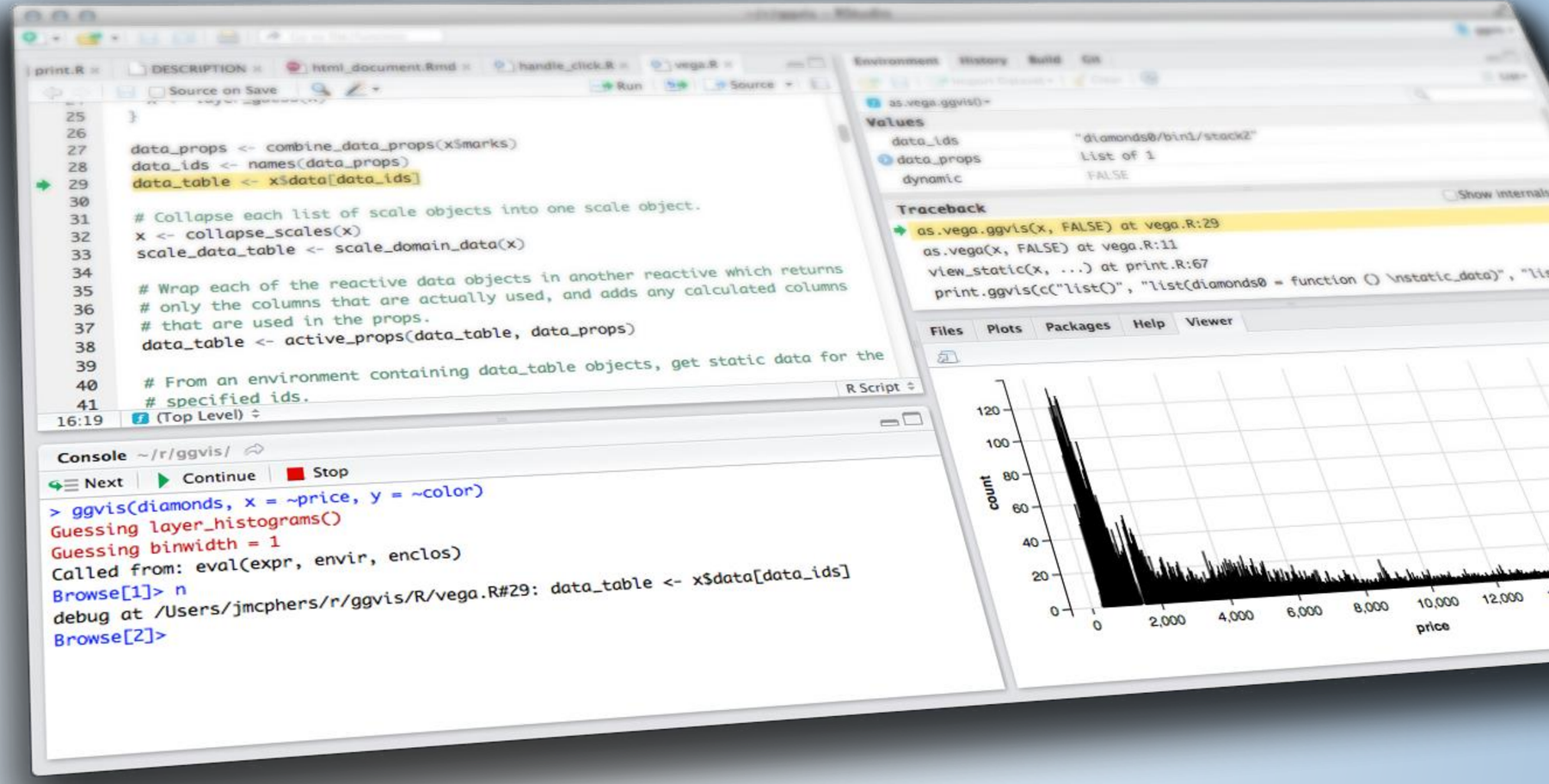


COURSE OVERVIEW

& INTRODUCTION TO SHINY

Shiny from posit®

OUTLINE

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

Course Overview

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* - Leaflet & LeafletProxy

Intro to Github

“Experience with version control is fast becoming a requirement for all data scientists”

–Rebecca Vickery

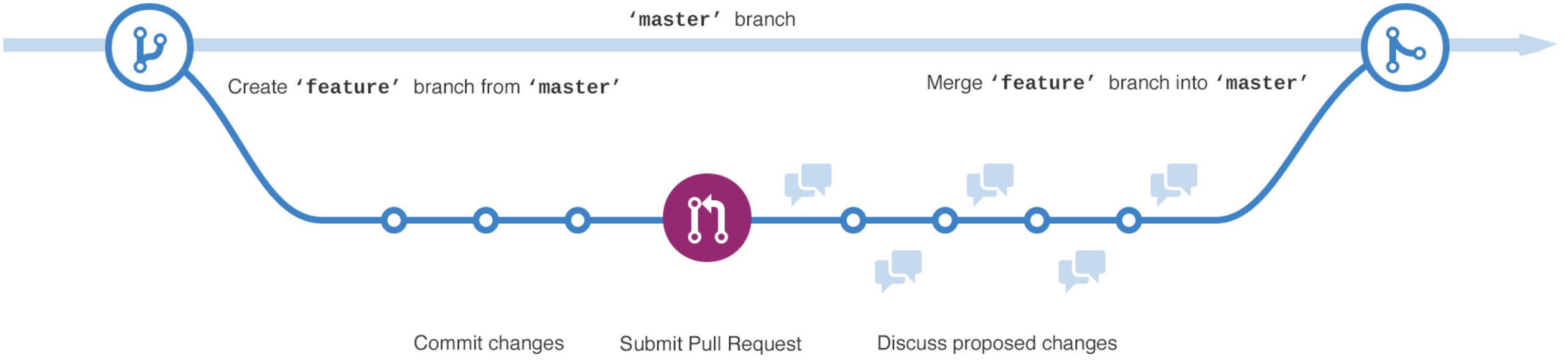
WHAT IS GITHUB?

- Git is a Version Control Software
- Github stores the files for your project in a remote location and checks the differences as you change your code
- This allows you to roll back to previous versions of your project if you need to go back
- Makes sharing and collaboration much easier using the Github website

OTHER VERSION CONTROL

There are other kinds of version control software, GitLab also uses git.

We will be using Github in this course, but if in a future life, you might use these others.


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

Github and Projects in RStudio

Projects in RStudio

“R Projects are great.”

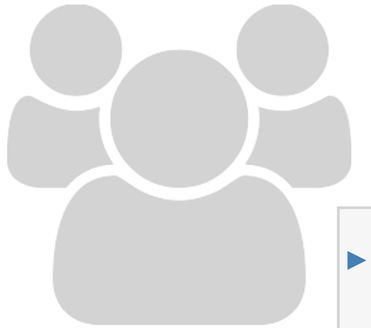
–Geoffrey Arnold

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?

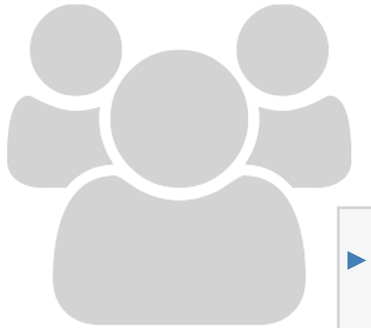
- R typically saves your environment information in a default location (typically your Documents folder)
- When you create a project it gets its own .RData file for the project in the Directory/folder you created
- This is also the default working directory for your project, so no need to put all of 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”
          - Look at the “Git” tab

