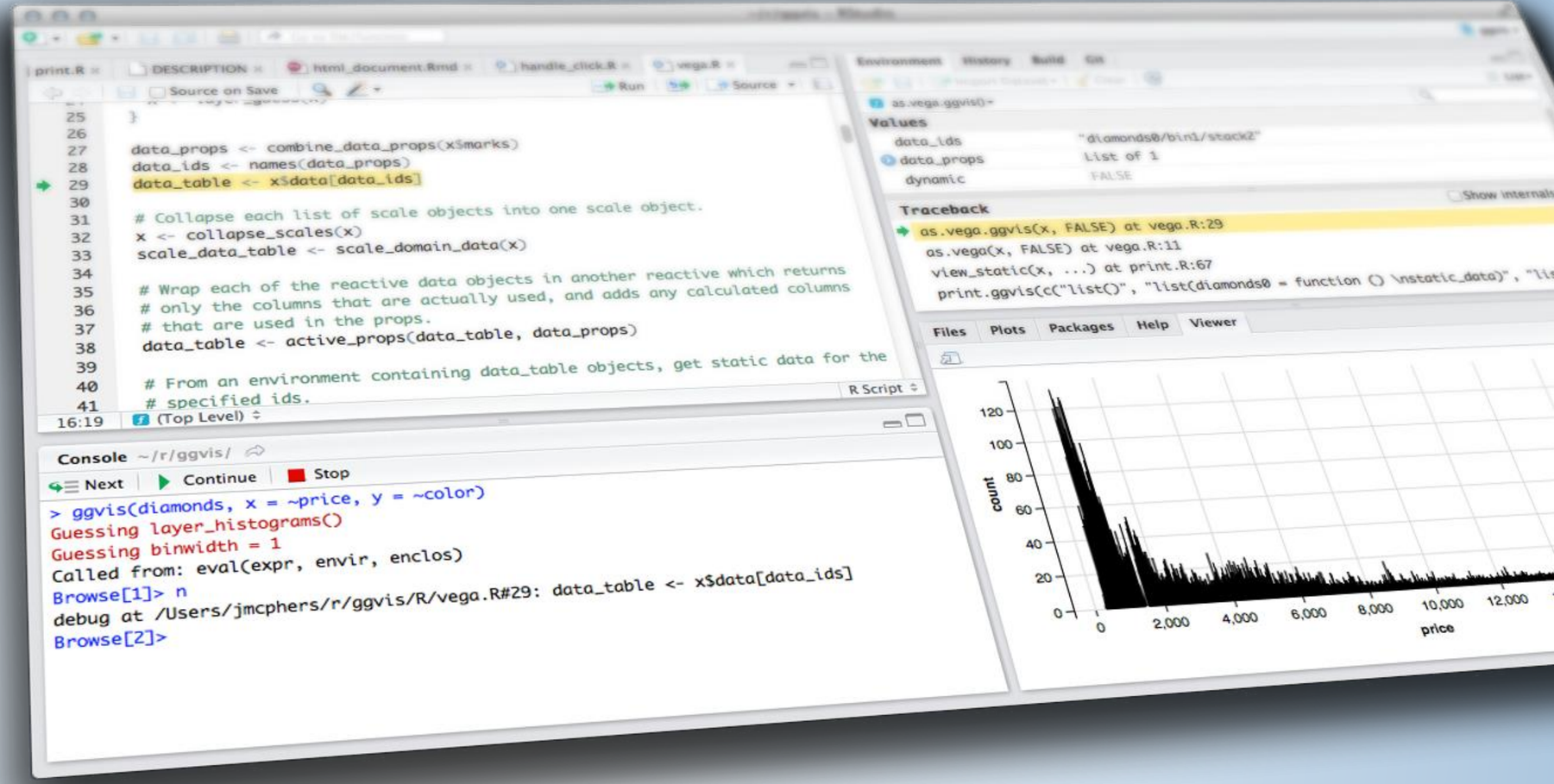


COURSE OVERVIEW

& INTRODUCTION TO SHINY

Shiny from posit

OUTLINE

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

Course Overview

HELLO

my name is

GEOFFREY ARNOLD

Geoffrey.Lloyd.Arnold@gmail.com

ABOUT ME

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

COURSE SCHEDULE

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

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

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

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

Class 5 - 8/5 – Modules & Bookmarking

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

Class 7* - 8/19 - Leaflet & LeafletProxy

Projects in RStudio

“R Projects are great.”

–Geoffrey Arnold

R PROJECTS

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

HOW DO PROJECTS WORK?

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

EXERCISE

- Create a “New Project”
  - Select “New Directory”
    - Select "New Project”
      - Make sure the “Create a git repository” is selected
      - Give the project any name you want
        - Click “Create Project”

