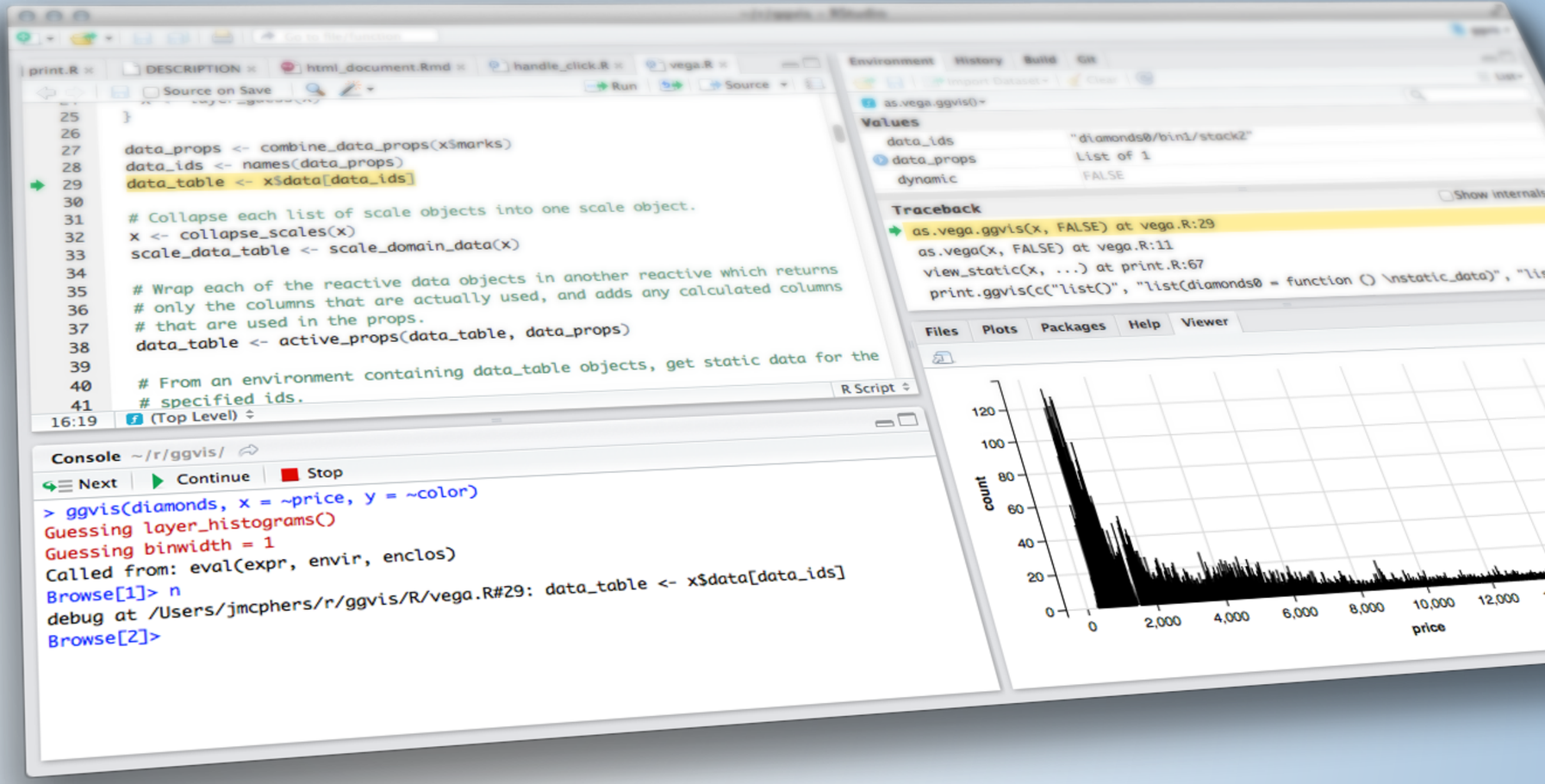


A FAST INTRODUCTION

TO SHINY

Shiny from RStudio™

BACKGROUND

- You know what a Shiny app is, and may have even made a few of them.
- You are familiar with R as a programming language.

SHINY SHOWCASE

Shiny User Showcase

Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts. Our users create fantastic examples, and some have shared them with the community. Here are some examples that we particularly like.

Shiny Apps for the Enterprise

MARKETING EFFECTS

See the effects of your marketing campaigns.

LOCATION TRACKER

Track locations over time with streaming data.

DOWNLOAD MONITOR

Streaming download rates visualized as a bubble chart.

PERSISTENT STORAGE

Save data from your apps to local files, servers, databases, and more.

Industry Specific Shiny Apps

TOURISM DASHBOARD

GENOME BROWSER

ER OPTIMIZATION

SUPPLY AND DEMAND

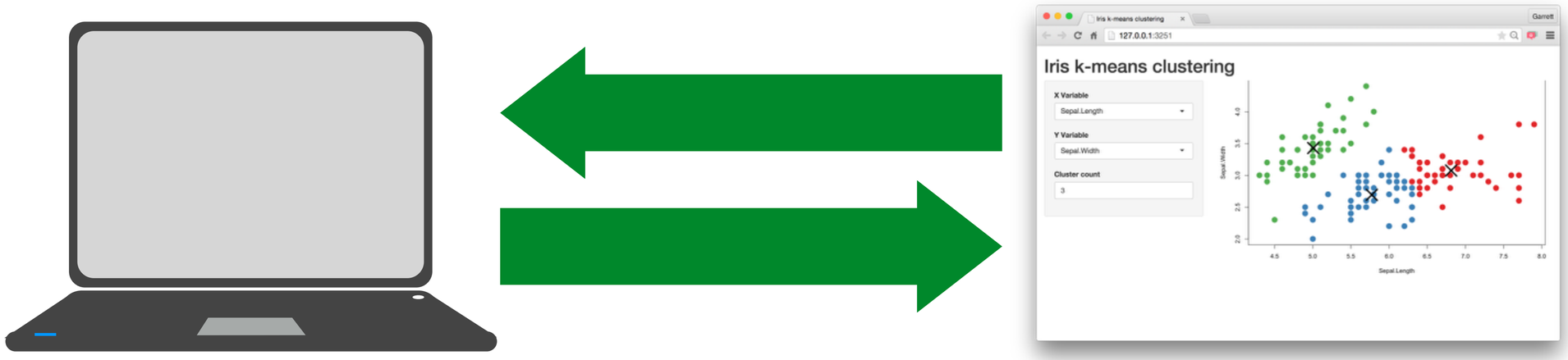
https://www.rstudio.com/products/shiny/shiny-user-showcase/

