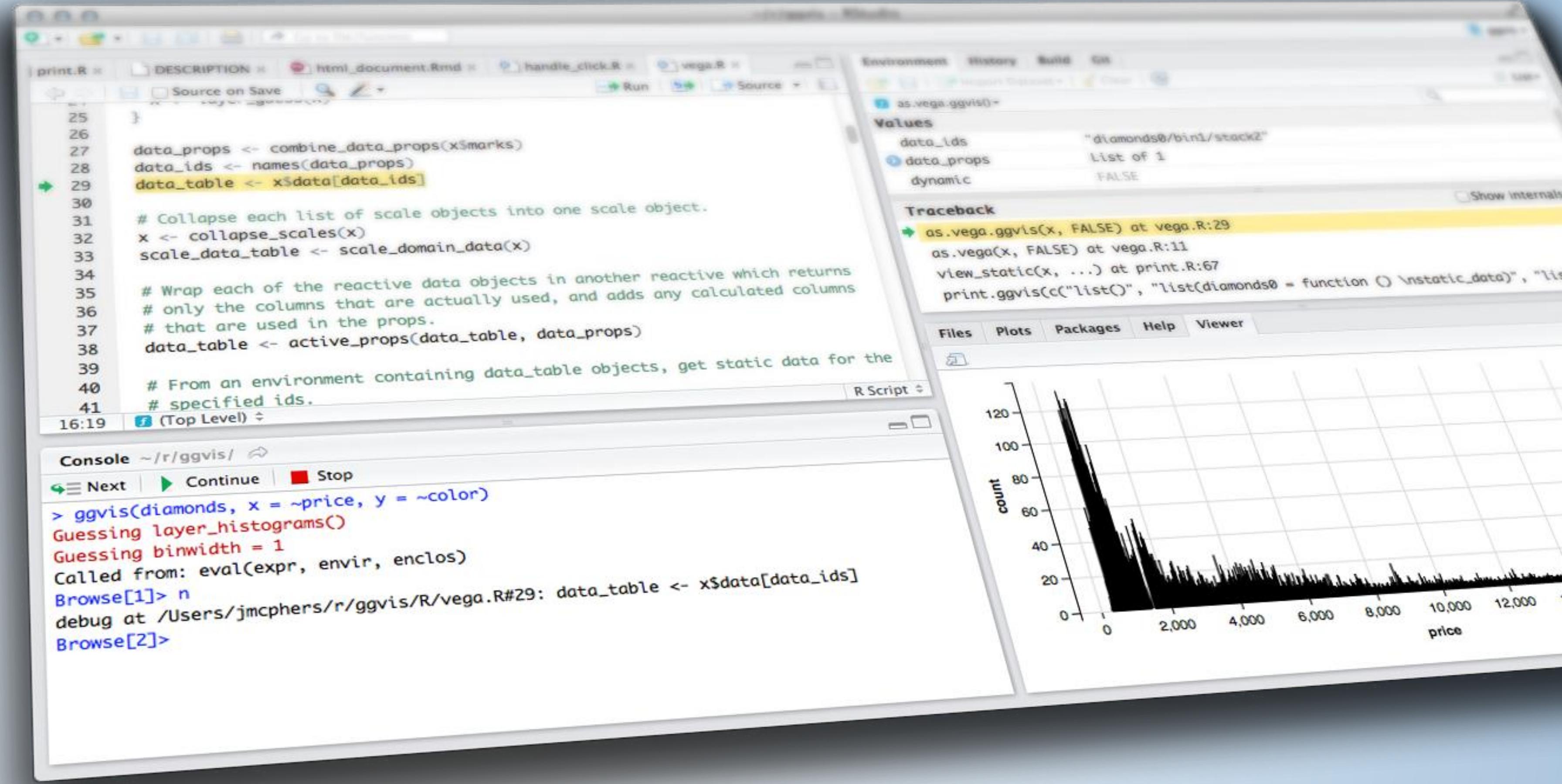


# COURSE OVERVIEW

& INTRODUCTION TO GITHUB & SHINY



Shiny from  R Studio™

# OUTLINE

- ▶ Course Overview
  - ▶ Instructor
  - ▶ Course Schedule
- ▶ GitHub for Data Science
  - ▶ Read: Introduction to Github for Data Scientists by Rebecca Vickery
  - ▶ <https://towardsdatascience.com/introduction-to-github-for-data-scientists-2cf8b9b25fba>
- ▶ 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**

**geoffreylarnold**  
@twitter  
[gla@andrew.cmu.edu](mailto:gla@andrew.cmu.edu)

# ABOUT ME

- ▶ Coordinator of Strategic Analytics
  - ▶ Allegheny County
- ▶ MSPPM 2015
- ▶ Heinz College

# COURSE SCHEDULE

Class 1 - 10/26 - Course Overview & Introduction to GitHub & Shiny

Class 2 - 11/2 - Reactive Programming & User Interfaces

Class 3 - 11/9 - Reactive Programming Pt. 2 & Dashboards

Class 4 - 11/16 - Interactive Visualizations & Advanced Reactivity

Class 5 - 11/30 - Leaflet & LeafletProxy

Class 6 - 12/7 - Bookmarking & Final Project Work

Class 7 - 12/14 - Connecting to Databases & API's

# ASSIGNMENTS

- ▶ Homework 1: Create a basic ShinyApp
  - ▶ Due: 11/11
- ▶ Homework 2: Create a Dashboard
  - ▶ Due: 12/2
- ▶ Final Project: Create a ShinyApp with an Interactive Map
  - ▶ Due: 12/16

# OFFICE HOURS

- ▶ Offer Regular Zoom help sometime during the week
- ▶ Offer a Course Slack Channel for “on-demand” help