5m 00s

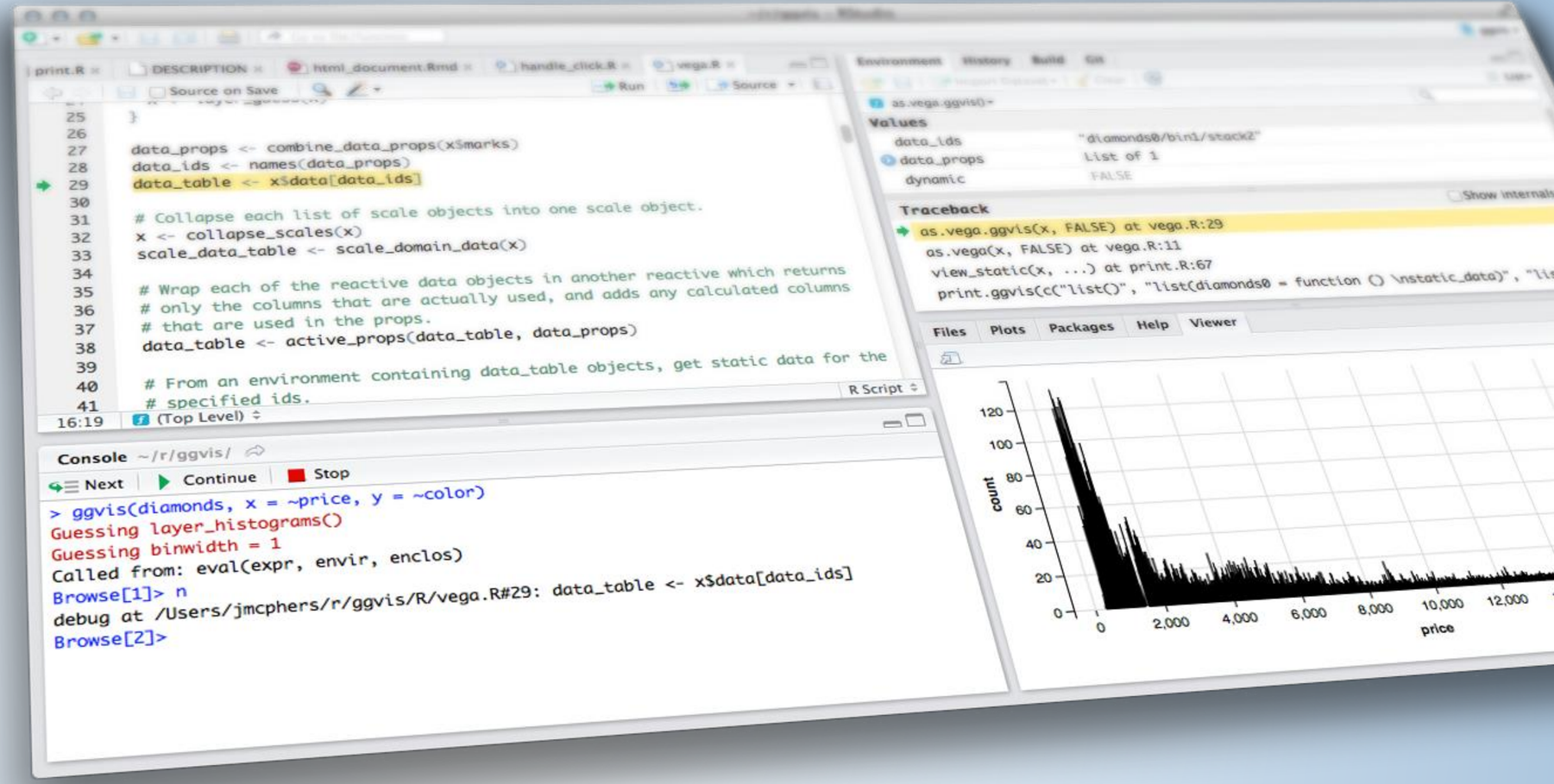
EXERCISE

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

5m 00s

Shiny Examples

- Port Authority Bus Tracker
- British Columbia CCISS Tool
- Commute Explorer
- Covid-19 Tracker