OUTLINE

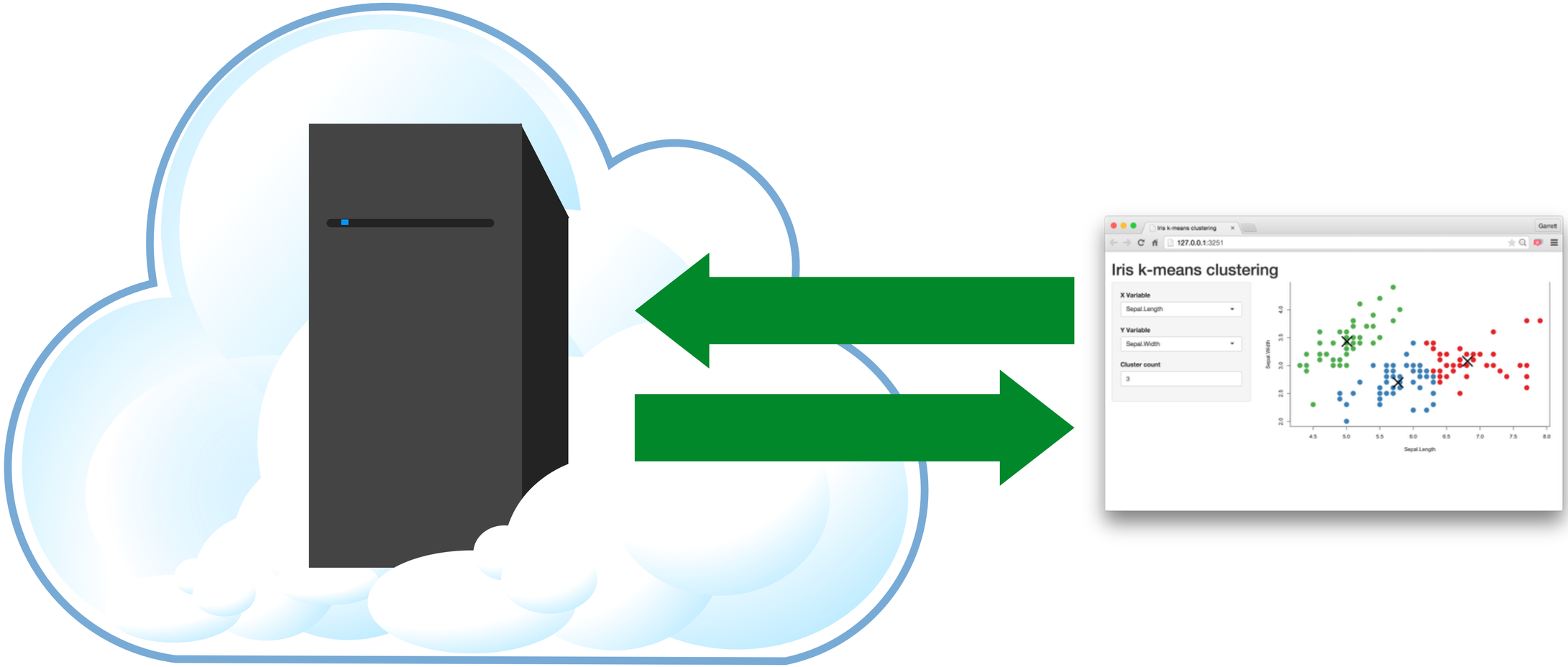
- High level view
- Anatomy of a Shiny app
  - User interface
  - Server function
  - Running the app
- File Structure
- Sharing your app

High level view

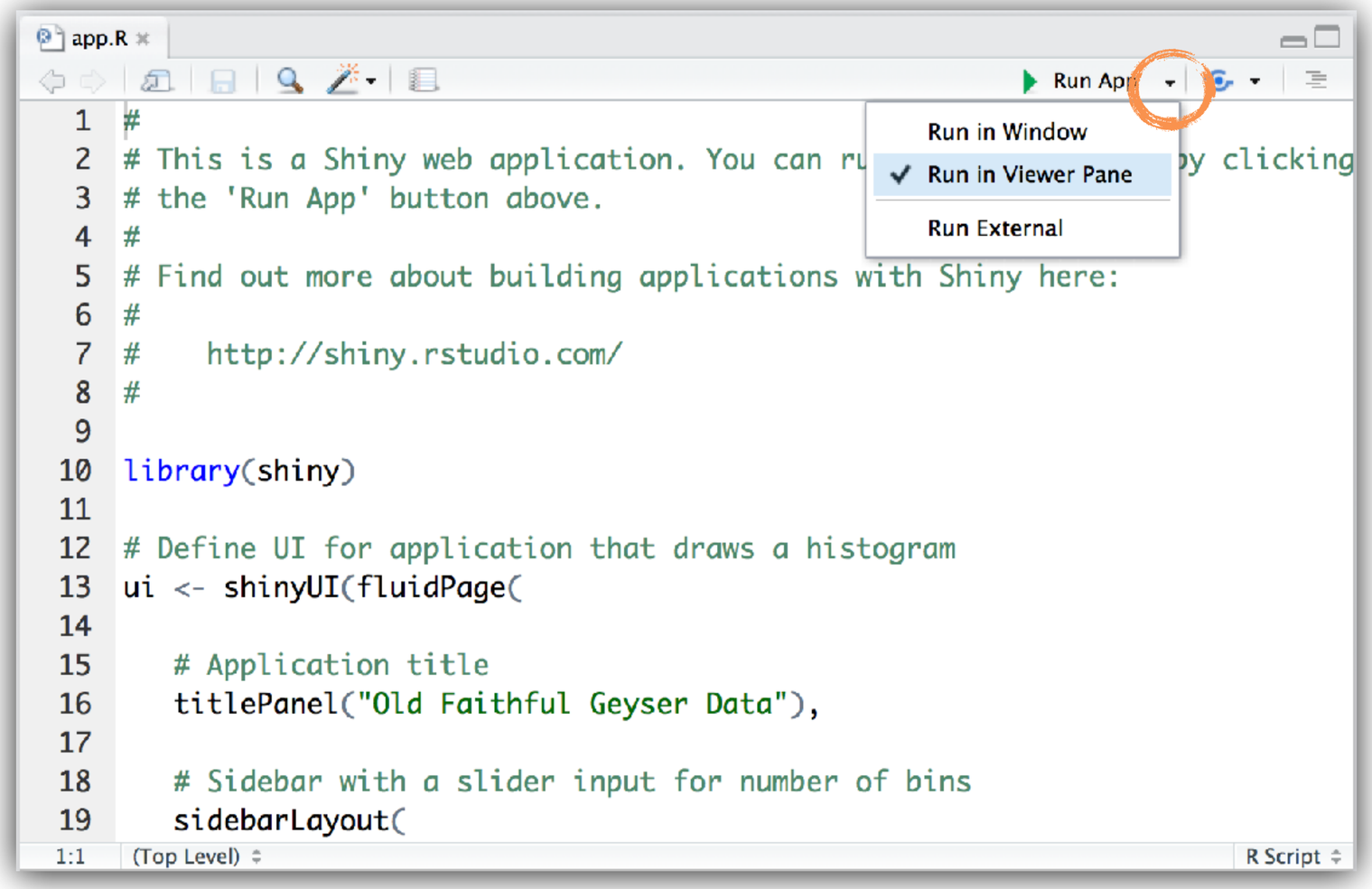
Every Shiny app is maintained by a computer running R