# Intro to Github



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

–Rebecca Vickery

# WHAT IS GITHUB?

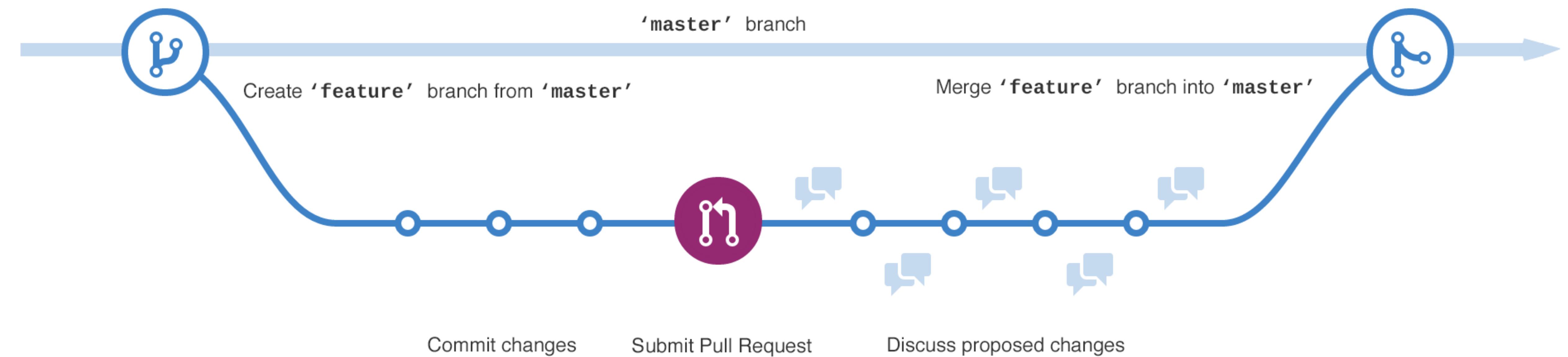
- Git is a Version Control Software
- Github stores the files for your project in a remote location and checks the differences as you change your code
- This allows you to roll back to previous versions of your project if you need to go back
- Makes sharing and collaboration much easier using the Github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



# Github Desktop

## and the Web



# EXERCISE

- ▶ Download Github Desktop: <https://desktop.github.com/>
- ▶ Sign up for Github: <https://github.com/join>
- ▶ Got to course page:  
<https://github.com/orgs/RforOperations2022/>
  - ▶ Clone Class 2 repo:  
<https://github.com/RforOperations2022/Class-2>
  - ▶ Create your own branch as your CMU username

5m 00s

# Github and 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

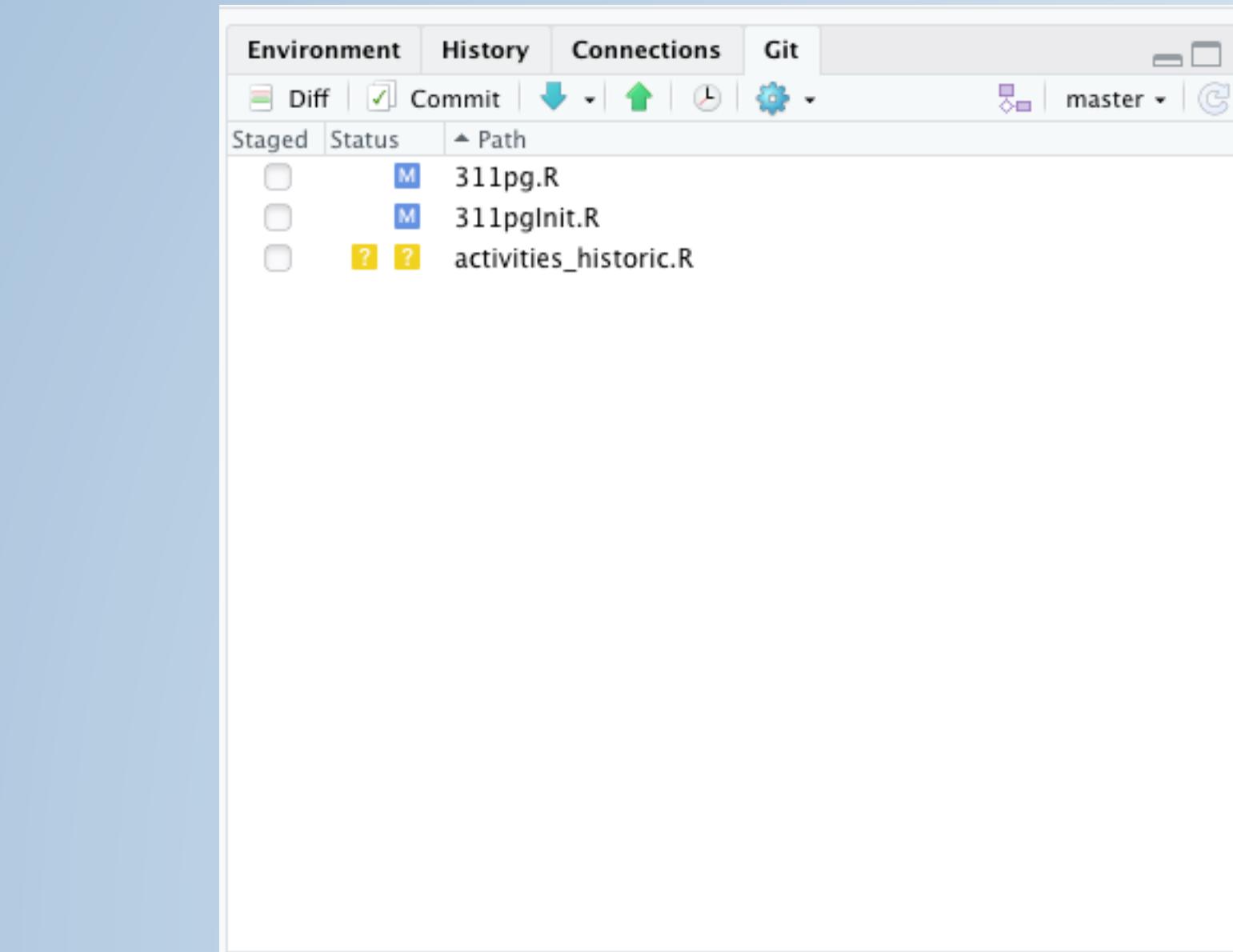
5m 00s



# EXERCISE

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

5m 00s



RStudio: Review Changes

Changes History master Stage Revert Ignore Pull Push

Staged	Status	Path
<input type="checkbox"/>	M	311pg.R
<input type="checkbox"/>	M	311pgInit.R
<input type="checkbox"/>	?	activities_historic.R

