny app to the Shiny workflow.

27. Add a new Shiny app to the Shiny workflow.

28. Add a new Shiny app to the Shiny workflow.

29. Add a new Shiny app to the Shiny workflow.

30. Add a new Shiny app to the Shiny workflow.

31. Add a new Shiny app to the Shiny workflow.

32. Add a new Shiny app to the Shiny workflow.

33. Add a new Shiny app to the Shiny workflow.

34. Add a new Shiny app to the Shiny workflow.

35. Add a new Shiny app to the Shiny workflow.

36. Add a new Shiny app to the Shiny workflow.

37. Add a new Shiny app to the Shiny workflow.

38. Add a new Shiny app to the Shiny workflow.

39. Add a new Shiny app to the Shiny workflow.

40. Add a new Shiny app to the Shiny workflow.

41. Add a new Shiny app to the Shiny workflow.

42. Add a new Shiny app to the Shiny workflow.

43. Add a new Shiny app to the Shiny workflow.

44. Add a new Shiny app to the Shiny workflow.

45. Add a new Shiny app to the Shiny workflow.

46. Add a new Shiny app to the Shiny workflow.

47. Add a new Shiny app to the Shiny workflow.

48. Add a new Shiny app to the Shiny workflow.

49. Add a new Shiny app to the Shiny workflow.

50. Add a new Shiny app to the Shiny workflow.

51. Add a new Shiny app to the Shiny workflow.

52. Add a new Shiny app to the Shiny workflow.

53. Add a new Shiny app to the Shiny workflow.

54. Add a new Shiny app to the Shiny workflow.

55. Add a new Shiny app to the Shiny workflow.

56. Add a new Shiny app to the Shiny workflow.

57. Add a new Shiny app to the Shiny workflow.

58. Add a new Shiny app to the Shiny workflow.

59. Add a new Shiny app to the Shiny workflow.

60. Add a new Shiny app to the Shiny workflow.

61. Add a new Shiny app to the Shiny workflow.

62. Add a new Shiny app to the Shiny workflow.

63. Add a new Shiny app to the Shiny workflow.

64. Add a new Shiny app to the Shiny workflow.

65. Add a new Shiny app to the Shiny workflow.

66. Add a new Shiny app to the Shiny workflow.

67. Add a new Shiny app to the Shiny workflow.

68. Add a new Shiny app to the Shiny workflow.

69. Add a new Shiny app to the Shiny workflow.

70. Add a new Shiny app to the Shiny workflow.

71. Add a new Shiny app to the Shiny workflow.

72. Add a new Shiny app to the Shiny workflow.

73. Add a new Shiny app to the Shiny workflow.

74. Add a new Shiny app to the Shiny workflow.

75. Add a new Shiny app to the Shiny workflow.

76. Add a new Shiny app to the Shiny workflow.

77. Add a new Shiny app to the Shiny workflow.

78. Add a new Shiny app to the Shiny workflow.

79. Add a new Shiny app to the Shiny workflow.

80. Add a new Shiny app to the Shiny workflow.

81. Add a new Shiny app to the Shiny workflow.

82. Add a new Shiny app to the Shiny workflow.

83. Add a new Shiny app to the Shiny workflow.

84. Add a new Shiny app to the Shiny workflow.

85. Add a new Shiny app to the Shiny workflow.

86. Add a new Shiny app to the Shiny workflow.

87. Add a new Shiny app to the Shiny workflow.

88. Add a new Shiny app to the Shiny workflow.

89. Add a new Shiny app to the Shiny workflow.

90. Add a new Shiny app to the Shiny workflow.

91. Add a new Shiny app to the Shiny workflow.

92. Add a new Shiny app to the Shiny workflow.

93. Add a new Shiny app to the Shiny workflow.

94. Add a new Shiny app to the Shiny workflow.

95. Add a new Shiny app to the Shiny workflow.

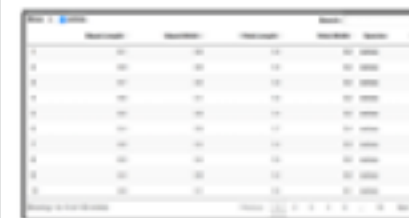
96. Add a new Shiny app to the Shiny workflow.