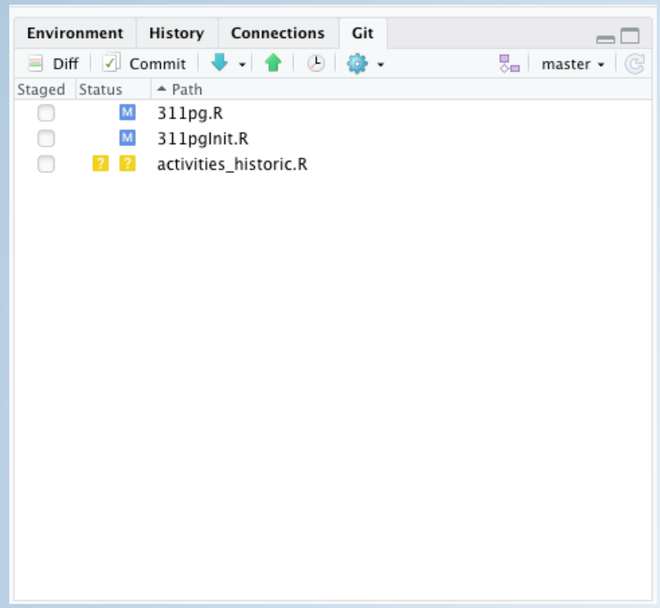
5m 00s

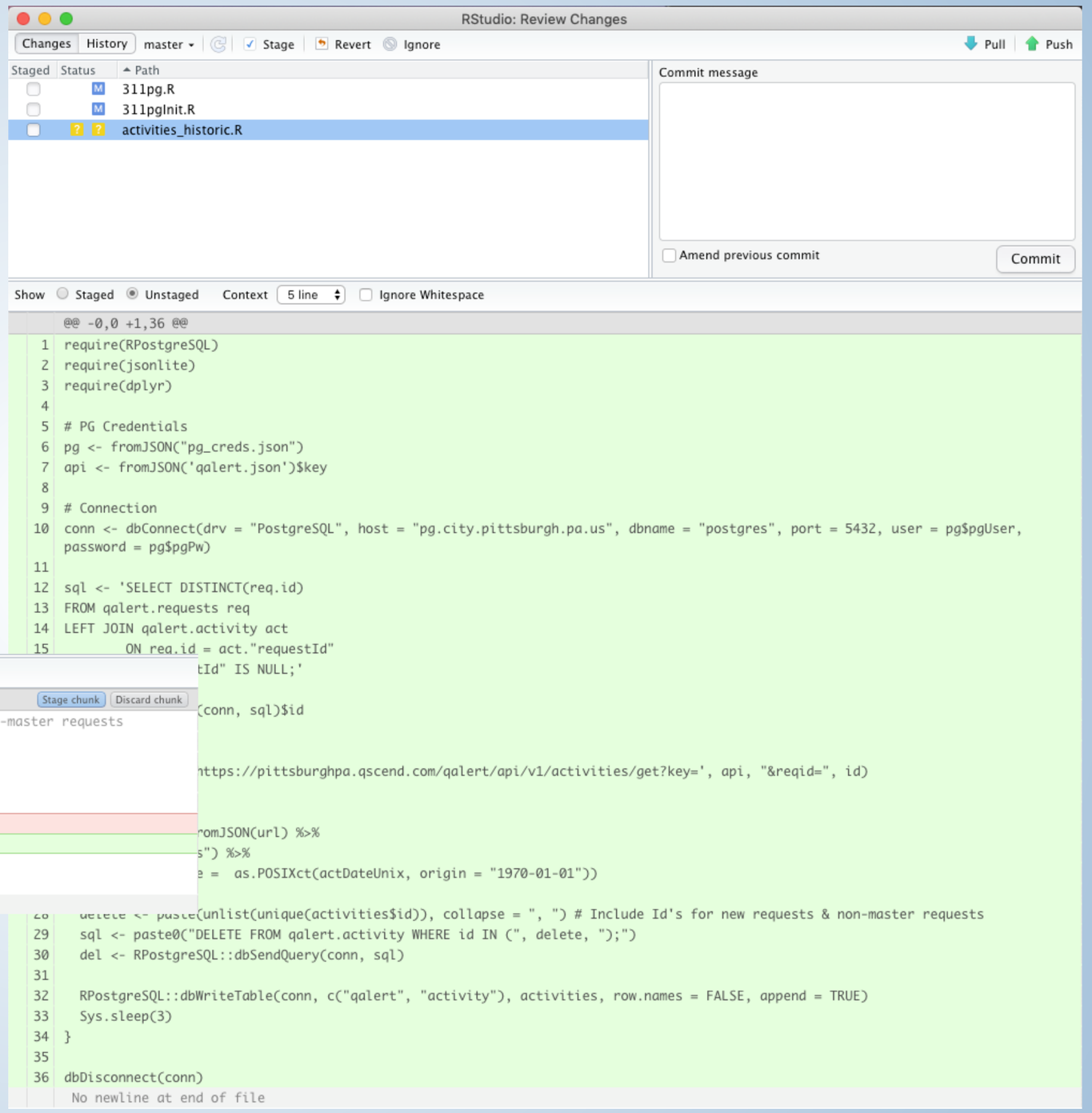


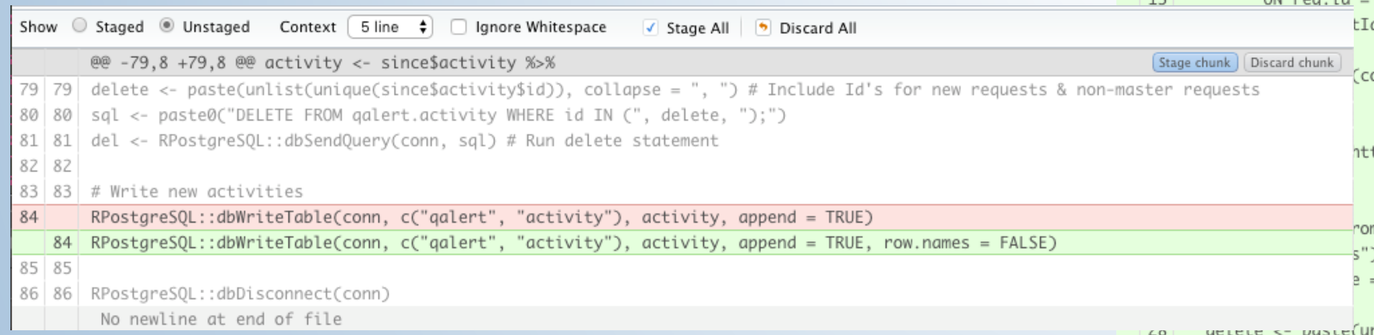
EXERCISE

- Click “File”
  - Click “New File”
    - Select “Shiny Web App”
      - Give the application a name and click “Create”
    - Click the “Git” tab
      - What’s changed?

5m 00s

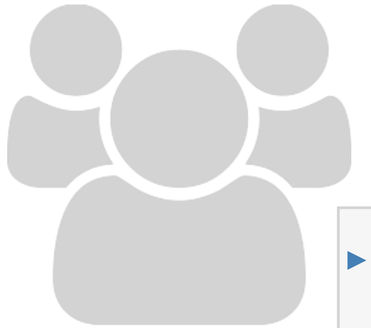

Environment
History
Connections
Git
Diff
Commit
master
Staged
Status
Path
311pg.R
311pgInit.R
activities_historic.R


RStudio: Review Changes
Changes
History
master
Stage
Revert
Ignore
Pull
Push
Staged
Status
Path
311pg.R
311pgInit.R
activities_historic.R
Commit message
Amend previous commit
Commit
Show
Staged
Unstaged
Context
5 line
Ignore Whitespace
@@ -0,0 +1,36 @@
1 require(RPostgreSQL)
2 require(jsonlite)
3 require(dplyr)
4
5 # PG Credentials
6 pg <- fromJSON("pg_creds.json")
7 api <- fromJSON('qalert.json')$key
8
9 # Connection
10 conn <- dbConnect(drv = "PostgreSQL", host = "pg.city.pittsburgh.pa.us", dbname = "postgres", port = 5432, user = pg$pgUser, password = pg$pgPw)
11
12 sql <- 'SELECT DISTINCT(req.id)
13 FROM qalert.requests req
14 LEFT JOIN qalert.activity act
15 ON req.id = act."requestId"
tId" IS NULL;'
(conn, sql)$id
https://pittsburghpa.qscend.com/qalert/api/v1/activities/get?key=', api, "&reqid=", id)
romJSON(url) %>%
s") %>%
e = as.POSIXct(actDateUnix, origin = "1970-01-01"))
delete <- paste(unlist(unique(activities$id)), collapse = ", ") # Include Id's for new requests & non-master requests
29 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ");")
30 del <- RPostgreSQL::dbSendQuery(conn, sql)
31
32 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activities, row.names = FALSE, append = TRUE)
33 Sys.sleep(3)
34 }
35
36 dbDisconnect(conn)
No newline at end of file


Show
Staged
Unstaged
Context
5 line
Ignore Whitespace
Stage All
Discard All
@@ -79,8 +79,8 @@ activity <- since$activity %>%
Stage chunk
Discard chunk
79 79 delete <- paste(unlist(unique(since$activity$id)), collapse = ", ") # Include Id's for new requests & non-master requests
80 80 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ");")
81 81 del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
82 82
83 83 # Write new activities
84 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
84 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
85 85
86 86 RPostgreSQL::dbDisconnect(conn)
No newline at end of file

WHATS THE COMMAND LINE

- Command or terminal git commands are how git was first used.
- You don’t need to know how to do these things as either the “Git” tab in RStudio or the Github desktop program provides a GUI (gooey user interface) for you.
- However, knowing the hard way to do something never hurt anybody.