Commit message

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,
11 password = pg$pgPW)
12 sql <- 'SELECT DISTINCT(req.id)
13 FROM qalert.requests req
14 LEFT JOIN qalert.activity act
15     ON req.id = act."requestId"
16
17 WHERE req.id IS NOT NULL AND act.requestId IS NULL'
18
19 # Write new activities
20 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
21 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
22
23 delete <- paste(unlist(unique(activity$id)), collapse = ", ") # Include Id's for new requests & non-master requests
24 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
25 del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
26
27 RPostgreSQL::dbDisconnect(conn)
28
29 No newline at end of file
```

@@ -79,8 +79,8 @@ activity <- since\$activity %>%
79 delete <- paste(unlist(unique(since\$activity\$id)), collapse = ", ") # Include Id's for new requests & non-master requests
80 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
81 del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
82
83 # Write new activities
84 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
85 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
86
87 delete <- paste(unlist(unique(activity\$id)), collapse = ", ") # Include Id's for new requests & non-master requests
88 sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ")");
89 del <- RPostgreSQL::dbSendQuery(conn, sql)
90
91 RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
92 Sys.sleep(3)
93
94 dbDisconnect(conn)
95
96 No newline at end of file



# EXERCISE

- ▶ Go to “Global Options”
  - ▶ Click “Git/SVN”
    - ▶ Ensure “Enable version control interface for Studio projects” is selected
      - ▶ Click “Create RSA 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



# EXERCISE

- ▶ Click the “Git” tab
  - ▶ Click “Commit”
    - ▶ Type a message into the “Commit message” box
    - ▶ Stage the files by making sure they are selected on the left
    - ▶ Click “Commit”
    - ▶ Click “Push”
    - ▶ Login to GitHub with either your password or personal access token

5m 00s