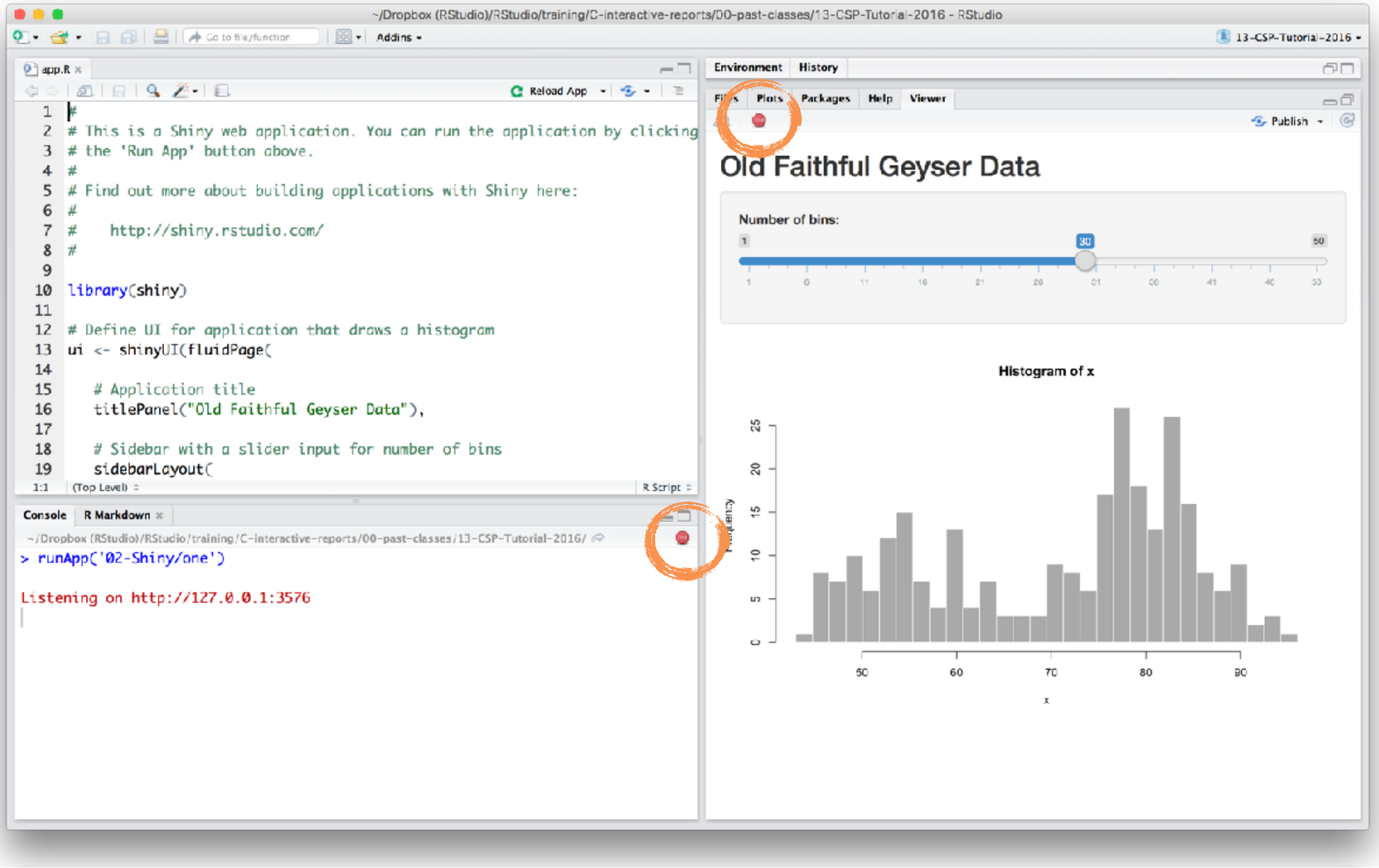
Every Shiny app is maintained by a computer running R