OPTIONAL EXERCISE

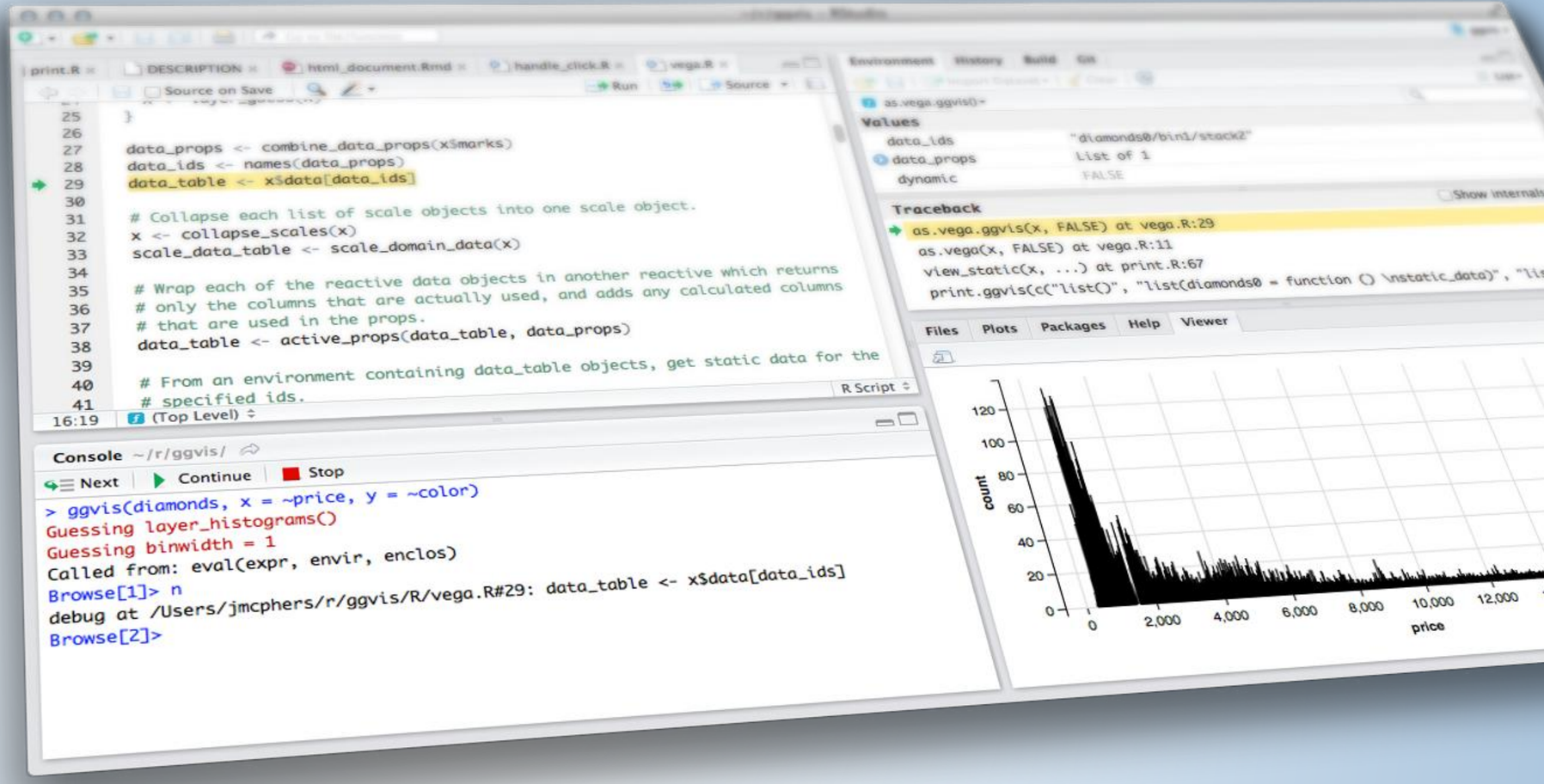
- https://learngitbranching.js.org/
- Complete Introduction Sequences
  - Git Commits
  - Branching Git
  - Merging in Git

5m 00s

OPTIONAL 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



CLASS BREAK

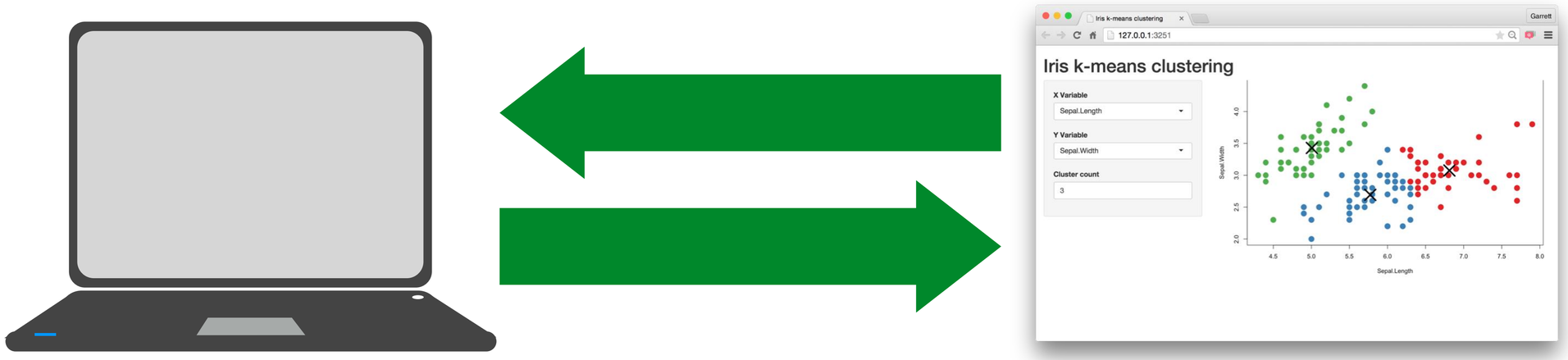
Shiny from posit®

Shiny Examples

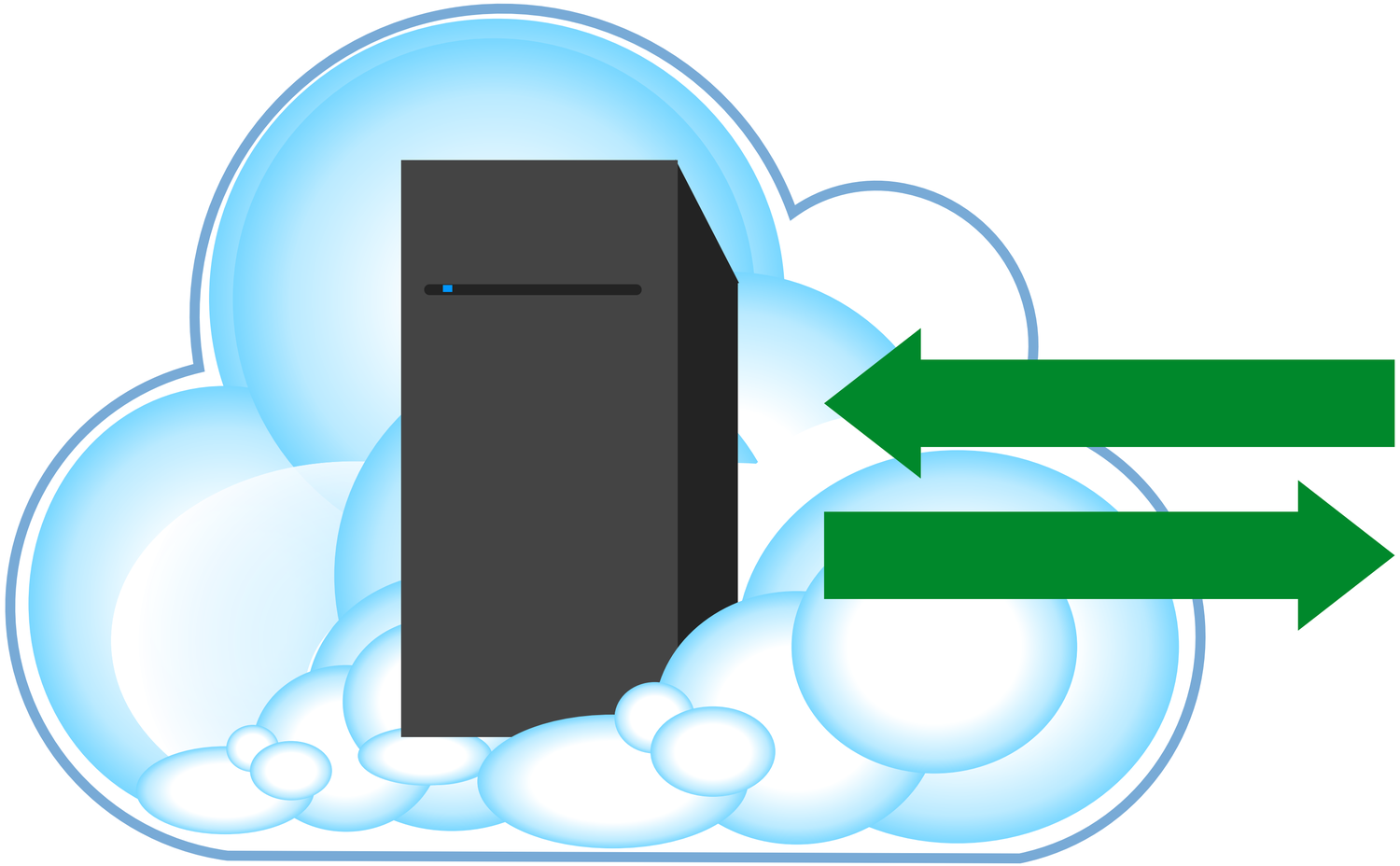
- Port Authority Bus Tracker
- British Columbia CCISS Tool
- Commute Explorer
- Covid-19 Tracker