97. Add a new Shiny app to the Shiny workflow.

98. Add a new Shiny app to the Shiny workflow.

99. Add a new Shiny app to the Shiny workflow.

100. Add a new Shiny app to the Shiny workflow.



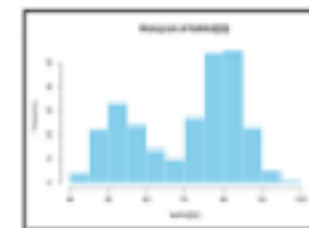
DT::renderDataTable(expr,
options, callback, escape,
env, quoted)



dataTableOutput(outputId, icon, ...)



renderImage(expr, env, quoted, deleteFile)



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

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

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

Report Length	Report Width	Plot Length	Plot Width	Report
1	1.10	1.10	1.10	Report
2	1.10	1.10	1.10	Report
3	1.10	1.10	1.10	Report
4	1.10	1.10	1.10	Report
5	1.10	1.10	1.10	Report
6	1.10	1.10	1.10	Report
7	1.10	1.10	1.10	Report
8	1.10	1.10	1.10	Report
9	1.10	1.10	1.10	Report
10	1.10	1.10	1.10	Report

renderText(expr, env, quoted, func)

foo

renderUI(expr, env, quoted, func)



imageOutput(outputId, width, height, click,
dbclick, hover, hoverDelay, hoverDelayType,
brush, clickId, hoverId, inline)

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

verbatimTextOutput(outputId)

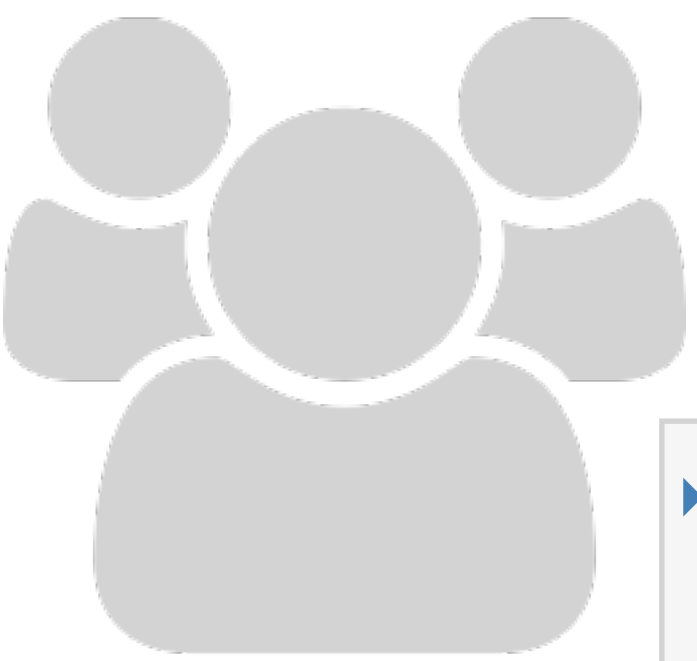
tableOutput(outputId)

textOutput(outputId, container, inline)