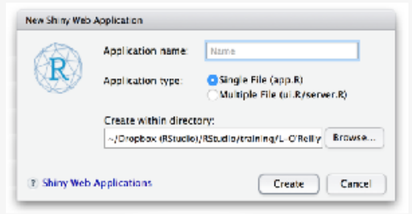
Change display



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)
```

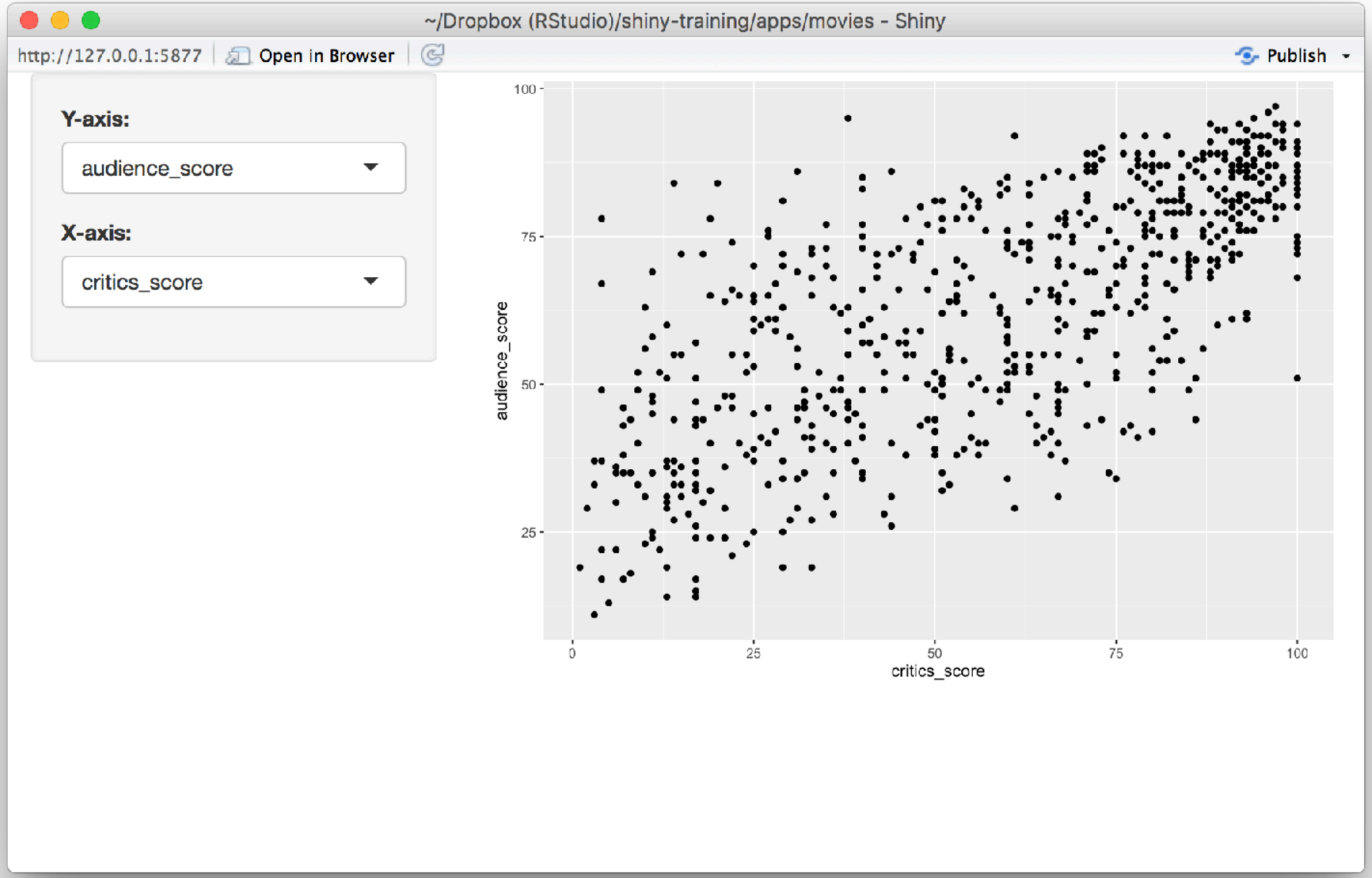
User interface
controls the layout and appearance of app

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")
    )
  )
)
```

```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

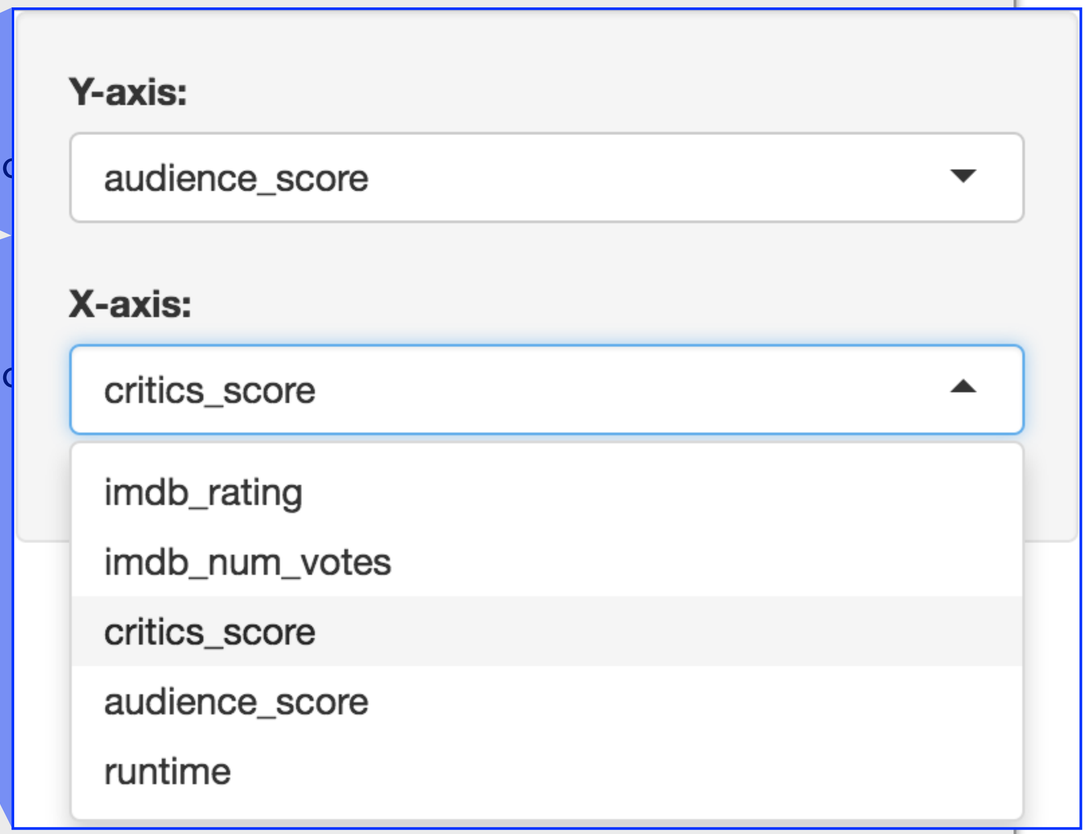
```
# Define UI for application that plots features of movies
ui <- fluidPage(

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

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

Create a layout with a sidebar and main area

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

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

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

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

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

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



```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 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) {

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

Contains instructions needed to build app

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

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

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

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

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

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

Running the app