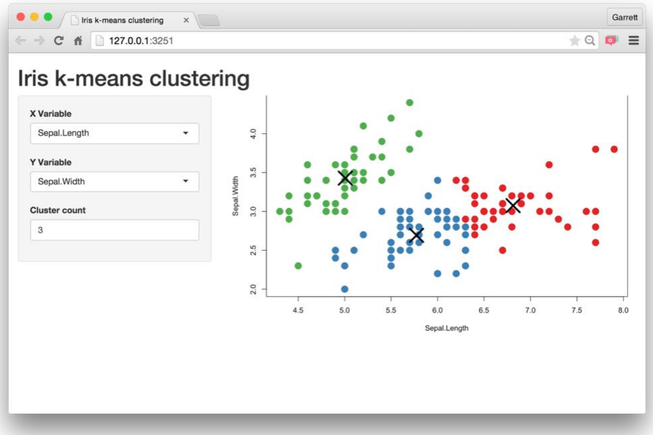
Shiny High level view

Every Shiny app is maintained by a computer running R

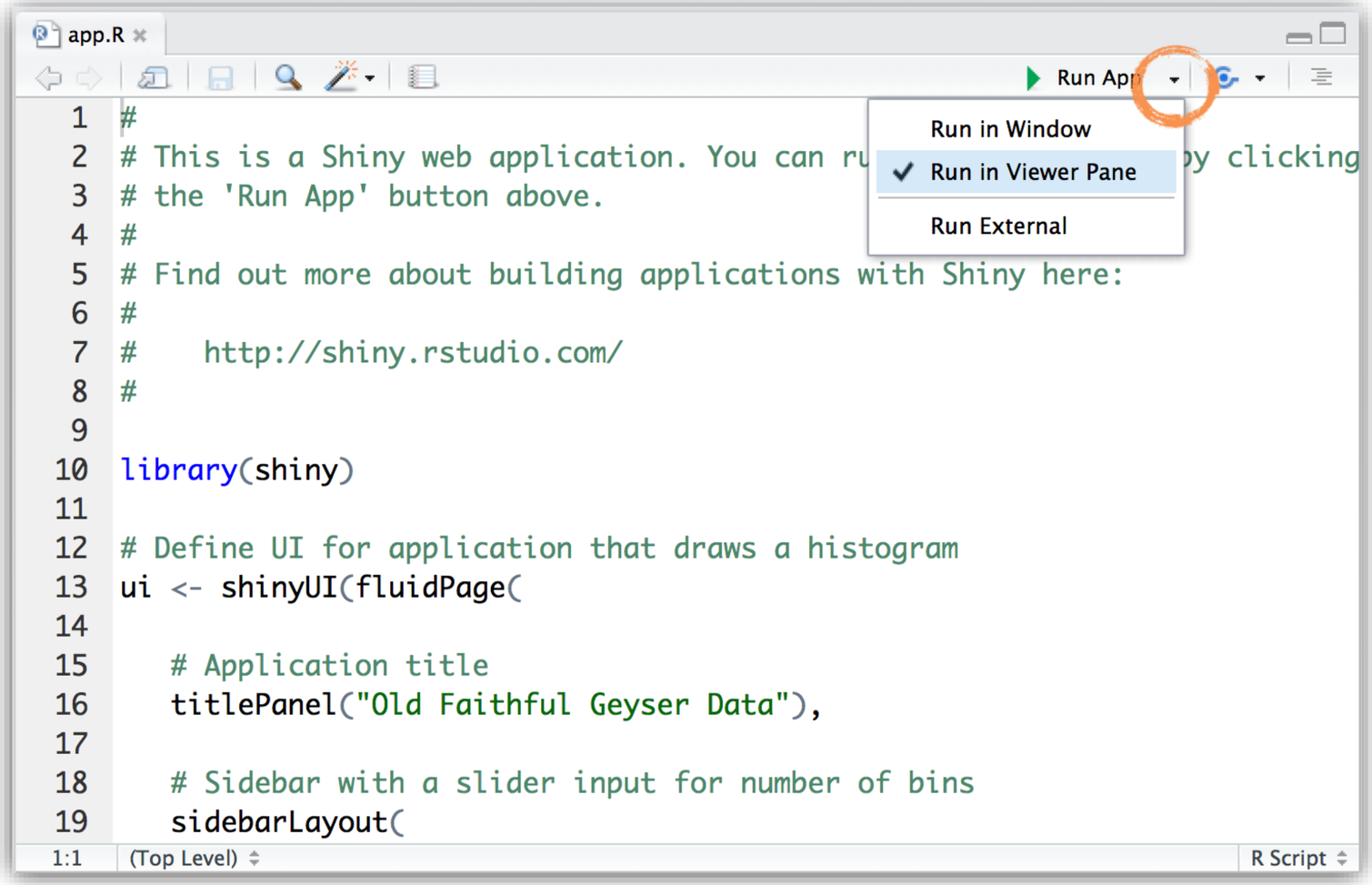


Every Shiny app is maintained by a computer running R

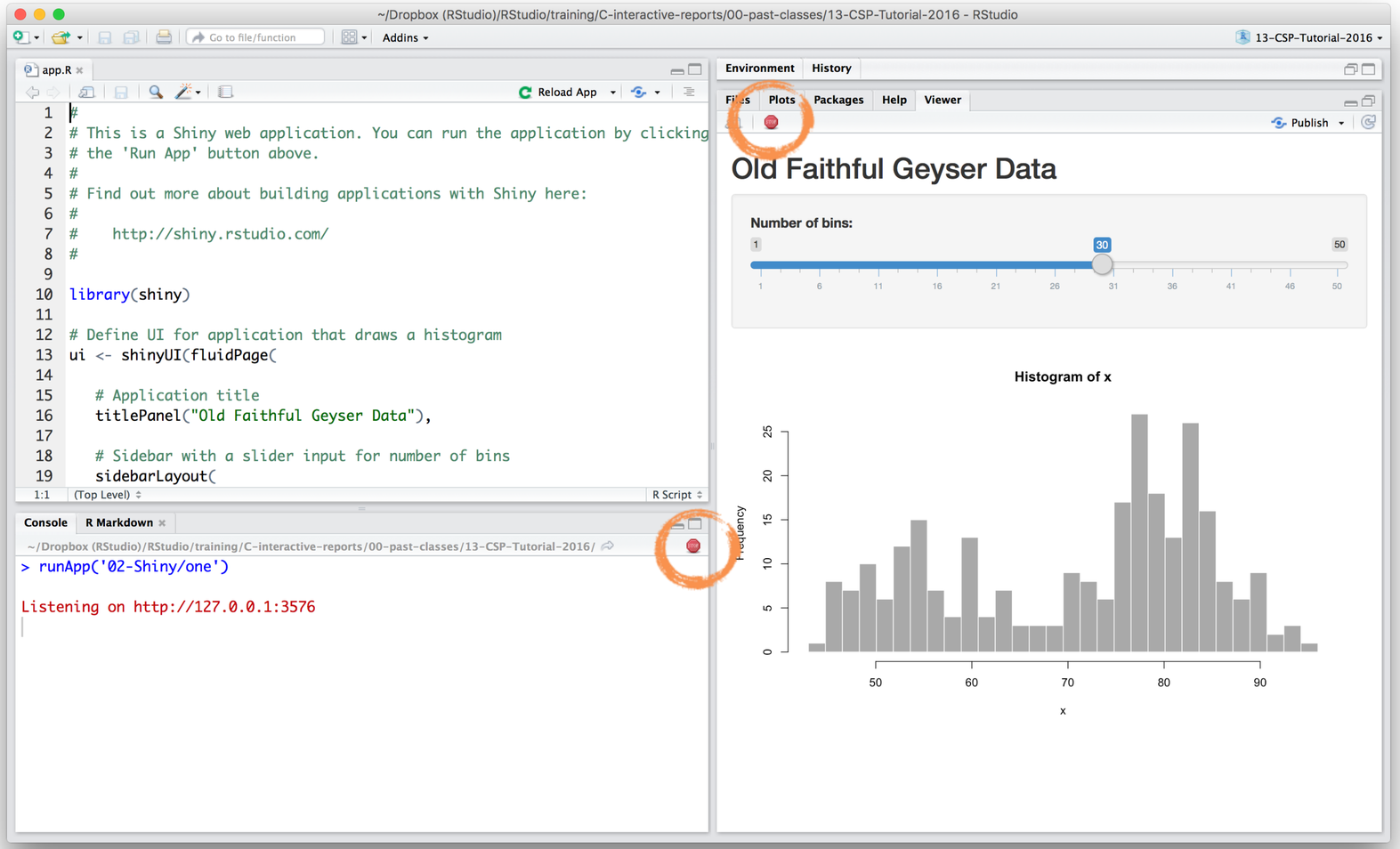




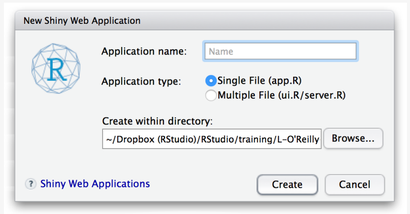
Change display



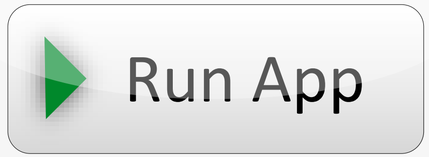
Close an app


~/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
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(
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
1 6 11 16 21 26 31 36 41 46 50
Histogram of x
Frequency
0 5 10 15 20 25
50 60 70 80 90
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)
```

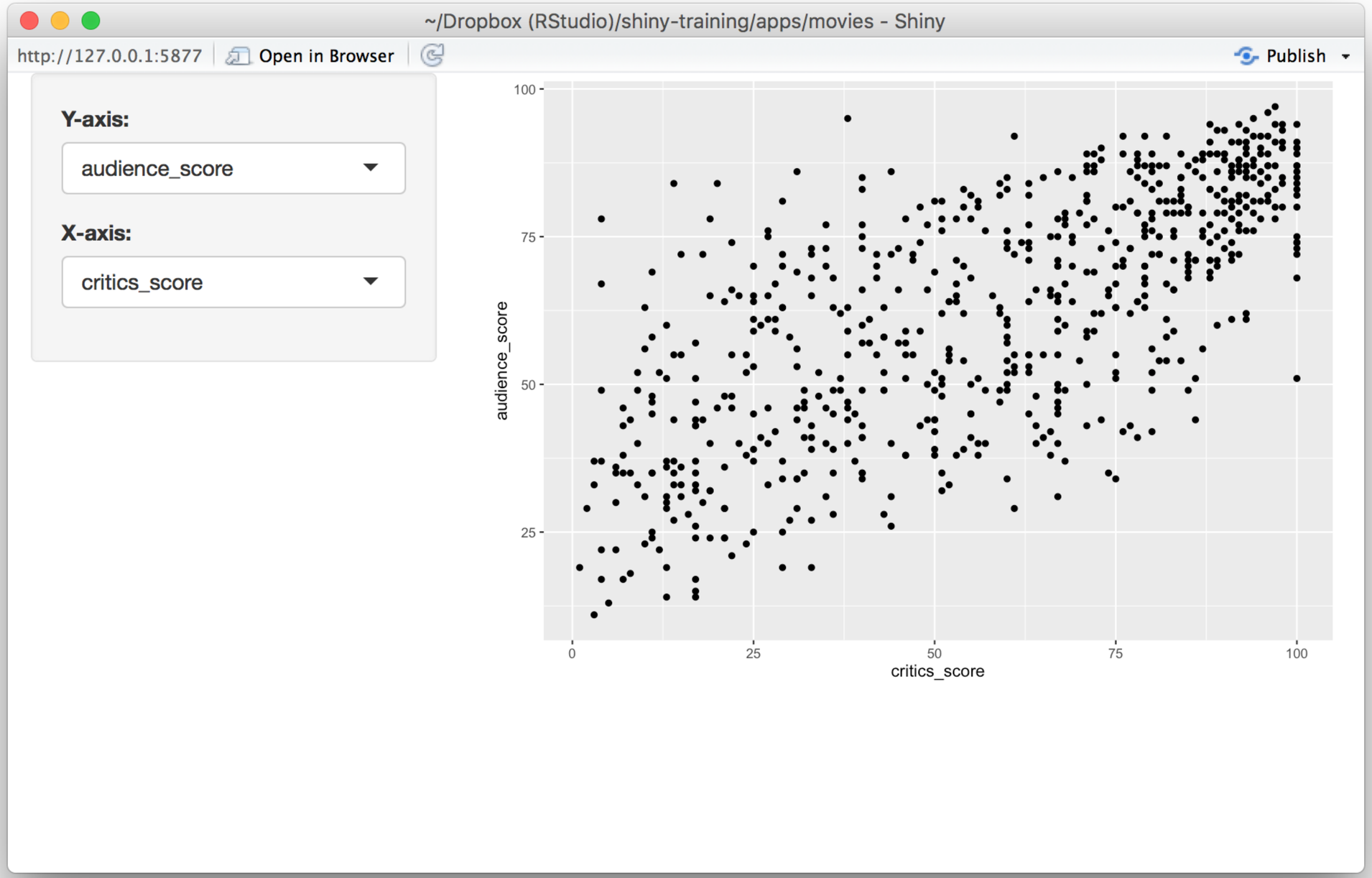
User interface
controls the layout and appearance of app

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
100
75
50
25
audience_score
0
25