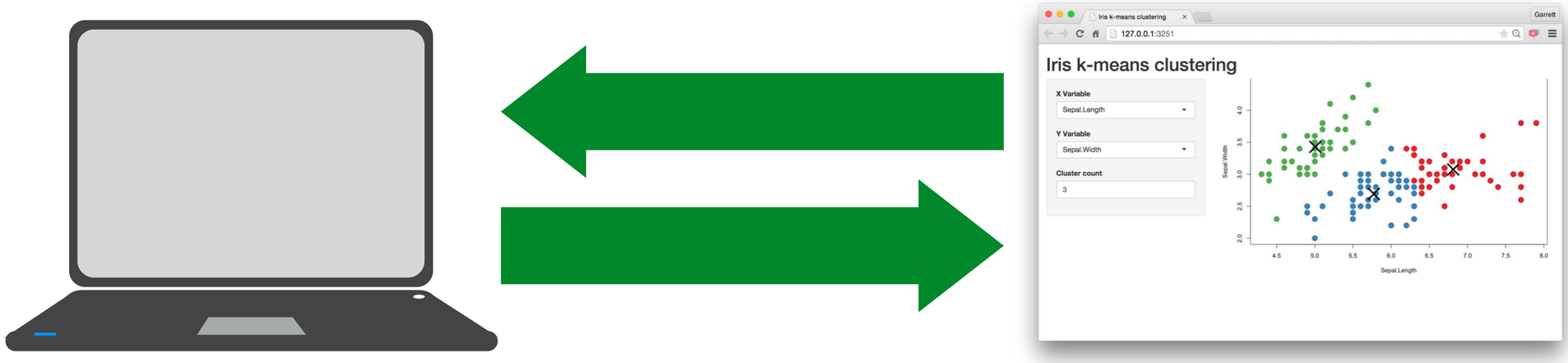


CLASS BREAK

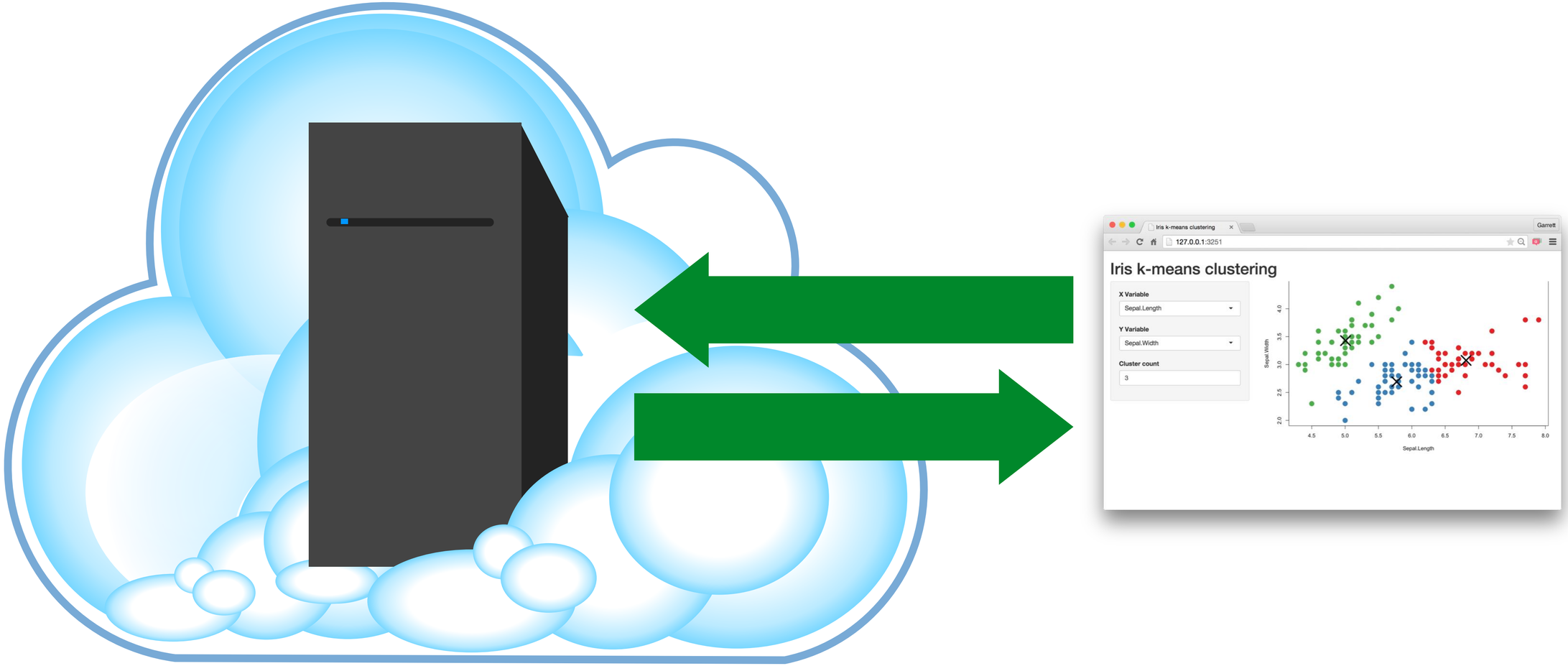
Shiny from posit

Shiny High level view

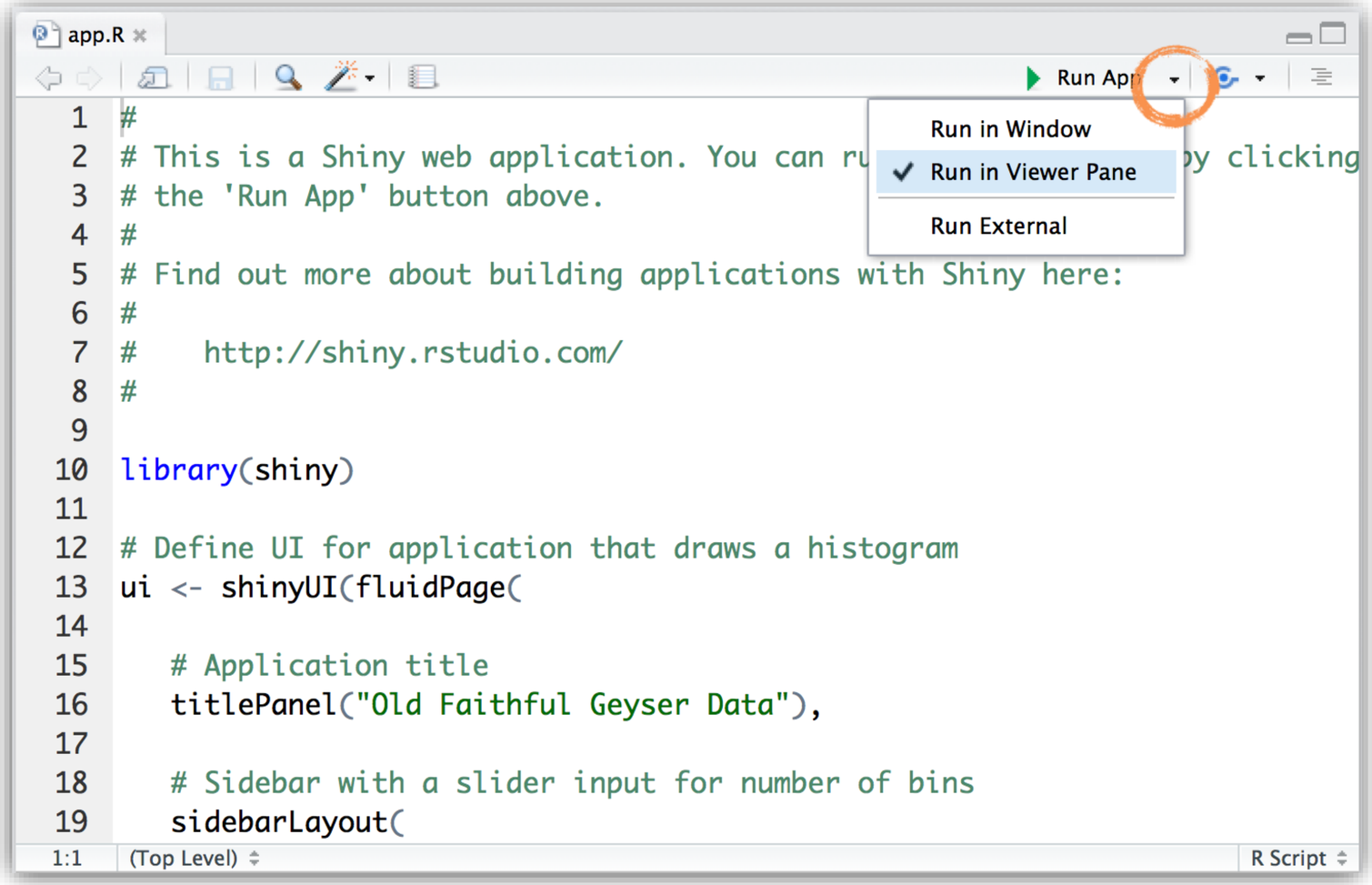
Every Shiny app is maintained by a computer running R



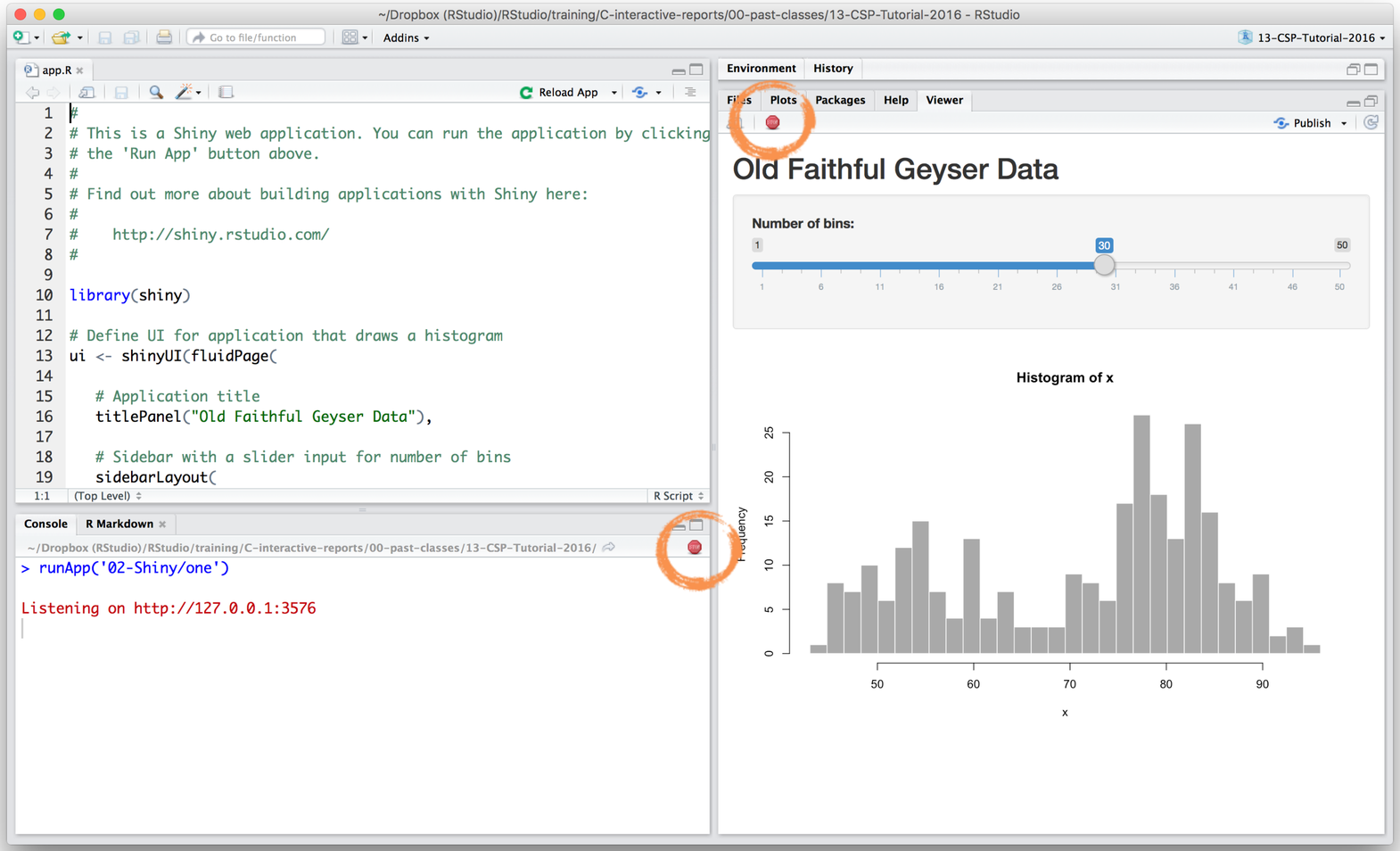
Every Shiny app is maintained by a computer running R



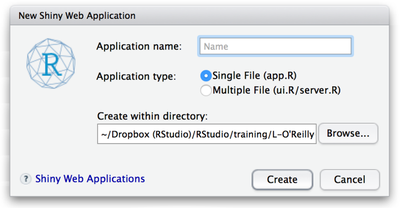
Change display



Close an app


~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/00-past-classes/13-CSP-Tutorial-2016 - RStudio
Go to file/function
Addins
13-CSP-Tutorial-2016
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(
1:1
(Top Level)
R Script
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
Files
Plots
Packages
Help
Viewer
Publish
Old Faithful Geyser Data
Number of bins:
1
30
50
Histogram of x
Frequency
x