```
# 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

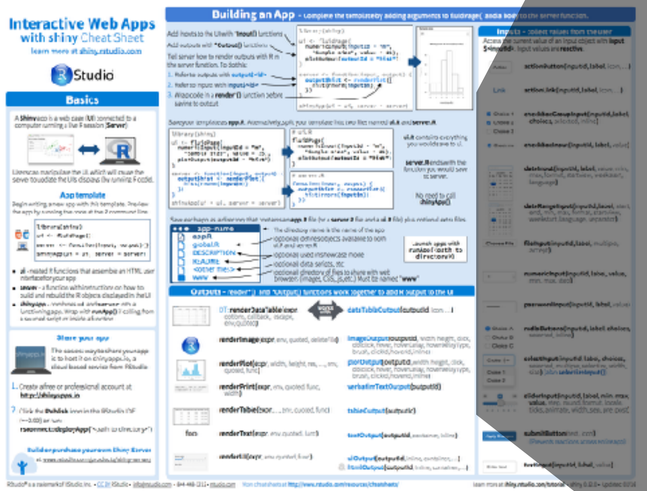
5m 00s

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)

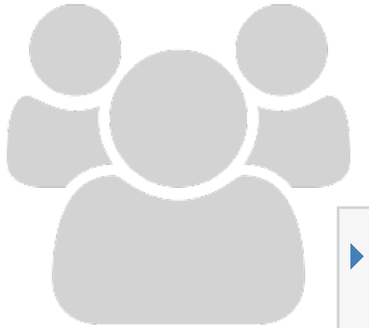
Choice A / Choice B / Choice C — **radioButtons**(inputId, label, choices, selected, inline)

Choice 1 (Choice 1, Choice 2) — **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)

0 5 10 — **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

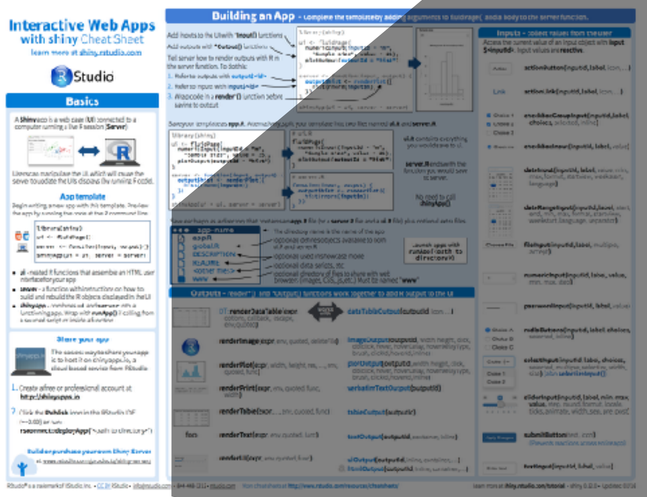
5m 00s

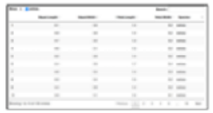
SOLUTION

Solution to the previous exercise

`movies_03.R`

OUTPUTS





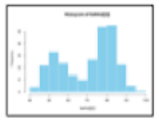
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

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

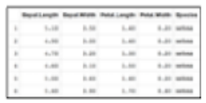


renderPlot(expr, width, height, res, …, env, quoted, func)

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

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

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, …)

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

5m 00s

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

SOLUTION

Solution to the previous exercise

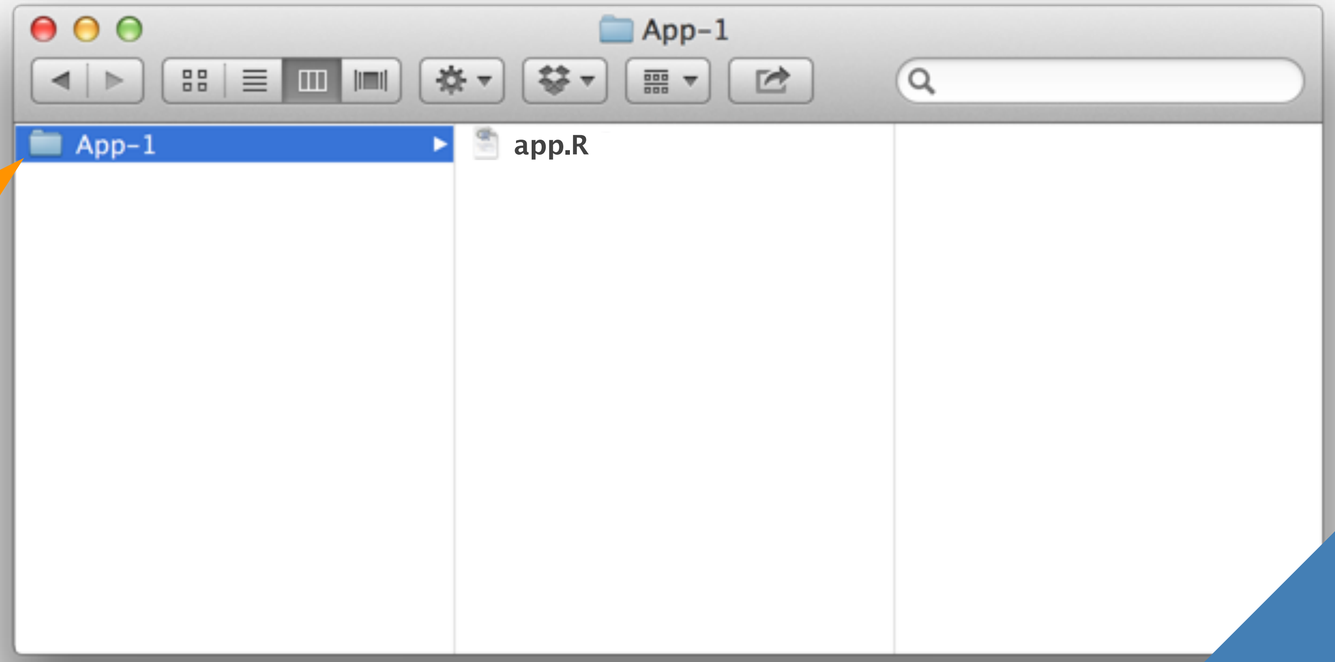
`movies_05.R`

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.