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

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

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

5_m 00_s



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

5_m 00_s



SOLUTION

Solution to the previous exercise

`movies_05.R`

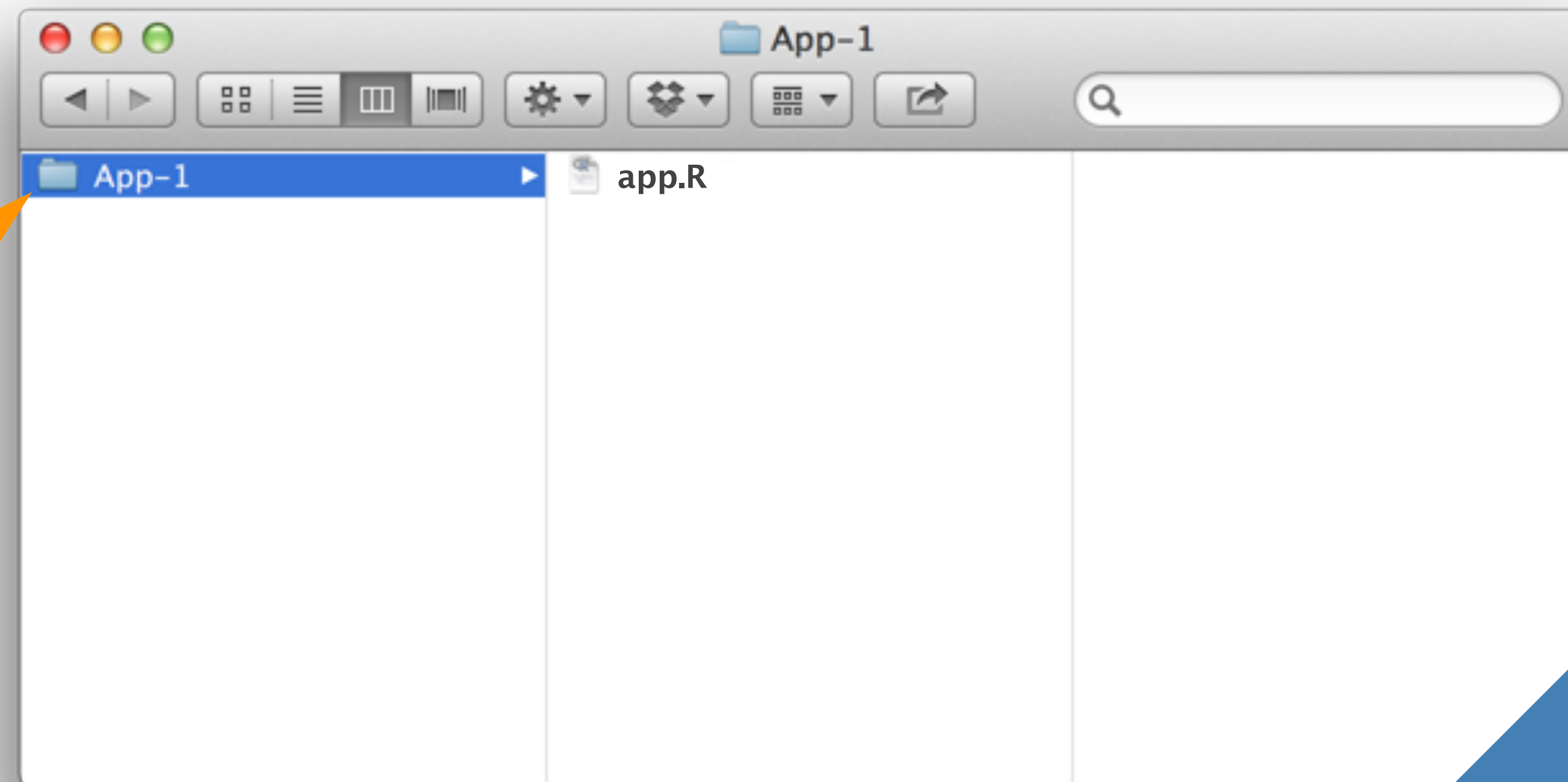
File structure

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 workshop**

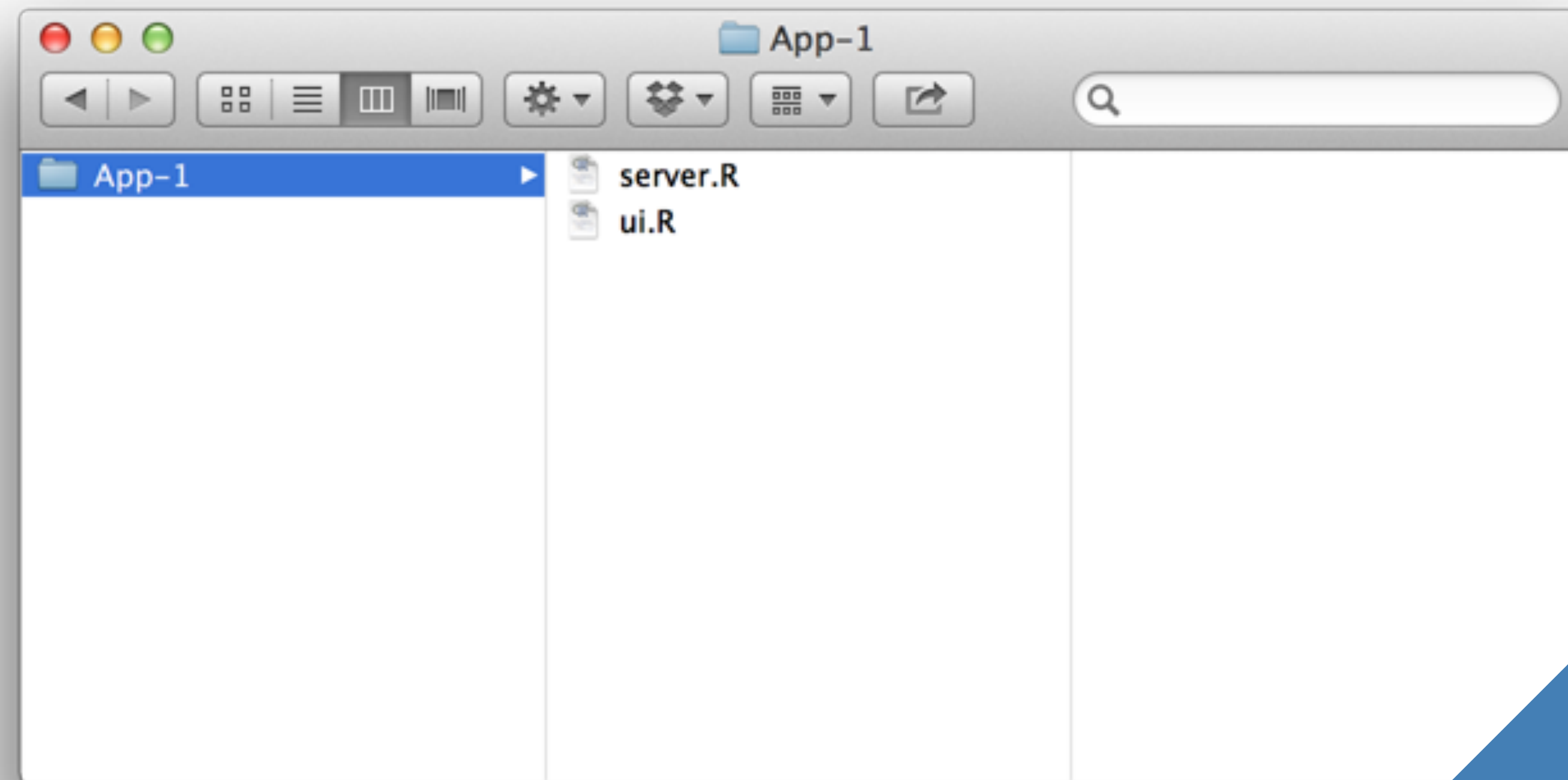