50
75
100
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")
    )
  )
)
```

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")
    )
  )
)
```

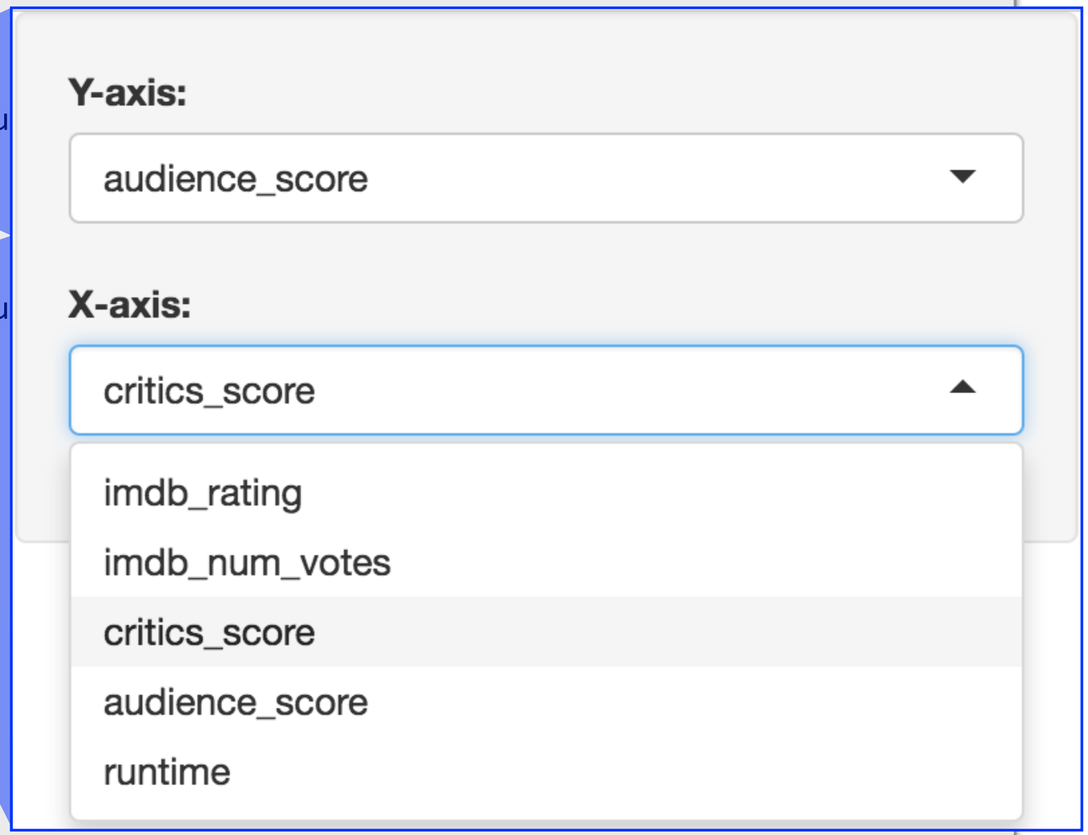
```r
# Define UI for application that plots features of movies
ui <- fluidPage(

  # 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")
    )
  )
```



```
# 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 **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

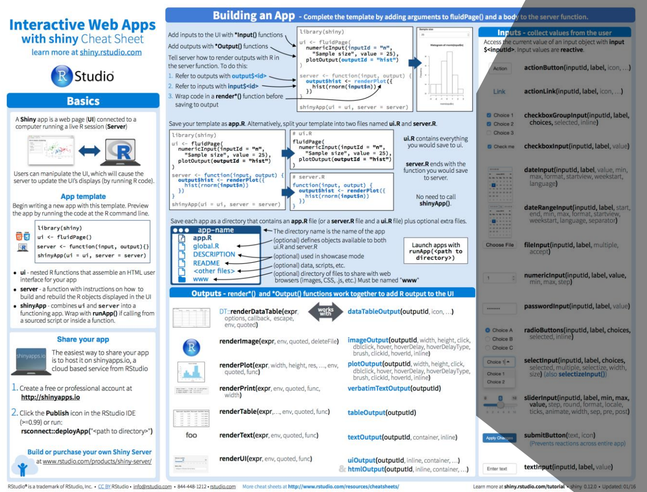
5m 00s

SOLUTION

Solution to the previous exercise

movies_02.R

INPUTS



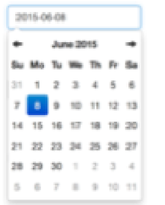
Action — **actionButton**(inputId, label, icon, …)

Link — **actionLink**(inputId, label, icon, …)

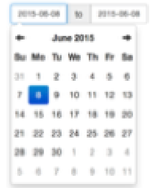
- [x] Choice 1
- [x] Choice 2
- [ ] Choice 3

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()**)