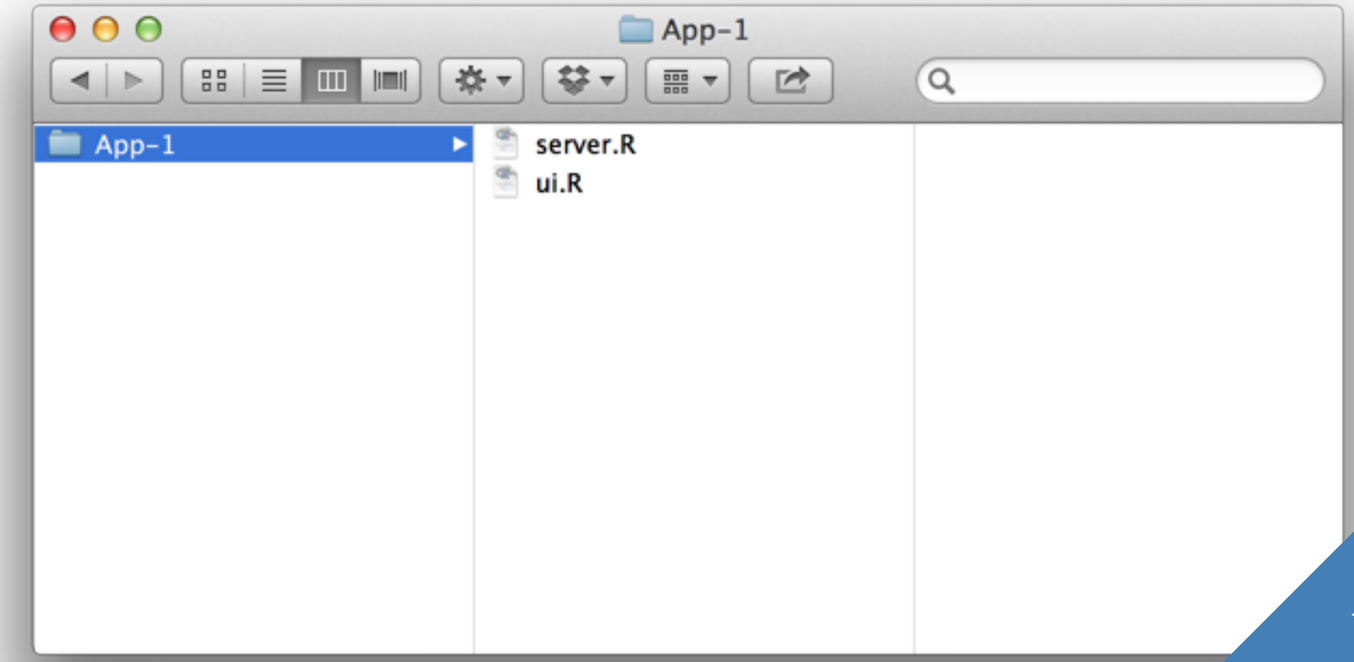
We will focus on the single file format throughout the workshop