# Github Desktop

The screenshot shows the GitHub desktop application interface. At the top, the menu bar includes File, Edit, View, Repository, Branch, and Help. The title bar displays the current repository as "desktop" and the current branch as "esc-pr" with a green checkmark and the number "#3972". A "Fetch origin" button indicates it was last fetched 2 minutes ago.

The main area has two tabs: "Changes" and "History". The "History" tab is selected, showing a list of commits:

- Appease linter
- Add event handler to dropdown component
- Move escape behavior to correct component
- Remove event handler from the branches..
- Merge branch 'master' into esc-pr
- Merge pull request #4044 from desktop/...
- Merge pull request #4070 from desktop/...
- bump to beta3
- Merge pull request #4057 from desktop/...
- Merge pull request #4067 from desktop/...
- Release to 1.1.0-beta2

Details for the "Add event handler to dropdown component" commit are expanded:

**Commit Details:** iAmWillShepherd and Markus Olsson committed c79e71c 1 changed file  
Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>

**File:** app\src\ui\toolbar\dropdown.tsx

**Diff View:**

Line	Line	Text
145	145	@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
146	146	this.state = { clientRect: null }
147	147	}
148	148	+ private get isOpen() {
149	149	+     return this.props.dropdownState === 'open'
150	150	+ }
151	151	+
148	152	152     private dropdownIcon(state: DropdownState): OcticonSymbol {
149	153	153         // @TODO: Remake triangle octicon in a 12px version,
150	154	154         // right now it's scaled badly on normal dpi monitors.
249	253	@@ -249,6 +253,13 @@ export class ToolbarDropdown extends React.Component<
250	254	}
251	255	}
256	256	+ private onFoldoutKeyDown = (event: React.KeyboardEvent<HTMLElement>) => {
257	257	+     if (!event.defaultPrevented && this.isOpen && event.key === 'Escape') {
258	258	+         event.preventDefault()

# Github in the command line / terminal

# WHAT'S 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.



# EXERCISE

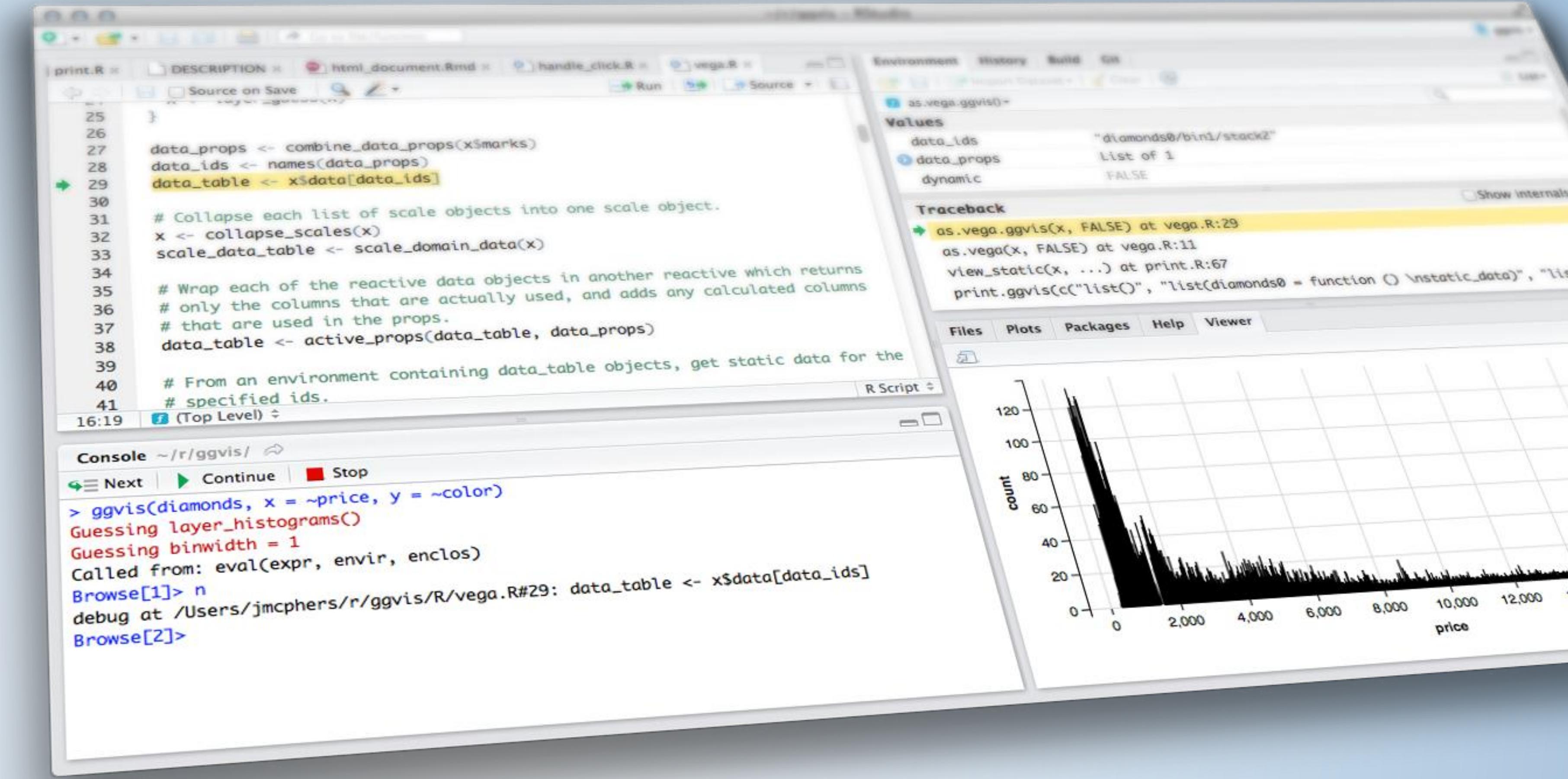
- ▶ <https://learngitbranching.js.org/>
- ▶ Complete Introduction Sequences
  - ▶ Git Commits
  - ▶ Branching Git
  - ▶ Merging in Git

5<sub>m</sub> 00<sub>s</sub>

# *Shiny Examples*

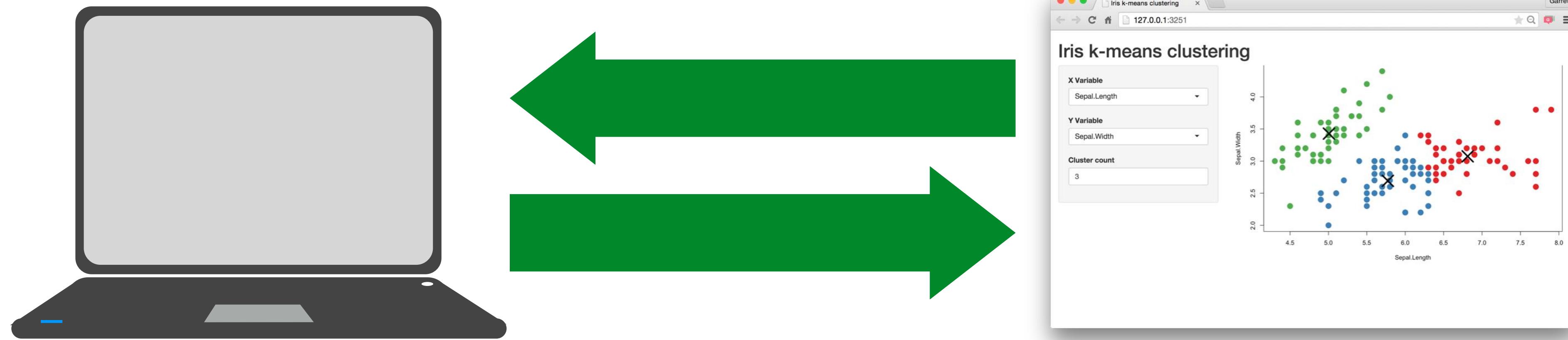
- [Burghs Eye View](#)
- [Covid-19 Tracker](#)