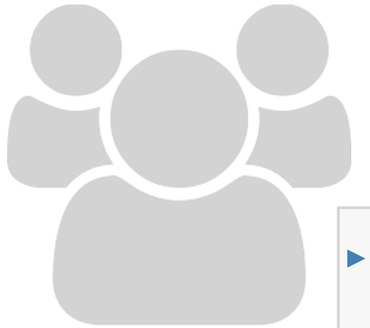
0 — 5 — 10
0 2 4 6 8 10

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

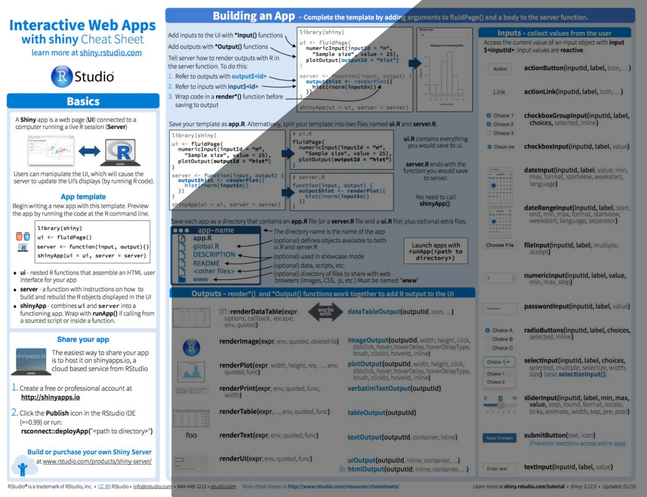
5m 00s

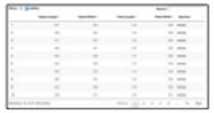
SOLUTION

Solution to the previous exercise

movies_03.R

OUTPUTS





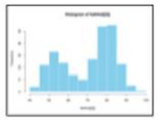
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

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



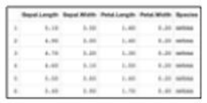
renderPlot(expr, width, height, res, …, env, quoted, func)