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

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

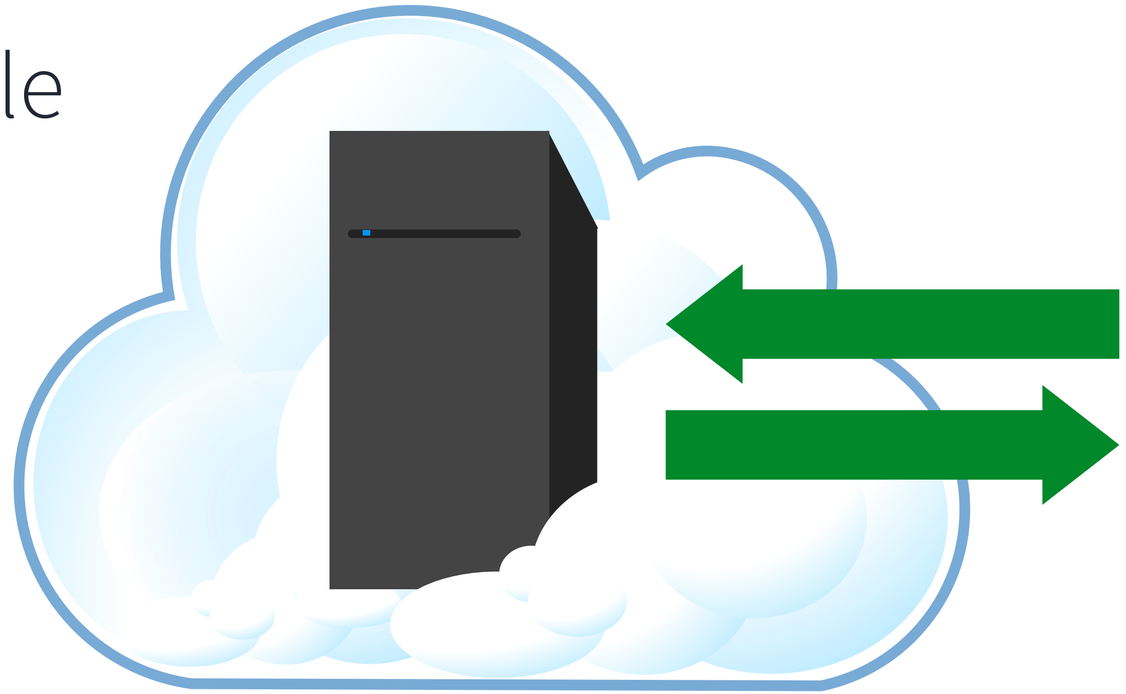
your app

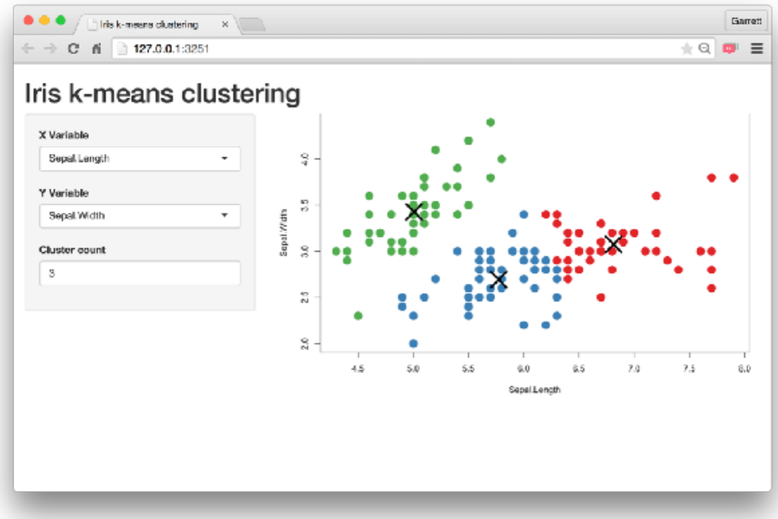
shinyapps.io

SHINYAPPS.IO

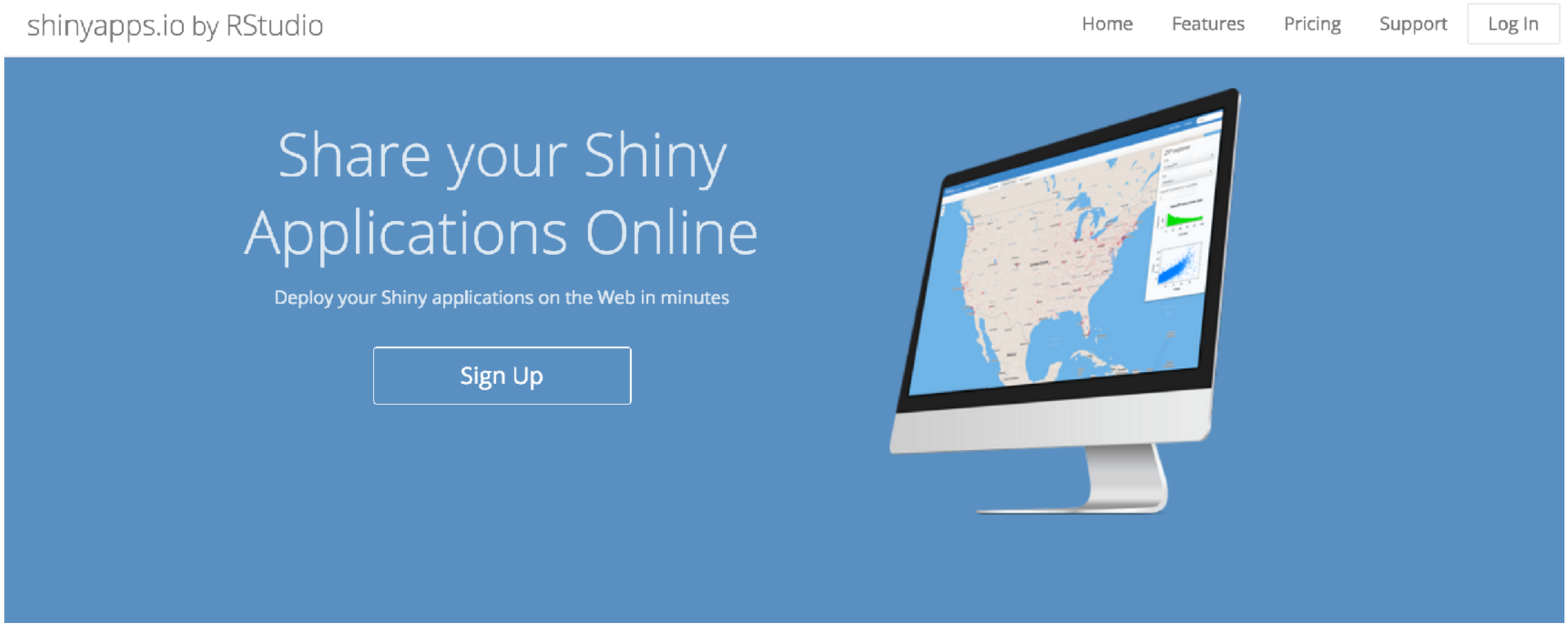
A server maintained by RStudio

- easy to use
- secure
- scalable

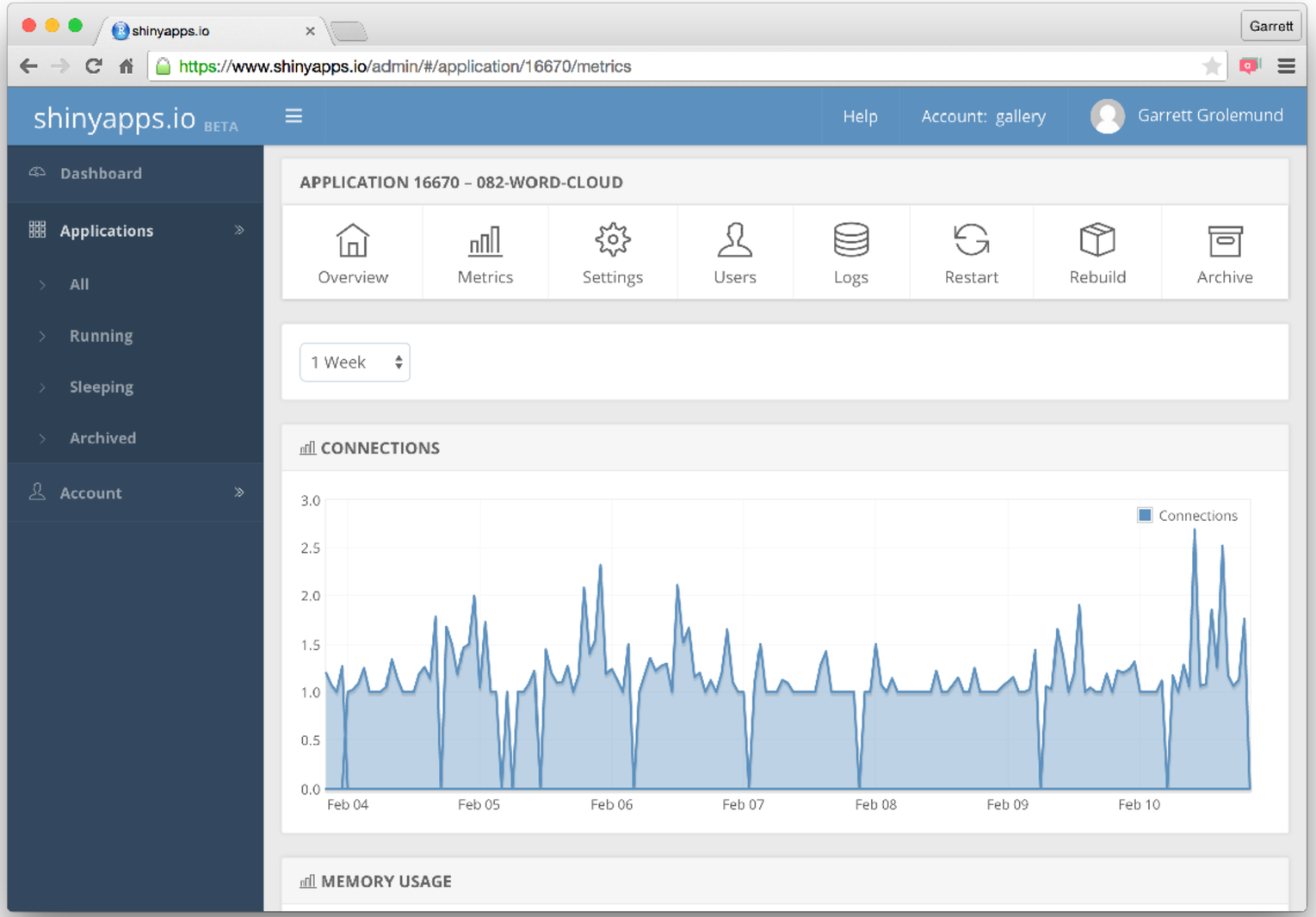




HASSLE-FREE CLOUD HOSTING FOR SHINY



WITH BUILT-IN METRICS



MEMBERSHIP PRICING

FREE

$0 /month

New to Shiny? Deploy your applications for FREE.

- **5** Applications
- **25** Active Hours
- Community Support
- RStudio Branding

STARTER

$9 /month

(or $100/year)

More applications. More active hours!

- **25** Applications
- **100** Active Hours
- Premium Support

BASIC

$39 /month

(or $440/year)

Take your users to the next level!