**You must use this
exact name (app.R)
for deploying the app**

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

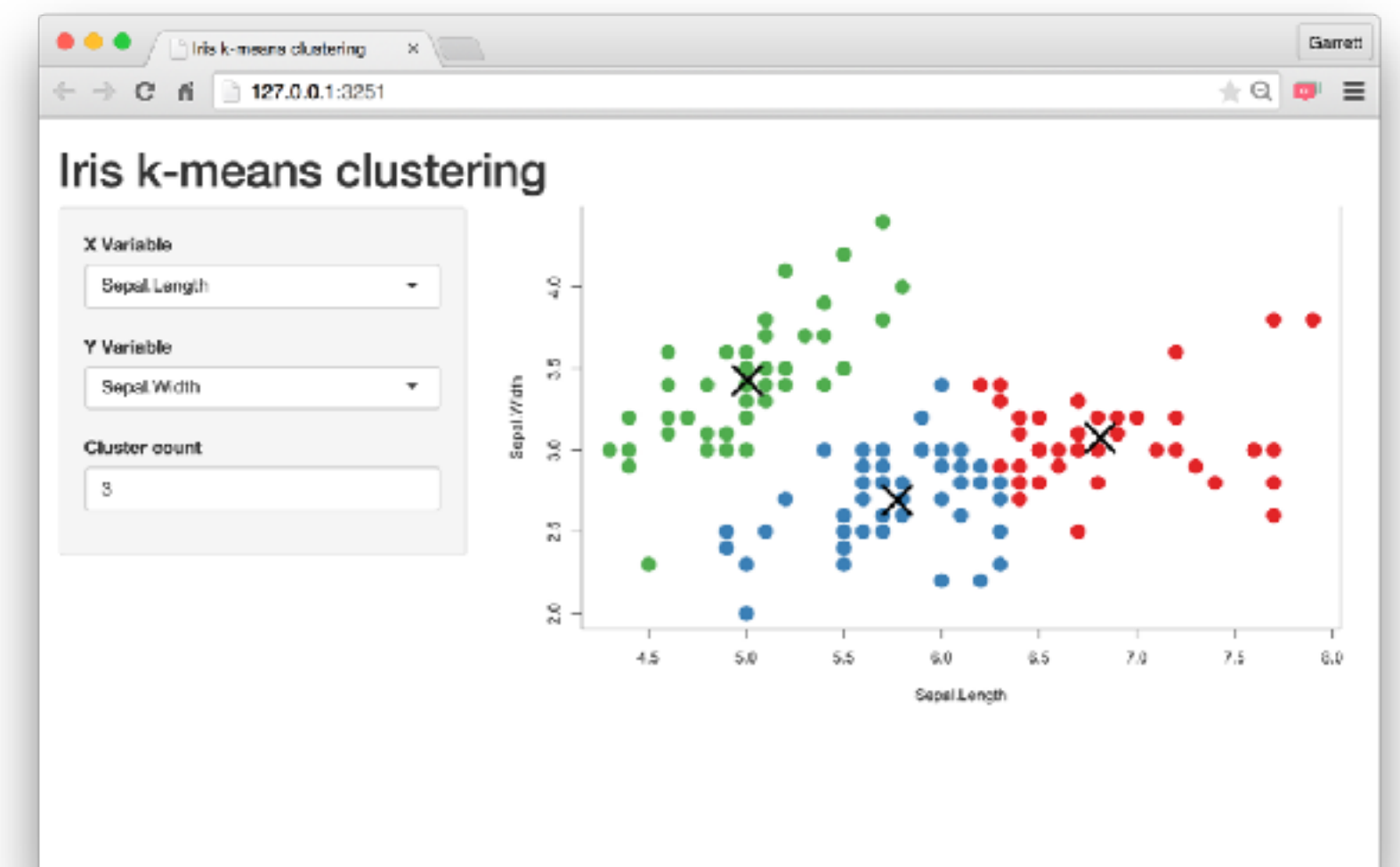
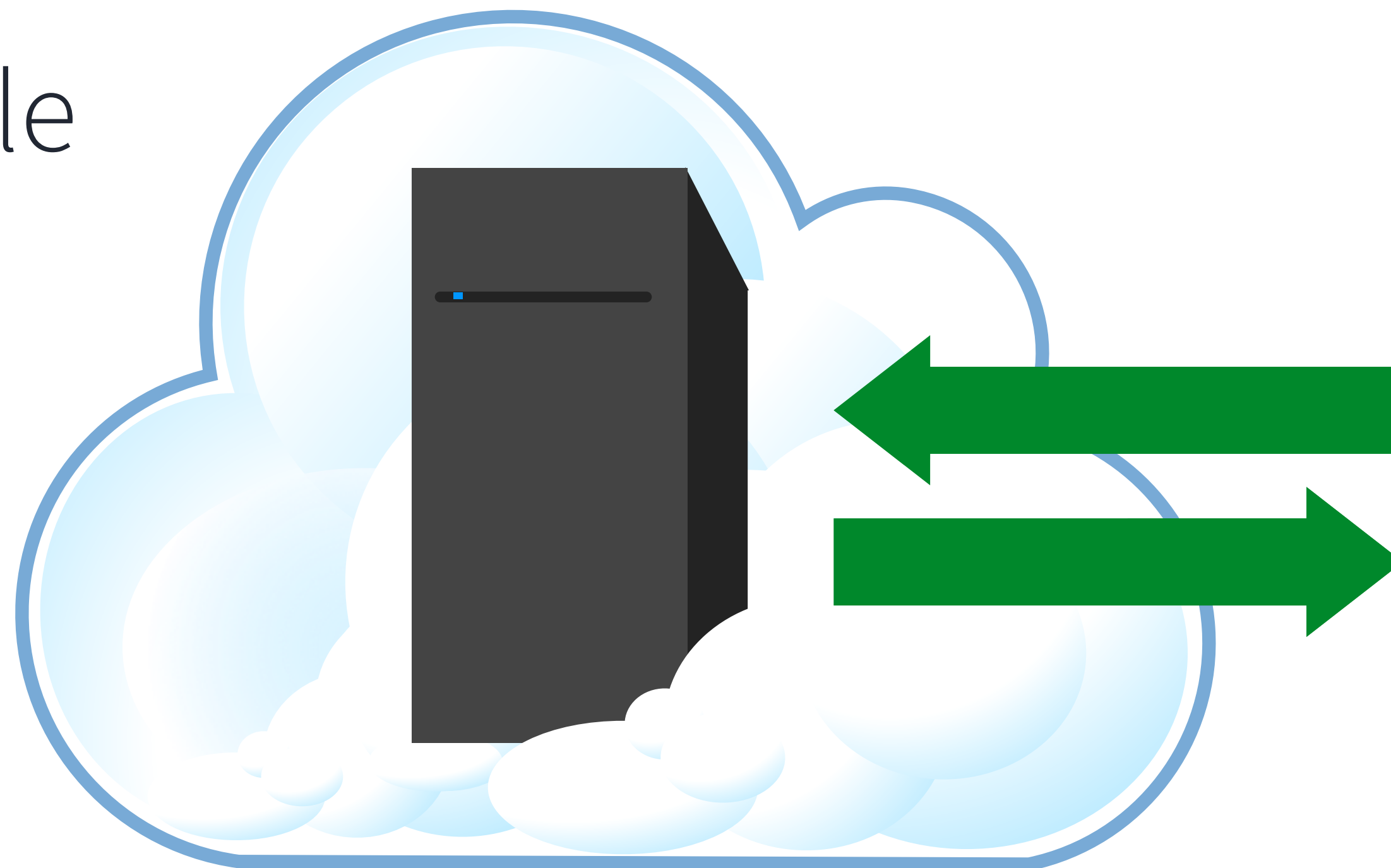
shinyapps.io