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

```
'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)



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

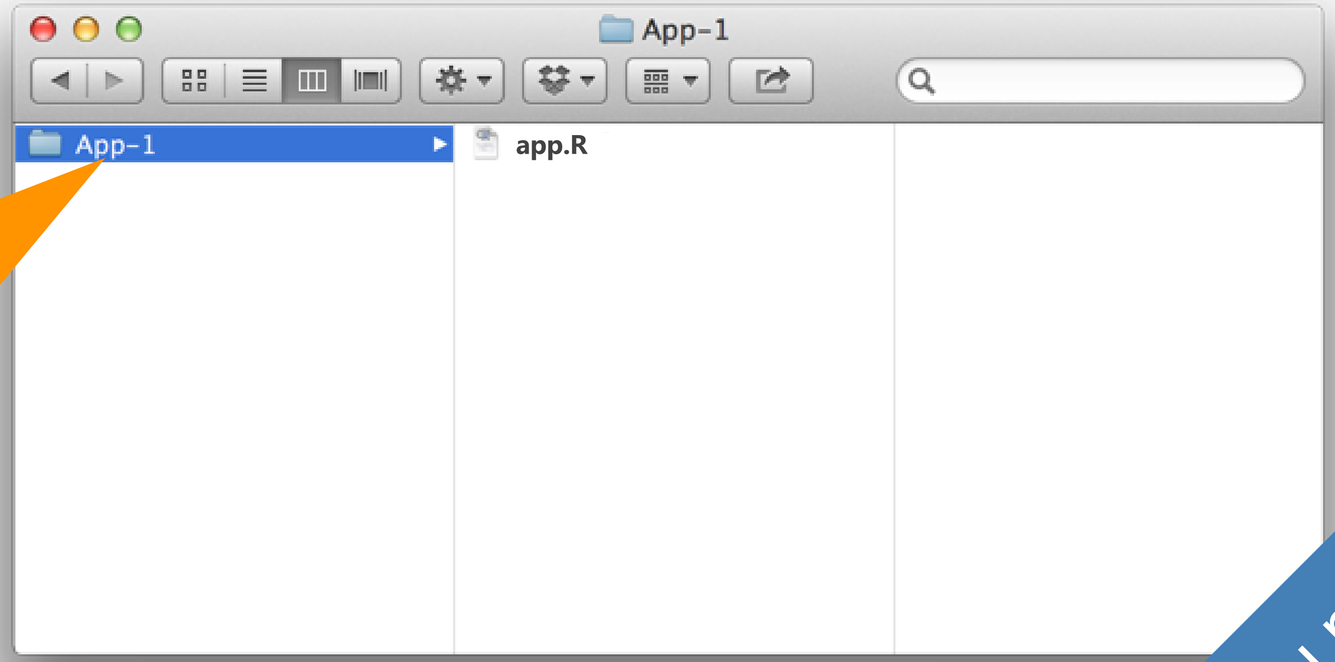
movies_05.R

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.



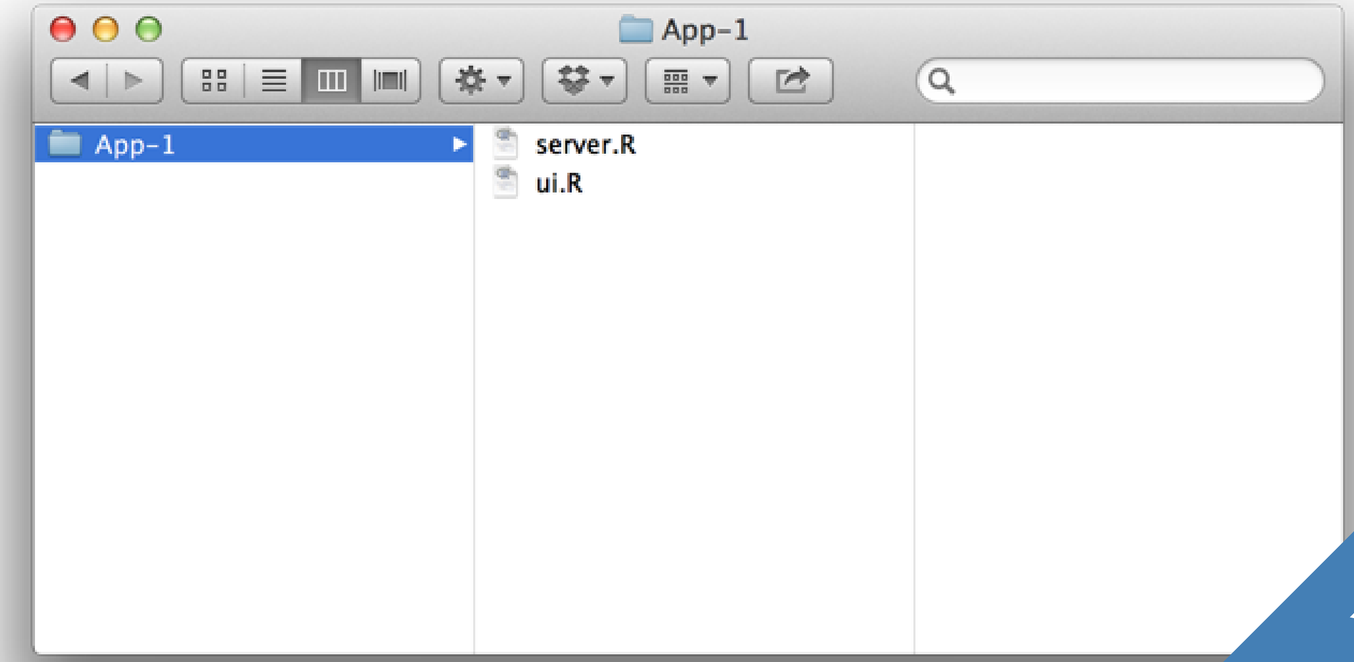
We will focus on the single file format throughout the course

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

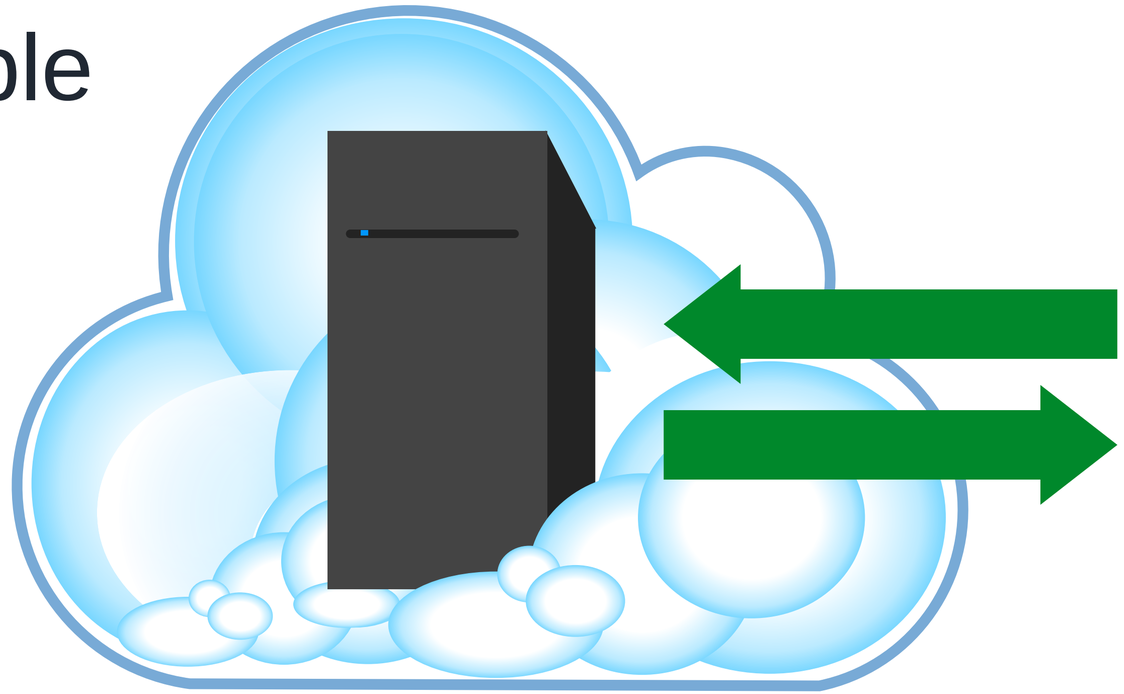
Sharing your app

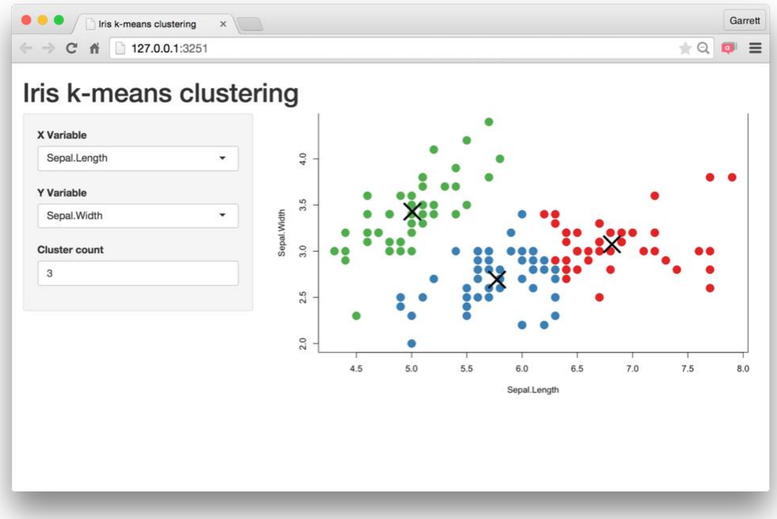
shinyapps.io

SHINYAPPS.IO

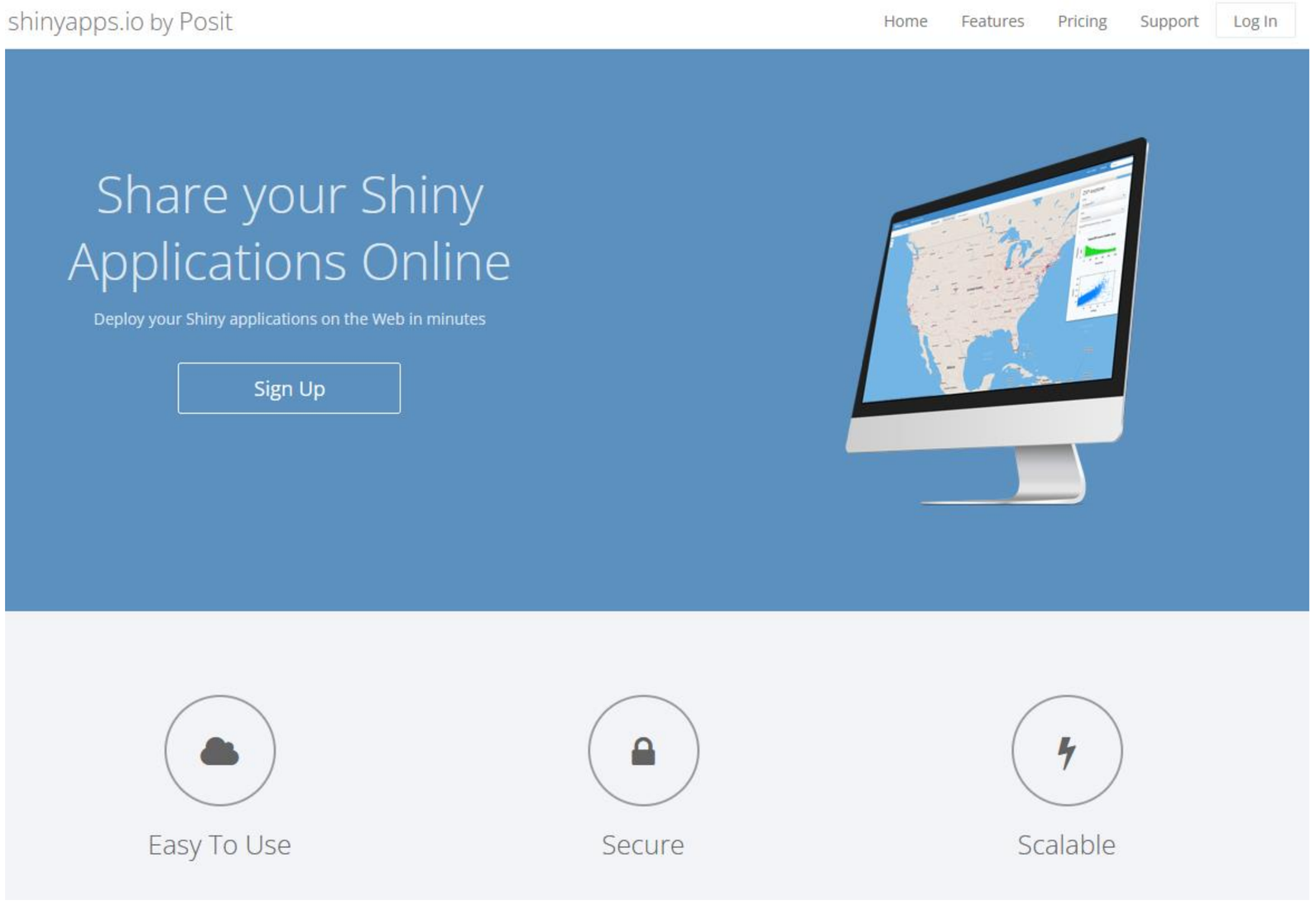
A server maintained by RStudio

- easy to use
- secure
- scalable

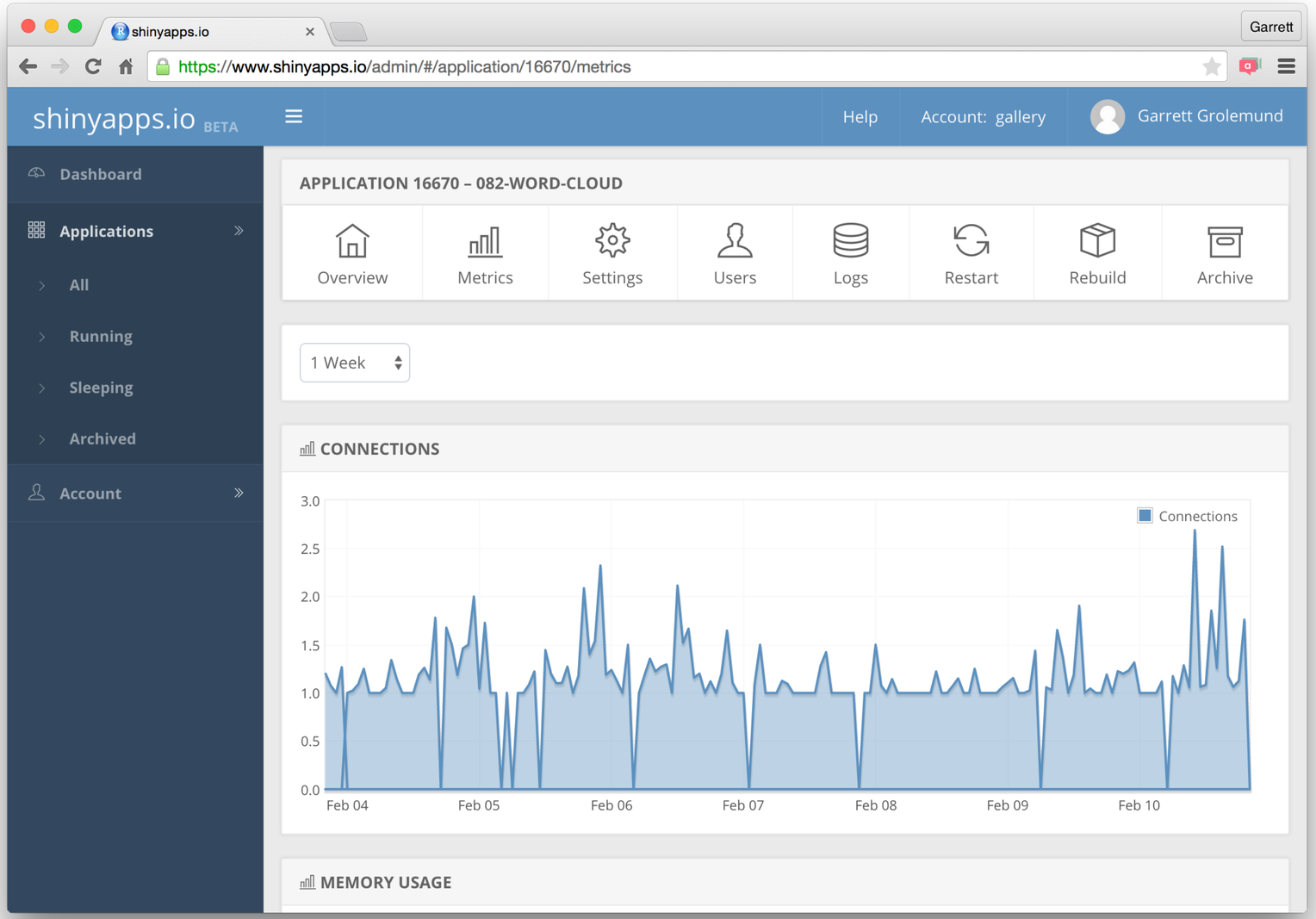




HASSLE-FREE CLOUD HOSTING FOR SHINY



WITH BUILT-IN METRICS



MEMBERSHIP PRICING

FREE

$0 /month

New to Shiny? Deploy your applications for FREE.

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