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

~/Dropbox (RStudio)/shiny-training/apps/movies - Shiny

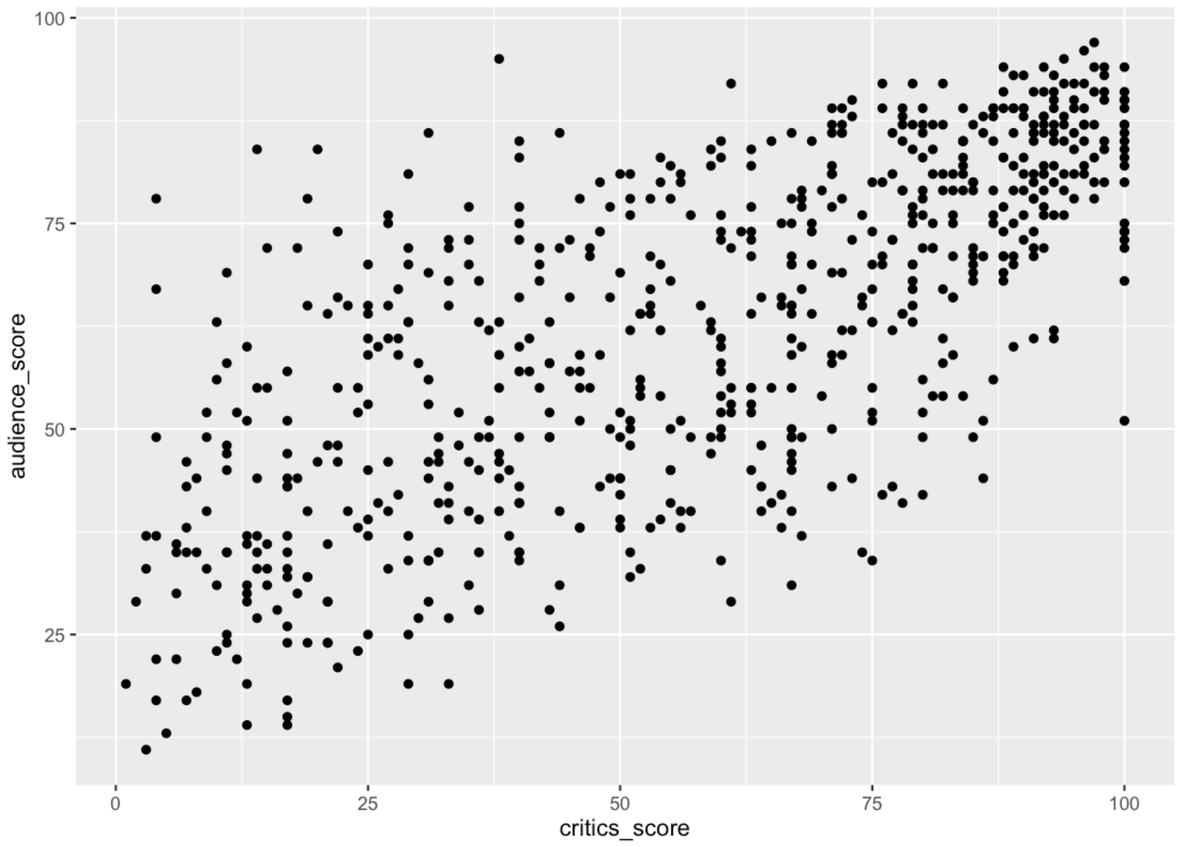
http://127.0.0.1:5877 Open in Browser Publish

Y-axis:

audience_score

X-axis:

critics_score



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

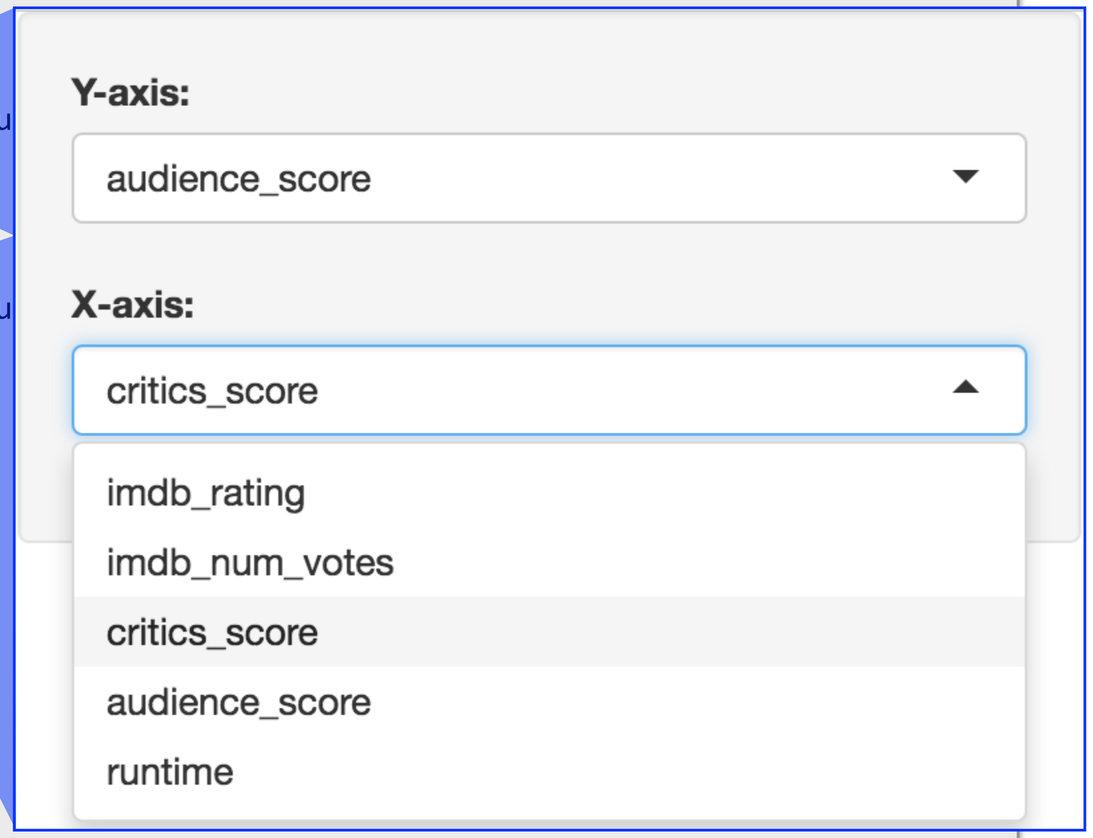
```r
# Define UI for application that plots features of movies
ui <- fluidPage(

  # 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

```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", "critics_score", "audience_score", "ru
                  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", "ru
                  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 scatterplot
server <- function(input, output) {