SHINYAPPS.IO

A server maintained by RStudio

- ▶ easy to use
- ▶ secure
- ▶ scalable



HASSLE-FREE CLOUD HOSTING FOR SHINY

shinyapps.io by RStudio

[Home](#)

[Features](#)

[Pricing](#)

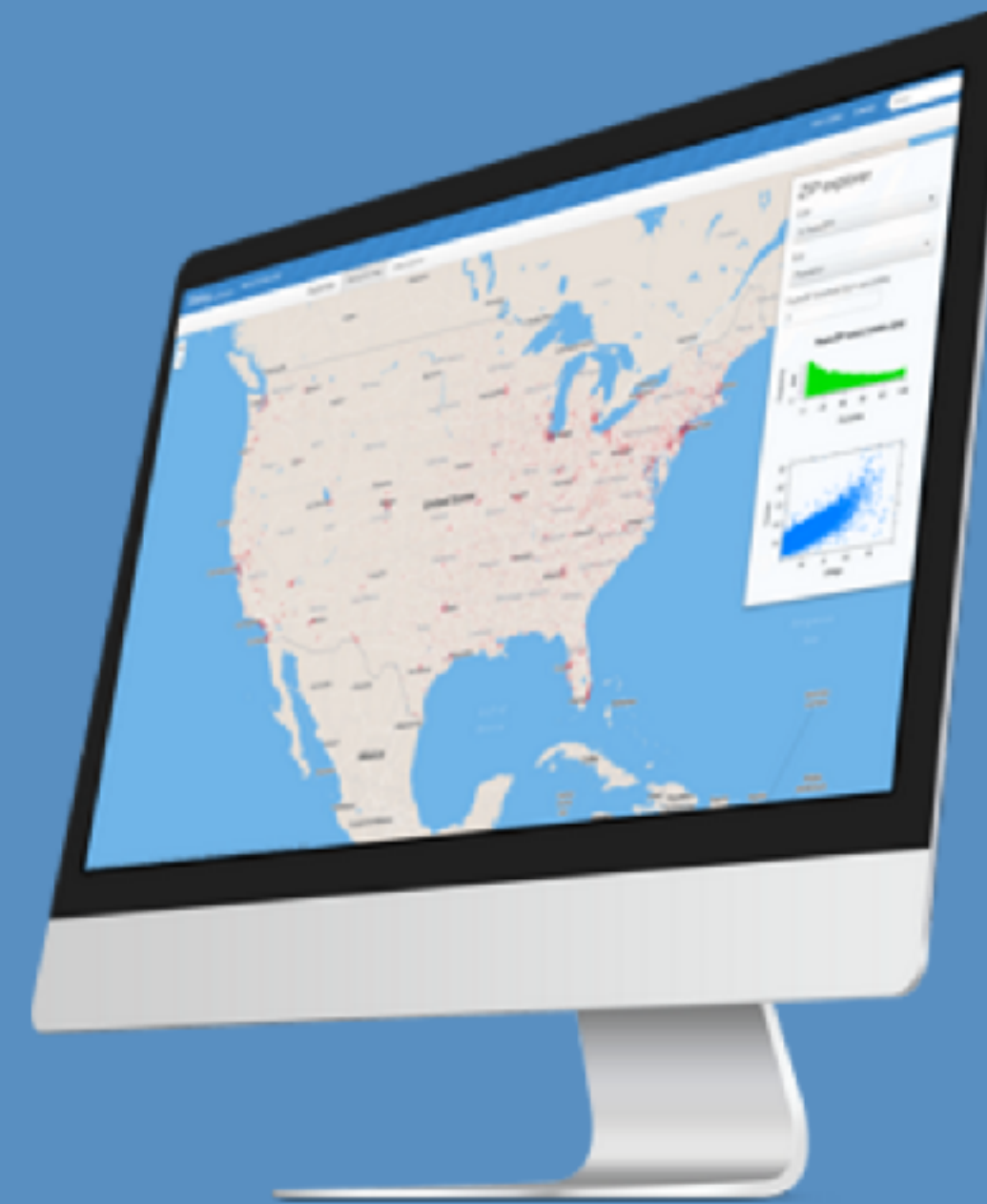
[Support](#)

[Log In](#)