- [Port Authority Bus Tracker](#)
- [British Columbia CCISS Tool](#)
- [Commute Explorer](#)

# CLASS BREAK

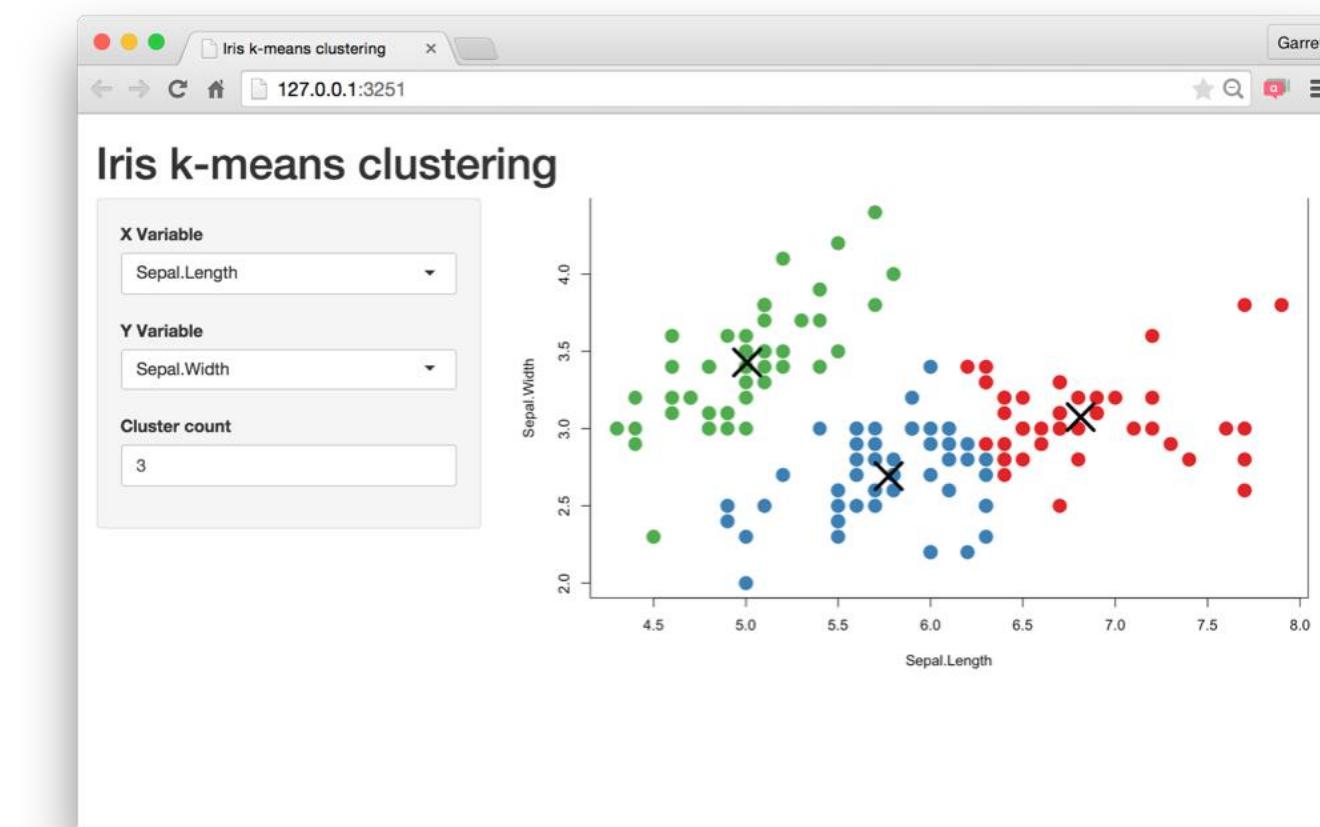
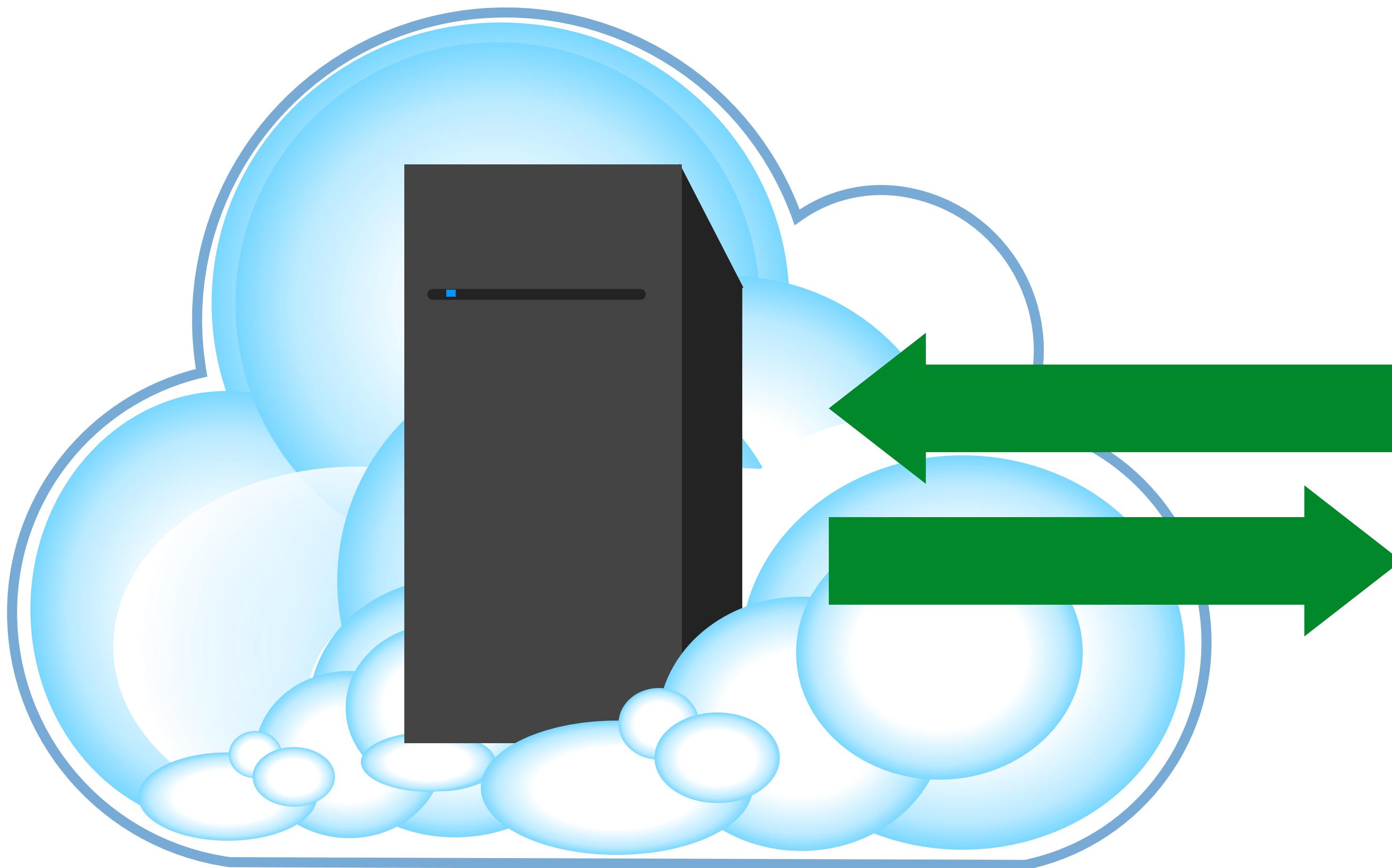


Shiny High level  
view

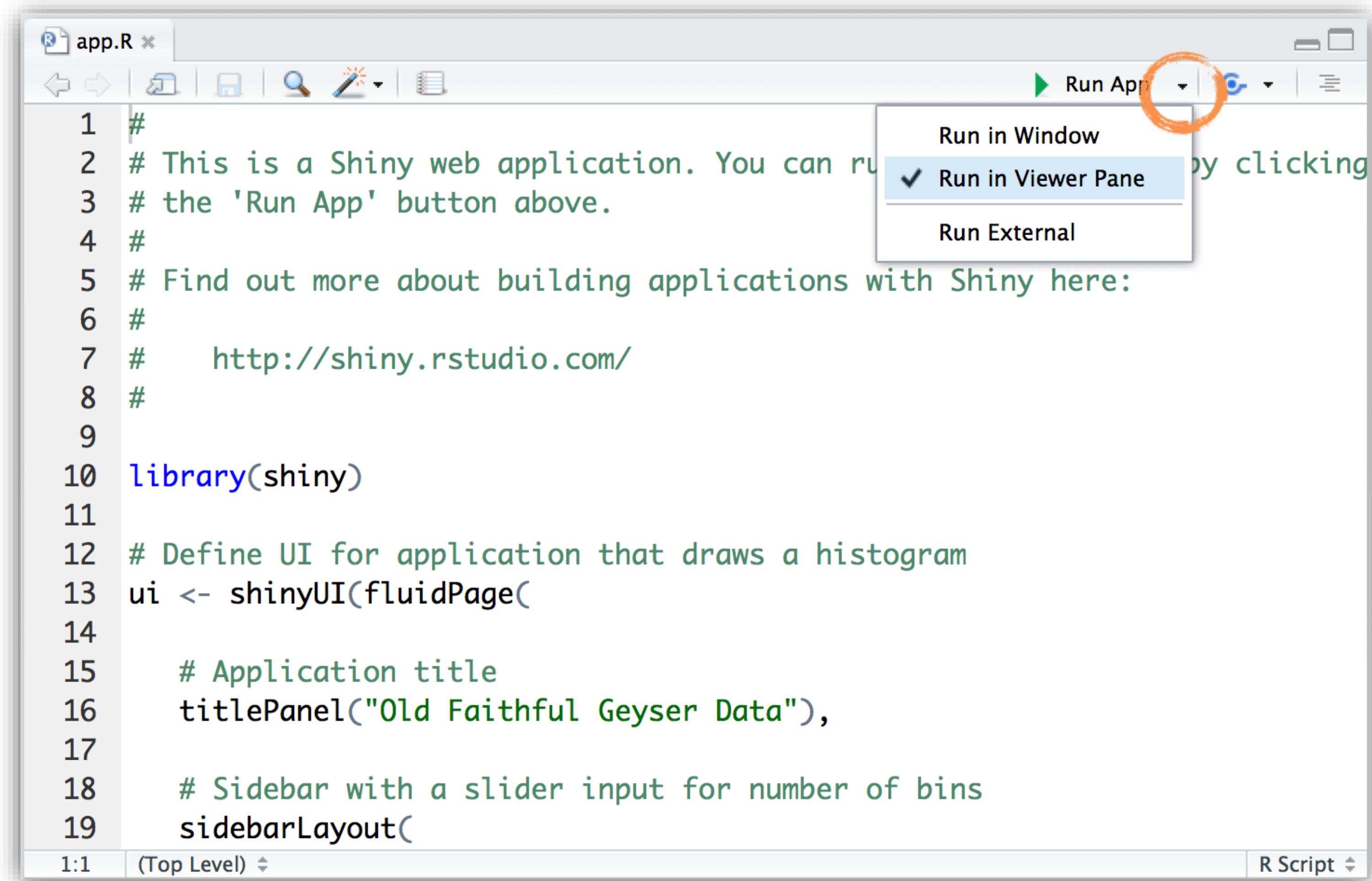
# Every Shiny app is maintained by a computer running R



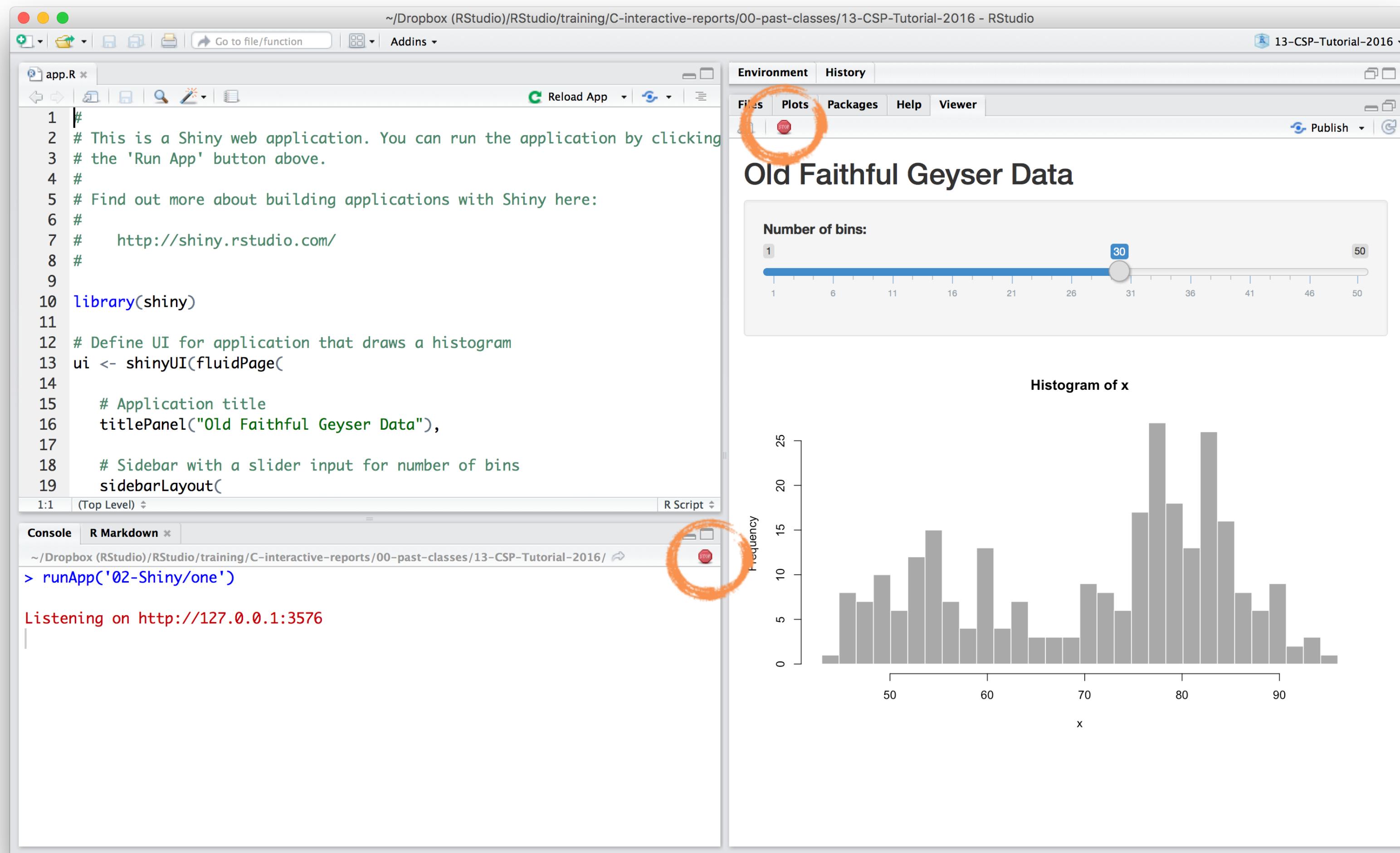
# Every Shiny app is maintained by a computer running R



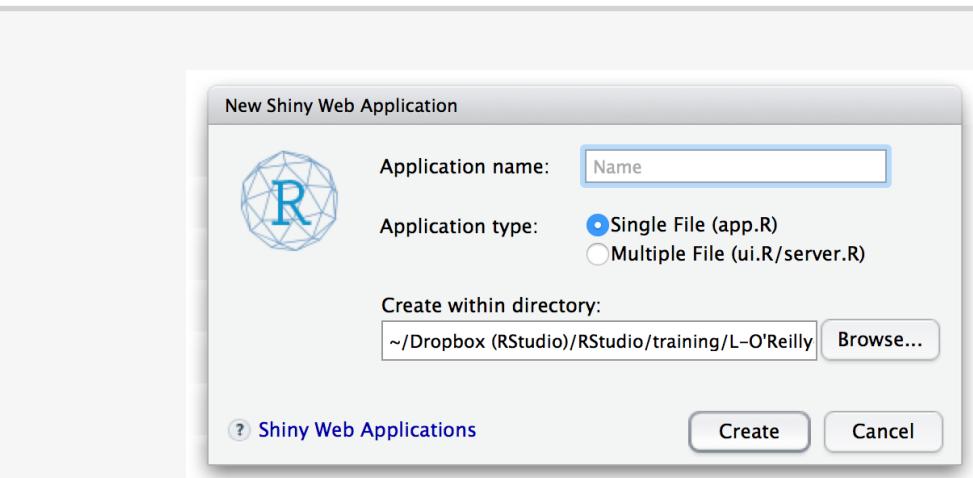
# Change display



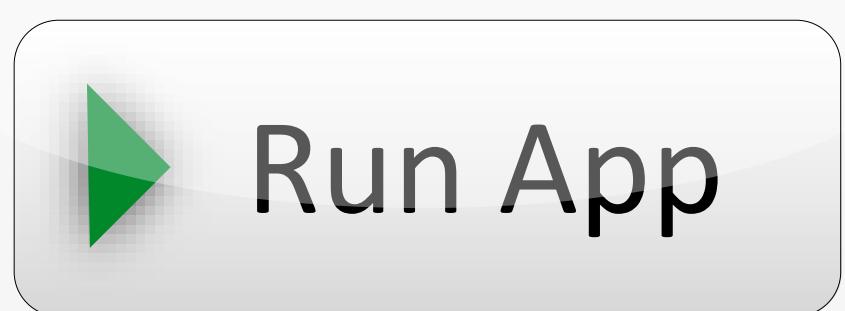
# Close an app



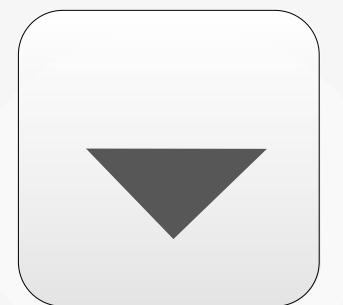
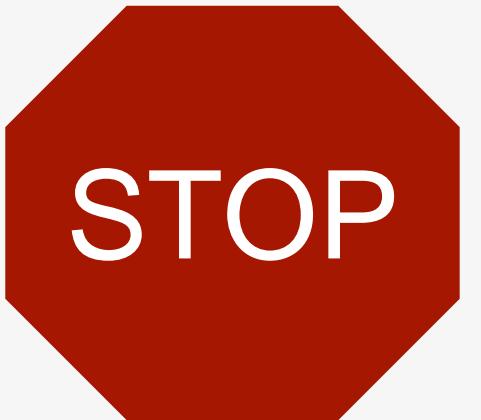
# EXERCISE



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

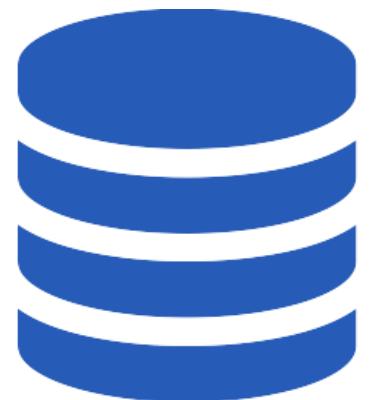
controls the layout and appearance of app

## Server function

contains instructions needed to build app

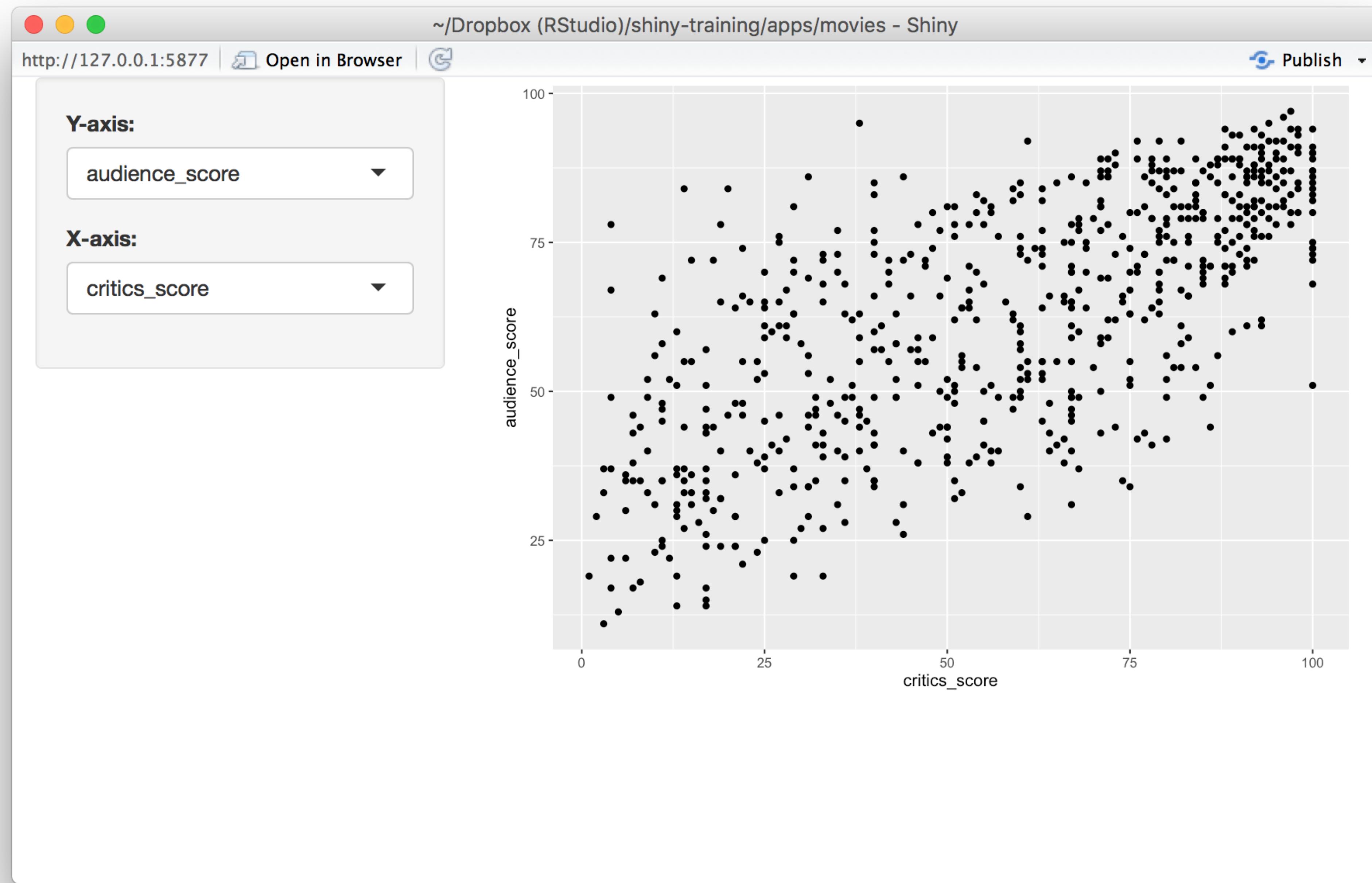


# Let's build a simple movie browser app!



movies.Rdata

Data from IMDB and Rotten Tomatoes on random sample of 651 movies released in the US between 1970 and 2014

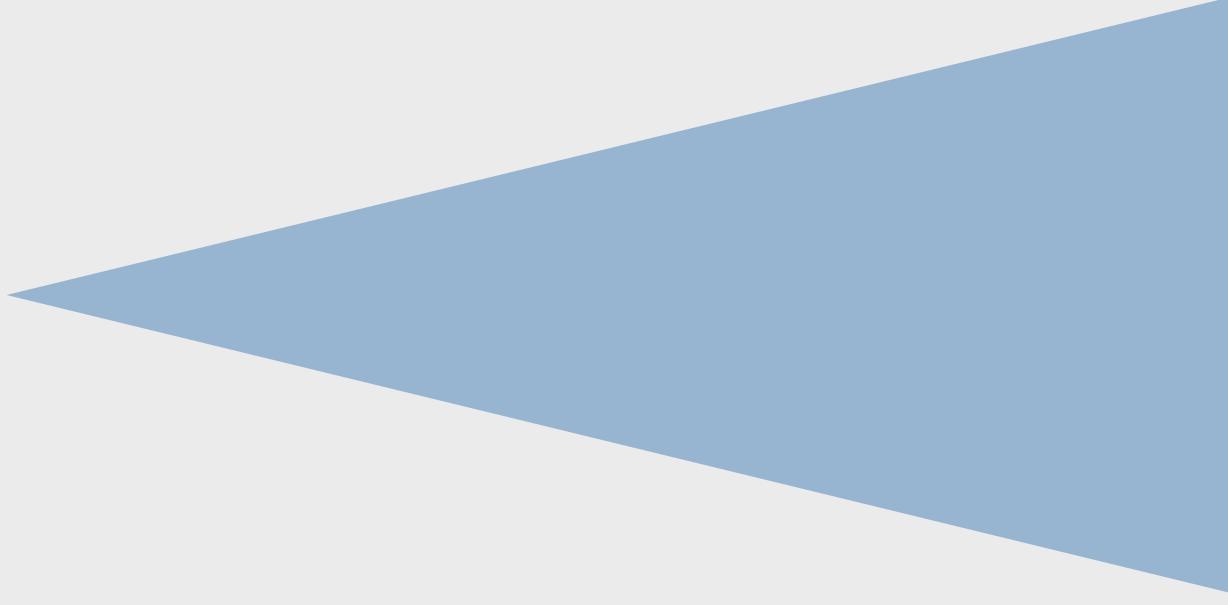


# APP TEMPLATE