Sign Up Now

STARTER

$13 /month

(or $145/year)

More applications. More active hours!

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

Sign Up Now

BASIC

$49 /month

(or $550/year)

Take your users to the next level!

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

Sign Up Now

STANDARD

$119 /month

(or $1,330/year)

Password protection? Authenticate your users!

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

Sign Up Now

PROFESSIONAL

$349 /month

(or $3,860/year)

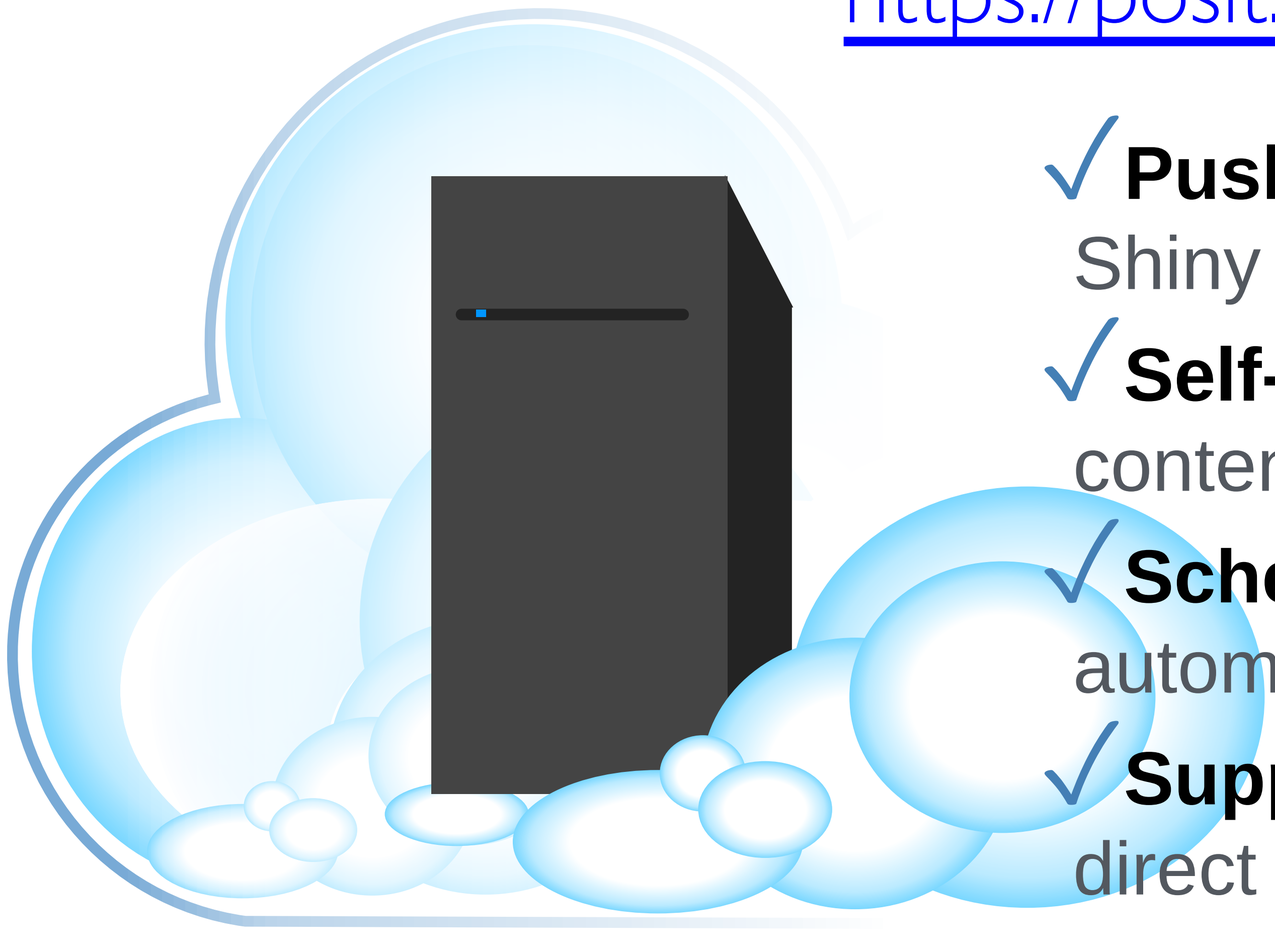
Professional has it all! Personalize your domains.

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

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

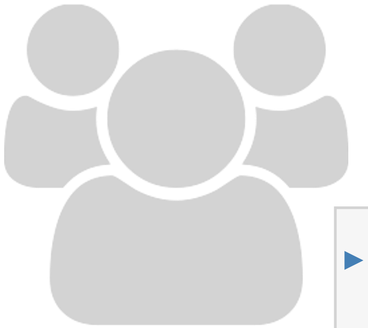
SHINY SERVER

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

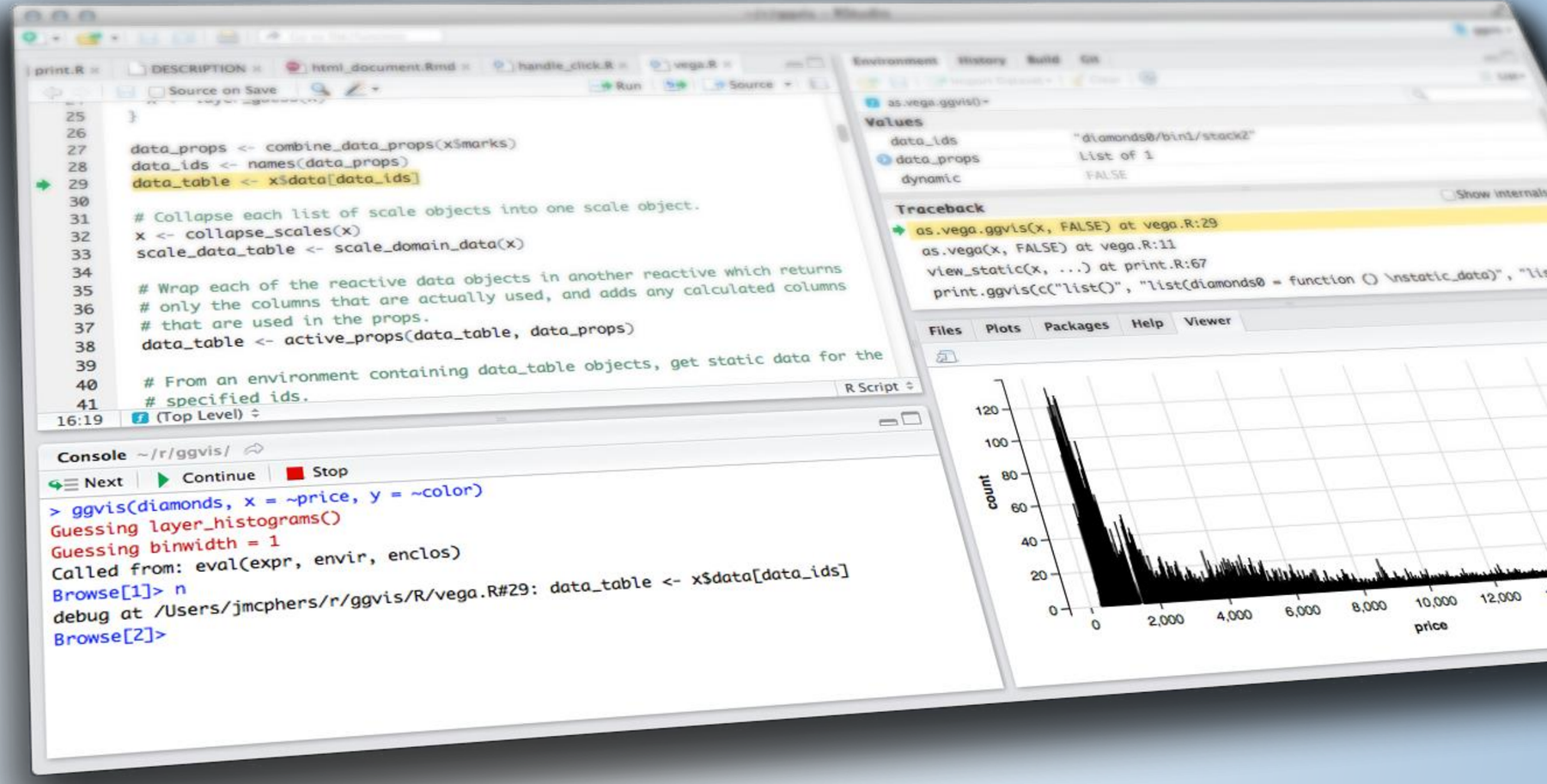
Free & open source!

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®