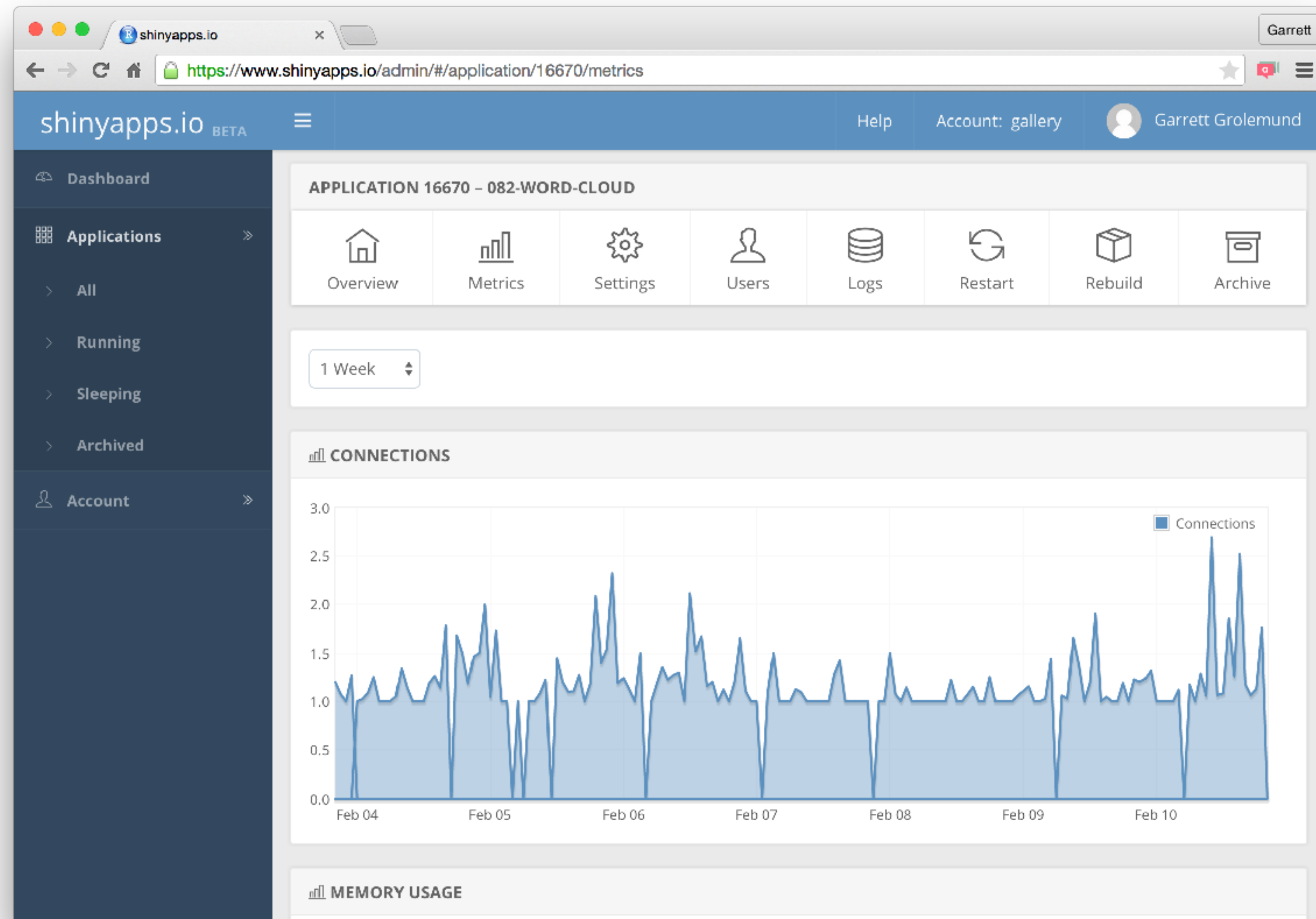
Share your Shiny Applications Online

Deploy your Shiny applications on the Web in minutes

[Sign Up](#)



WITH BUILT-IN METRICS



MEMBERSHIP PRICING

FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
\$0 /month	\$9 /month (or \$100/year)	\$39 /month (or \$440/year)	\$99 /month (or \$1,100/year)	\$299 /month (or \$3,300/year)
New to Shiny? Deploy your applications for FREE.	More applications. More active hours!	Take your users to the next level!	Password protection? Authenticate your users!	Professional has it all! Personalize your domains.
5 Applications	25 Applications	Unlimited Applications	Unlimited Applications	Unlimited Applications
25 <u>Active Hours</u>	100 <u>Active Hours</u>	500 <u>Active Hours</u>	2,000 <u>Active Hours</u>	10,000 <u>Active Hours</u>
✓ Community Support	✓ Premium Support	✓ <u>Performance Boost</u>	✓ Authentication	✓ Authentication
📄 <u>RStudio Branding</u>		✓ Premium Support	✓ <u>Performance Boost</u>	✓ <u>Account Sharing</u>
			✓ Premium Support	✓ <u>Performance Boost</u>
				✓ <u>Custom Domains</u>
				✓ Premium Support