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

Contains instructions needed to build app

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

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

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

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

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

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

Running the app

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

DEMO

Putting it all together…

movies_01.R

EXERCISE

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

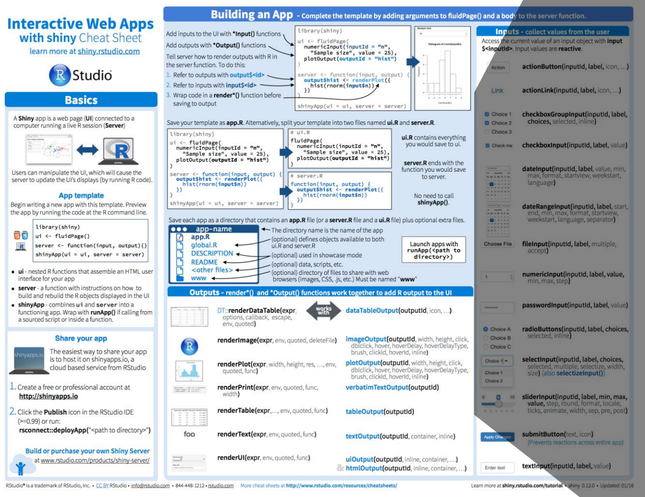
5m 00s

SOLUTION

Solution to the previous exercise

movies_02.R

INPUTS



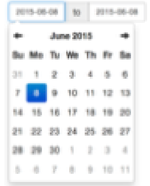
Action — **actionButton**(inputId, label, icon, …)

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 / 0 2 4 6 8 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)

Shiny from posit https://raw.githubusercontent.com/rstudio/cheatsheets/main/shiny.pdf

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

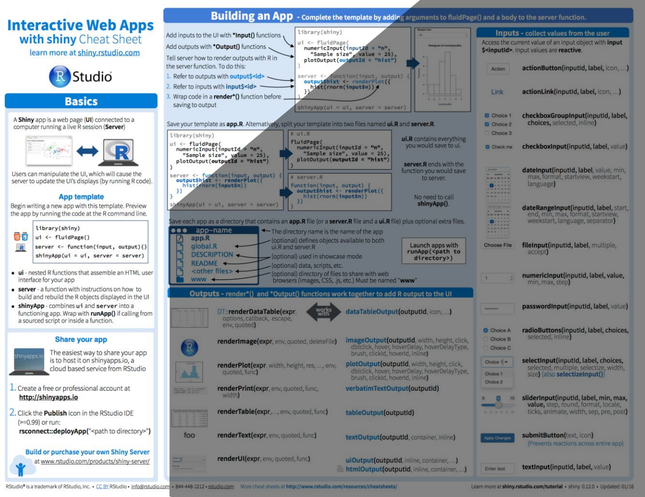
5m 00s

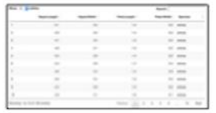
SOLUTION

Solution to the previous exercise

movies_03.R

OUTPUTS





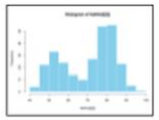
DT::renderDataTable(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

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

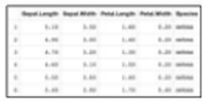


renderPlot(expr, width, height, res, …, env, quoted, func)

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
- **Optional:** If you finish early, move on to the next exercise

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

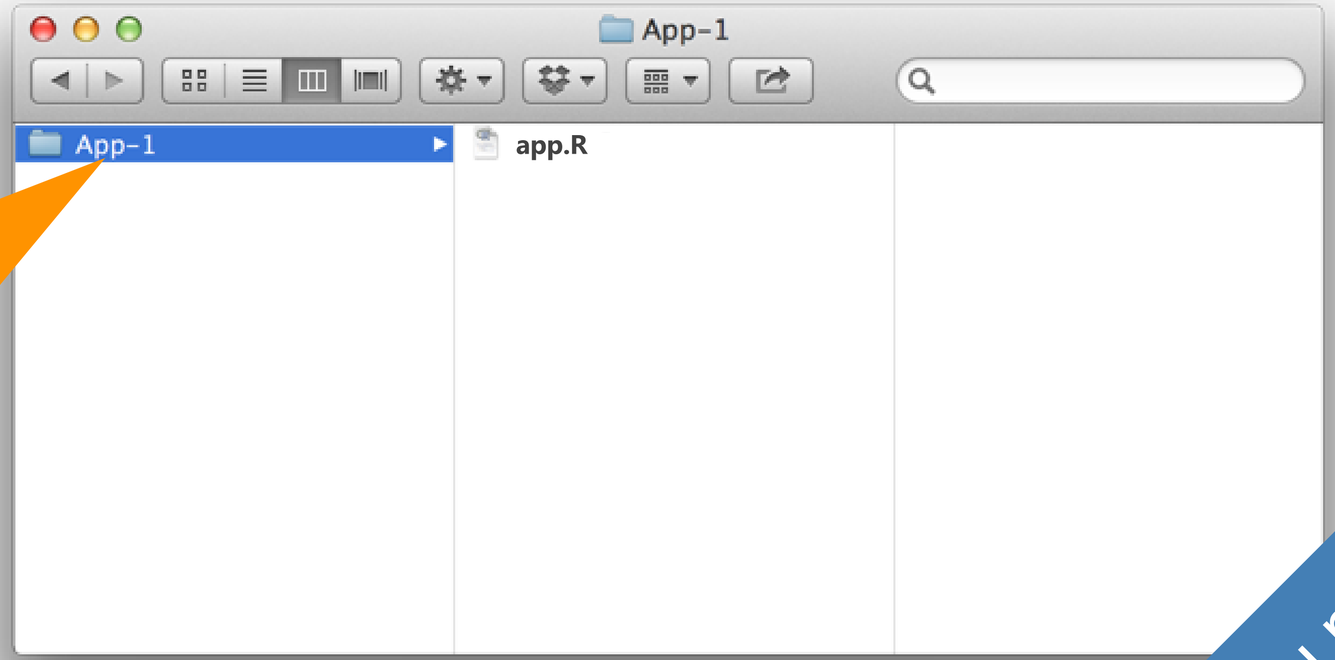
movies_05.R

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.



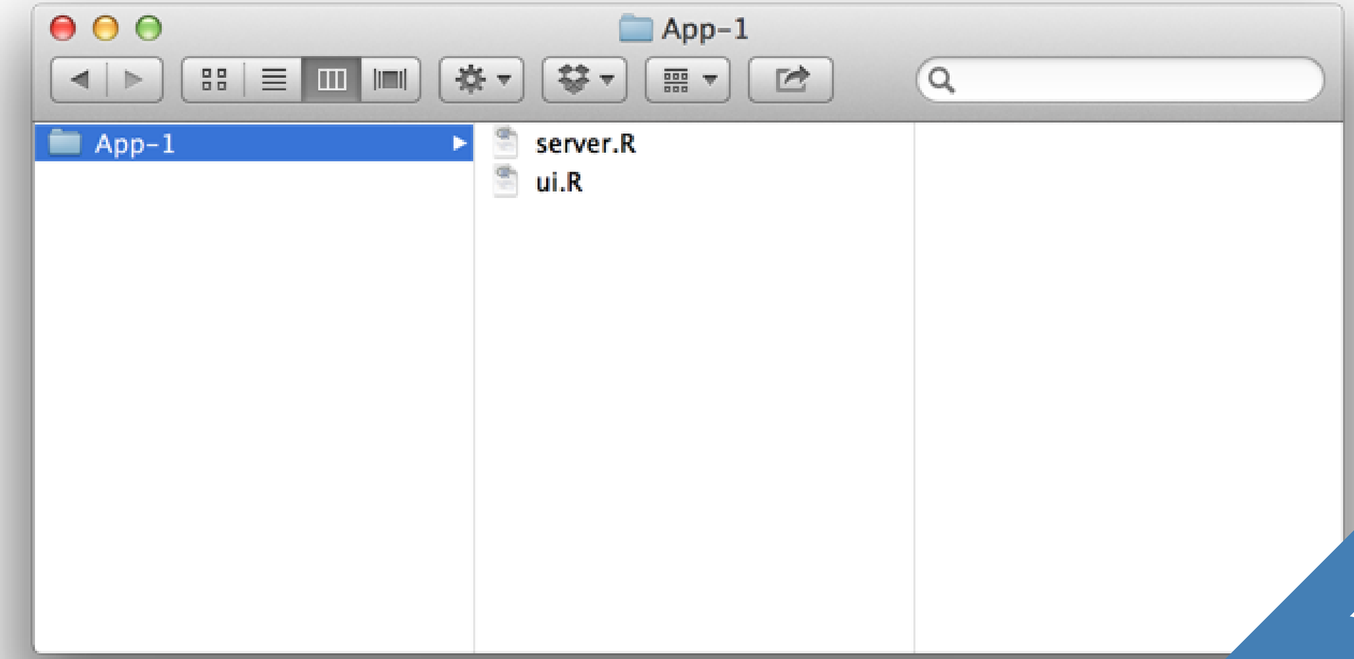
We will focus on the single file format throughout the course