- **Unlimited** Applications
- **500** Active Hours
- Performance Boost
- Premium Support

STANDARD

$99 /month

(or $1,100/year)

Password protection? Authenticate your users!

- **Unlimited** Applications
- **2,000** Active Hours
- Authentication
- Performance Boost
- Premium Support

PROFESSIONAL

$299 /month

(or $3,300/year)

Professional has it all! Personalize your domains.

- **Unlimited** Applications
- **10,000** Active Hours
- Authentication
- Account Sharing
- Performance Boost
- Custom Domains
- Premium Support

Shiny from RStudio™

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/

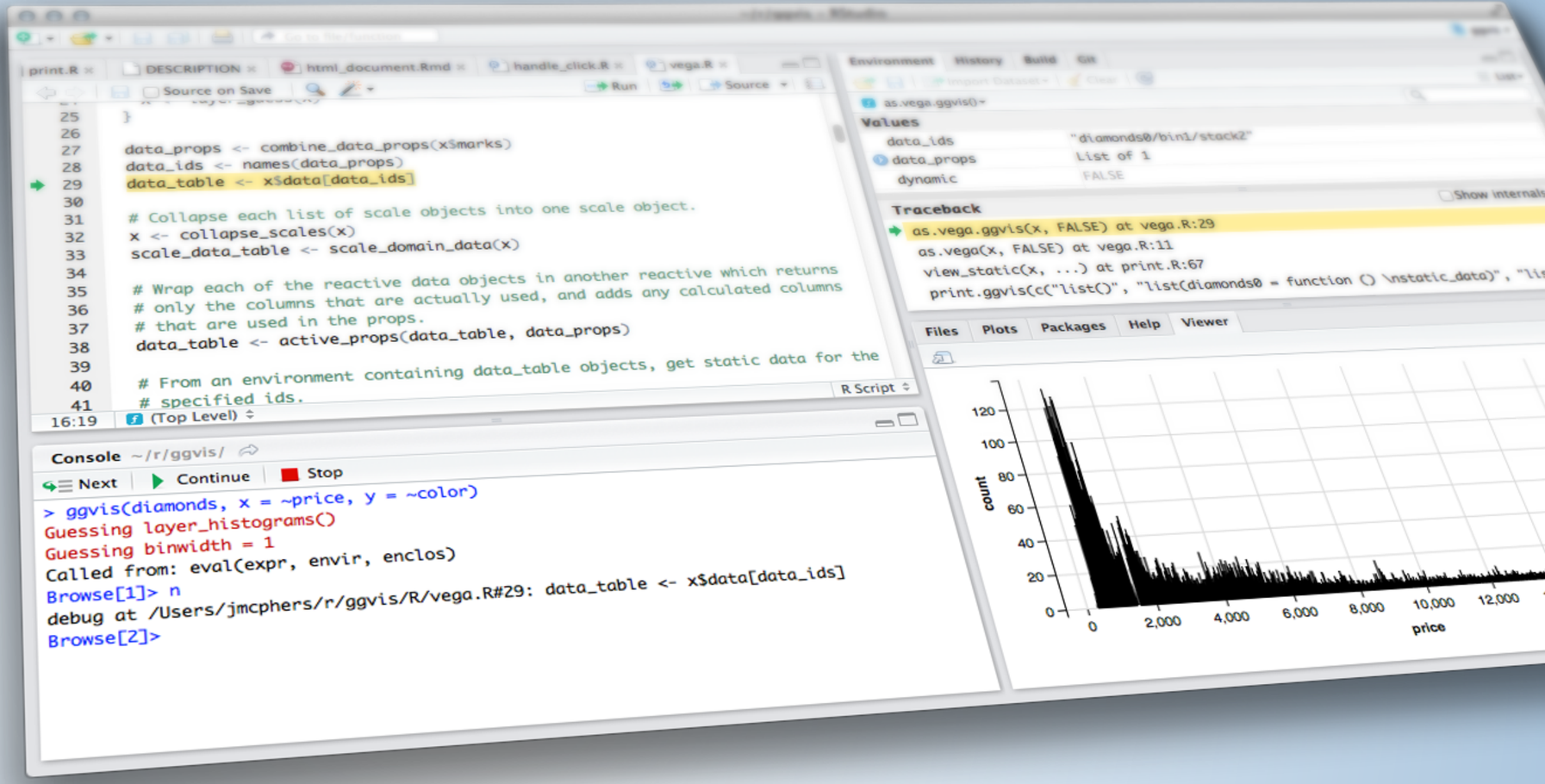
✓ **Push-button publish from RStudio**
Shiny apps, R Markdown docs, and more

✓ **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 RStudio™