Build your own server



SHINY SERVER

rstudio.com/products/shiny/shiny-server/

- 
- ✓ **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**

Free &
open
source

SHINY SERVER PRO

rstudio.com/products/shiny/shiny-server/

- 
- ✓ **Secure access**
LDAP, GoogleAuth, SSL, and more
 - ✓ **Performance**
fine tune at app and server level
 - ✓ **Management**
monitor and control resource use
 - ✓ **Support**
direct priority support

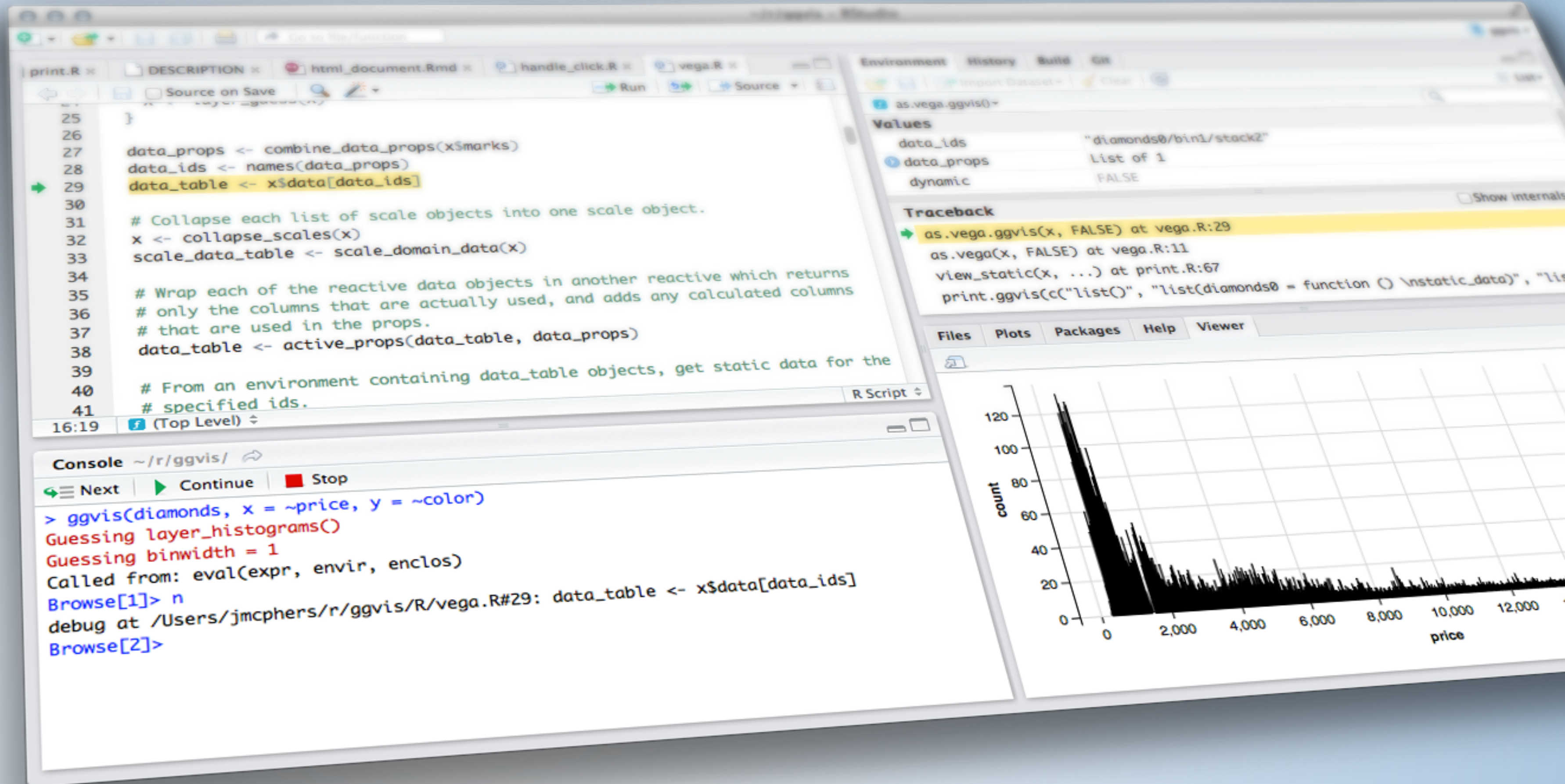
45 day
evaluation
free trial

RSTUDIO CONNECT

rstudio.com/products/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

45 day
evaluation
free trial



A FAST INTRODUCTION

TO SHINY

Shiny from  Studio™