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

SAVING YOUR MULTIPLE FILE APP

One directory with every file the app needs:

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



You must use these exact names

Sharing

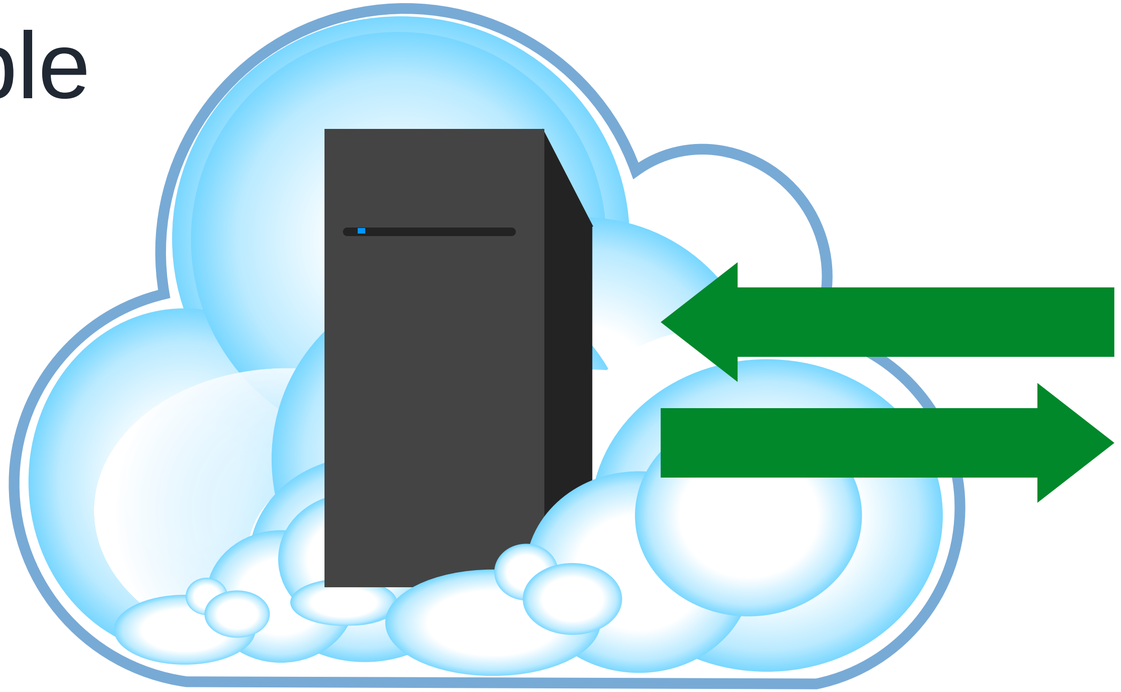
your app

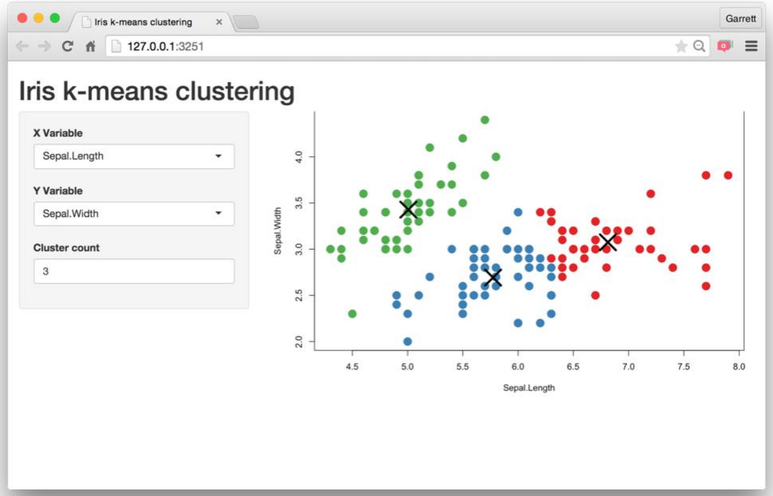
shinyapps.io

SHINYAPPS.IO

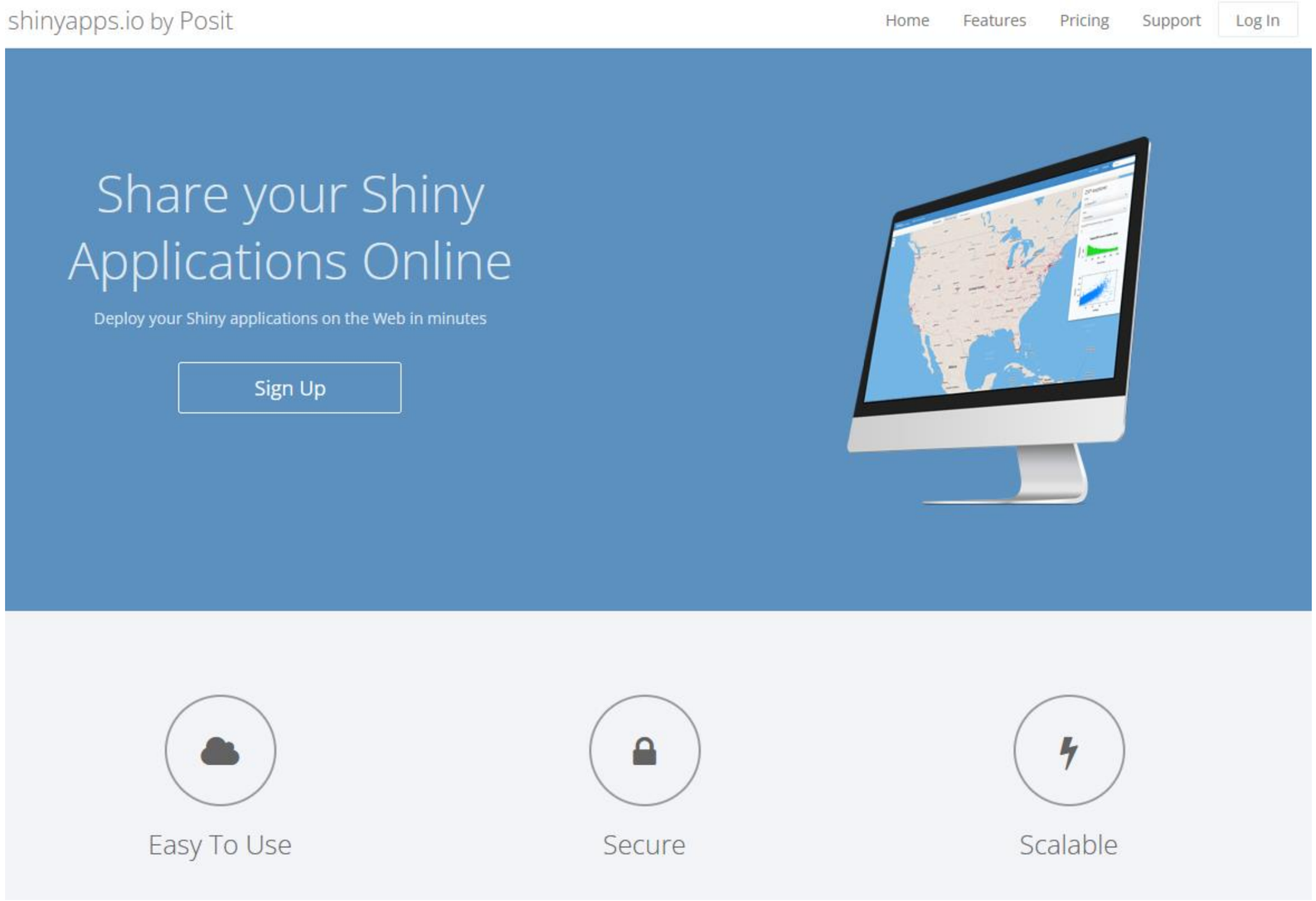
A server maintained by RStudio

- easy to use
- secure
- scalable

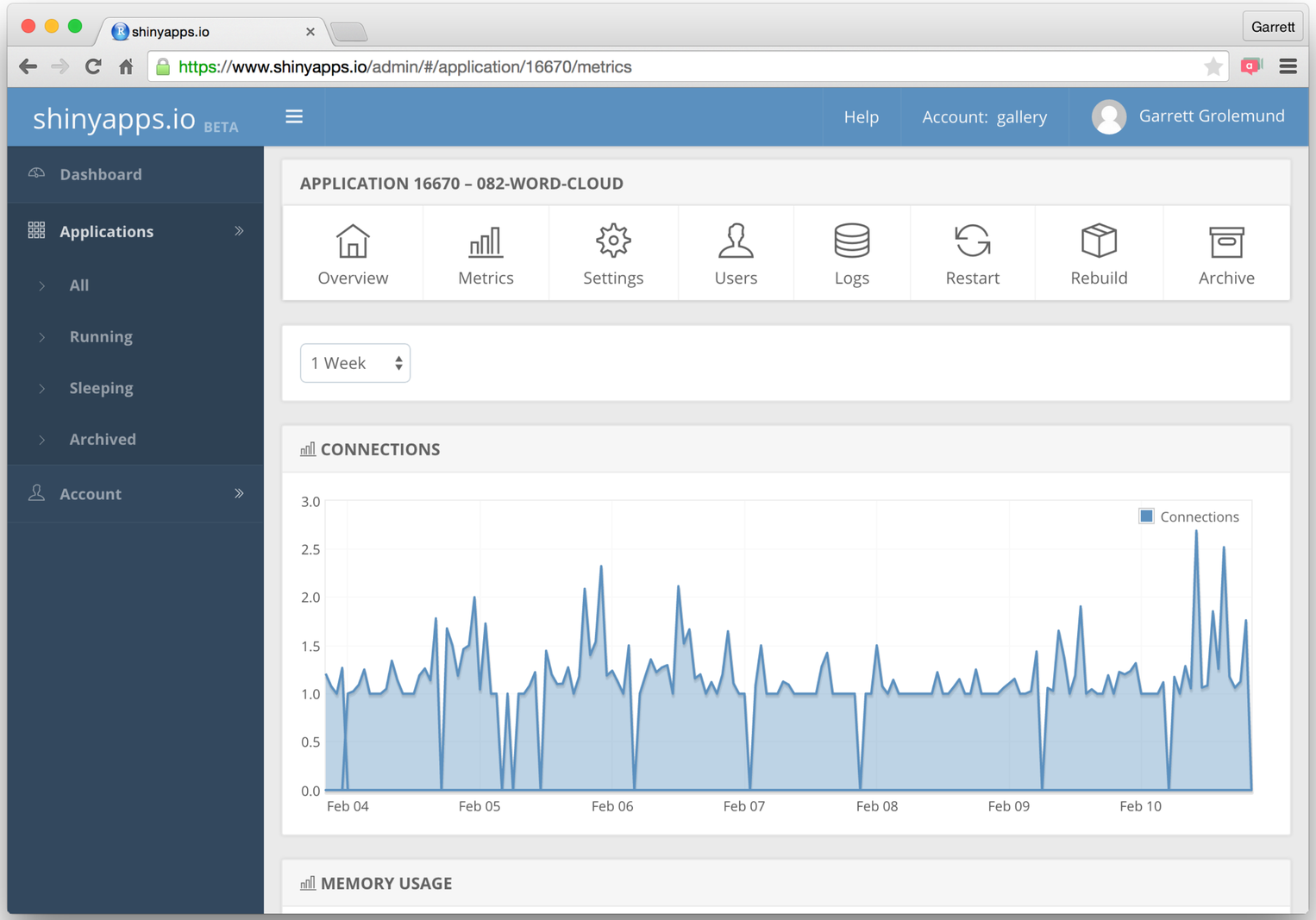




HASSLE-FREE CLOUD HOSTING FOR SHINY



WITH BUILT-IN METRICS

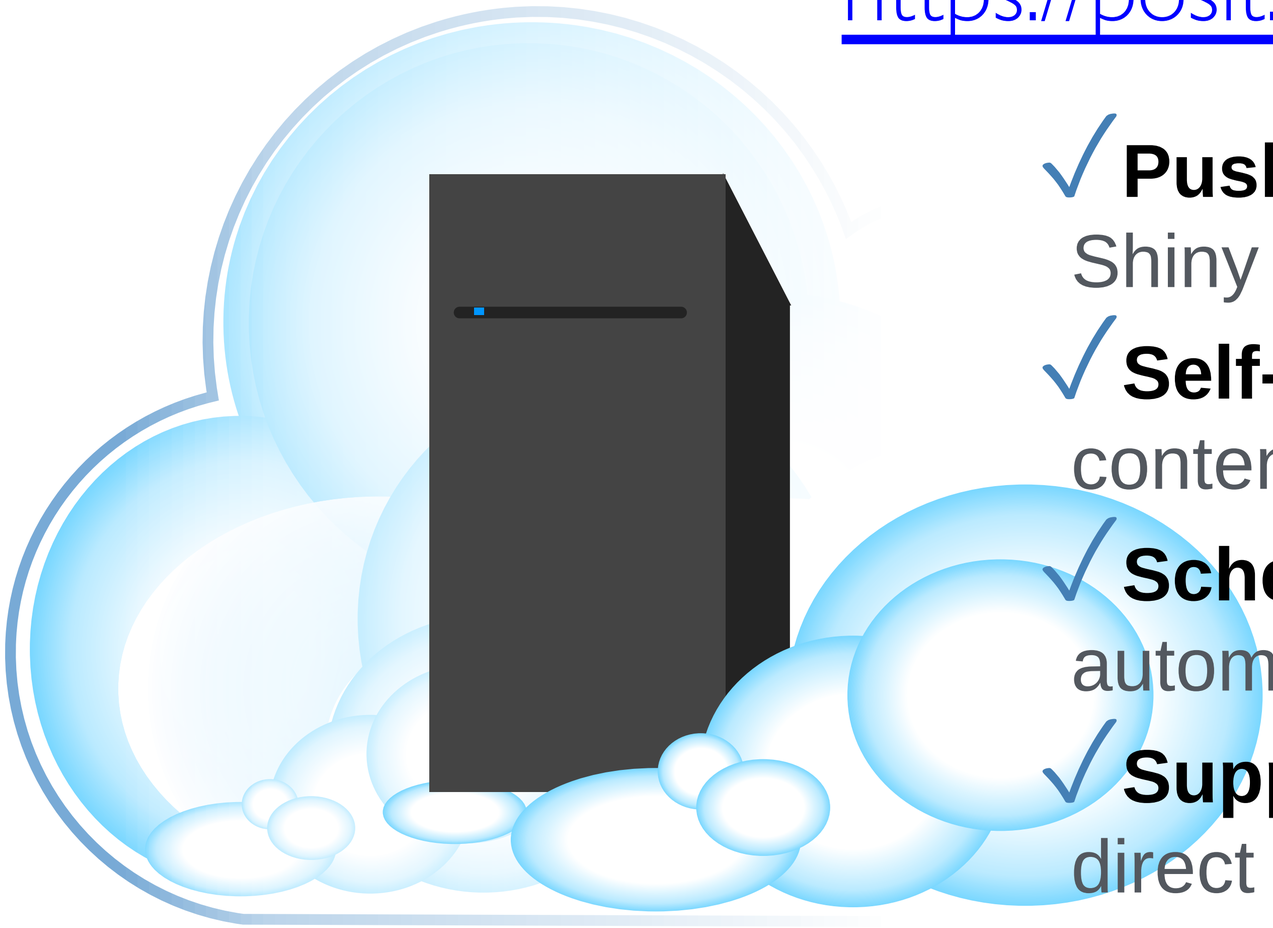


MEMBERSHIP PRICING

FREE	STARTER	BASIC	STANDARD	PROFESSIONAL
$ 0 /month	$ 13 /month (or $145/year)	$ 49 /month (or $550/year)	$ 119 /month (or $1,330/year)	$ 349 /month (or $3,860/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 Active Hours	**100** Active Hours	**500** Active Hours	**2,000** Active Hours	**10,000** Active Hours
Community Support	Premium Email Support	Performance Boost	Authentication	Authentication
		Premium Email Support	Performance Boost	Account Sharing
			Premium Email Support	Performance Boost
				Custom Domains
				Premium Email Support
Sign Up Now	Sign Up Now	Sign Up Now	Sign Up Now	Sign Up Now

POSIT CONNECT

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



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

evaluation free trial

Build your own server

SHINY SERVER

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

- ✓ **Deploy Shiny apps to the internet**
- ✓ **Run on-premises**

  move computation closer to the data
- ✓ **Host multiple apps on one server**
- ✓ **Deploy inside the firewall**
- ✓ **xcopy deployment**

Free & open source

EXERCISE

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

3m 00s

SHINYPROXY

https://shinyproxy.io/

✓ **Secure access**
LDAP, GoogleAuth, SSL, and more

✓ **Management**
View open apps and generate usage statistics

✓ **Docker**
uses docker to maintain dedicated app instances

✓ **Open Source**

✓ **ACL's**
Configurable to restrict app access

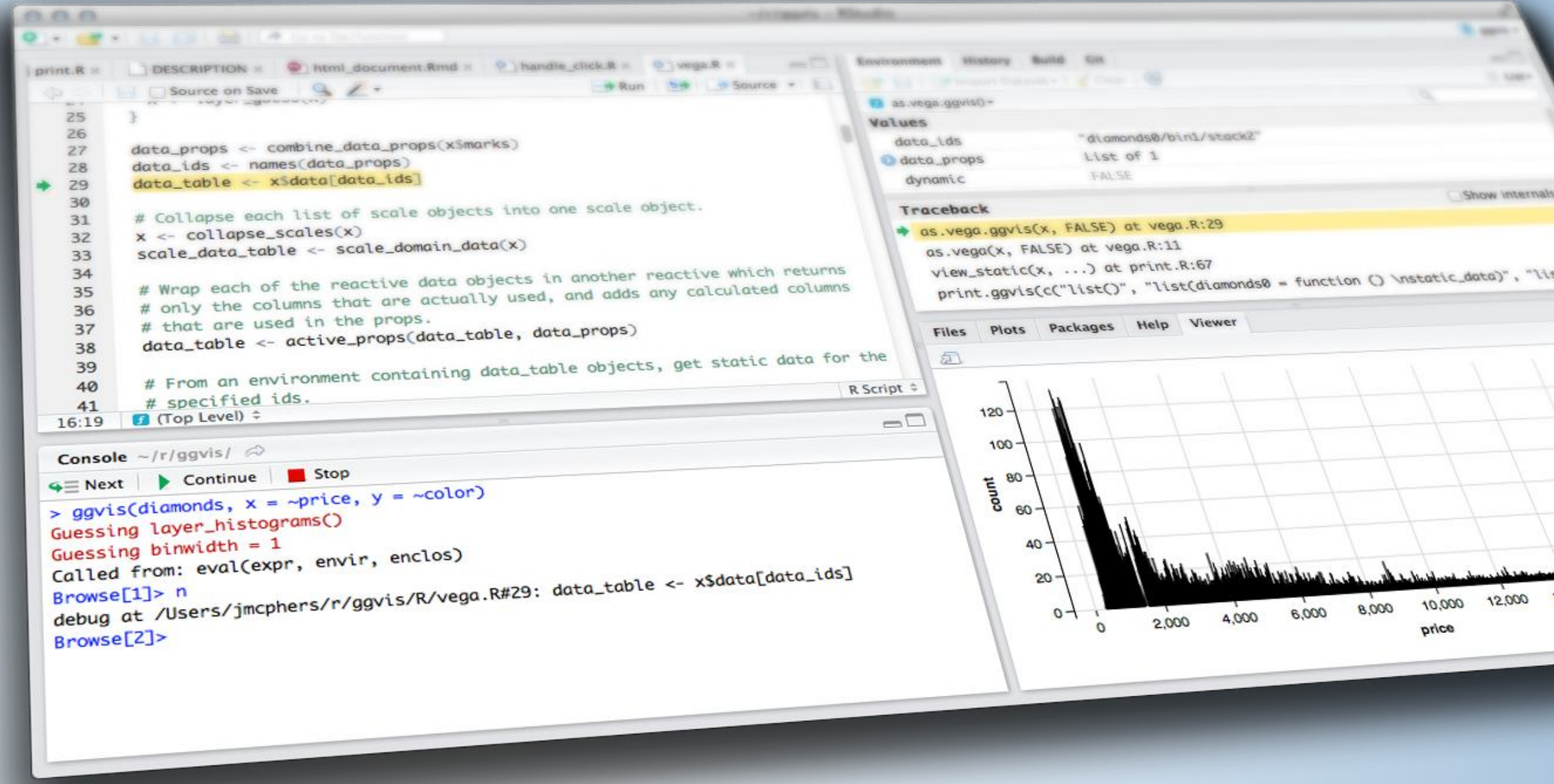
Free & open source!

More information

- Shinyproxy documentation: https://shinyproxy.io/documentation/deploying-apps/

- Shinyproxy github repo: https://github.com/openanalytics/shinyproxy

How do I deploy apps?



COURSE OVERVIEW

& INTRODUCTION TO SHINY

Shiny from posit