```
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")
    )
  )
)
```

```
# 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 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

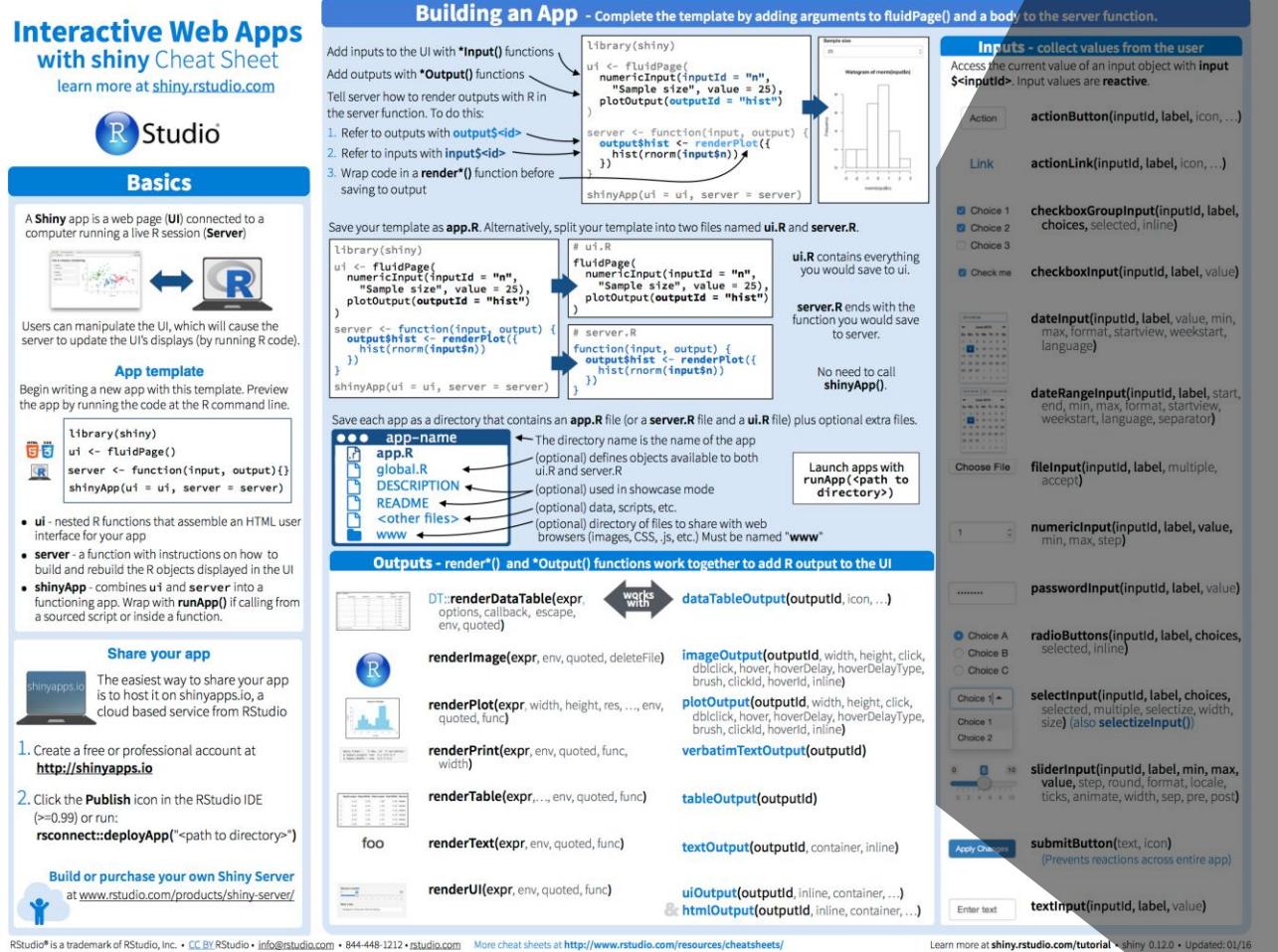
A large, light gray checkmark icon inside a circle, indicating a correct answer or solution.

# SOLUTION

## Solution to the previous exercise

`movies_02.R`

# INPUTS





# 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/](https://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

A large, light gray checkmark icon inside a circle, positioned in the top-left corner.

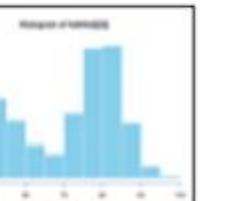
# SOLUTION

## Solution to the previous exercise

`movies_03.R`

# OUTPUTS

**works with**

<p><b>DT::renderDataTable(expr, options, callback, escape, env, quoted)</b></p>  <p><b>renderImage(expr, env, quoted, deleteFile)</b></p>  <p><b>renderPlot(expr, width, height, res, ..., env, quoted, func)</b></p>  <p><b>renderPrint(expr, env, quoted, func, width)</b></p> <p><b>renderTable(expr, ..., env, quoted, func)</b></p> <p><b>renderText(expr, env, quoted, func)</b></p> <p><b>renderUI(expr, env, quoted, func)</b></p> 	<p><b>dataTableOutput(outputId, icon, ...)</b></p> <p><b>imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)</b></p> <p><b>plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)</b></p> <p><b>verbatimTextOutput(outputId)</b></p> <p><b>tableOutput(outputId)</b></p> <p><b>textOutput(outputId, container, inline)</b></p> <p><b>uiOutput(outputId, inline, container, ...)</b>  <b>&amp; htmlOutput(outputId, inline, container, ...)</b></p>
---	--

**Interactive Web Apps with shiny Cheat Sheet**  
 learn more at [shiny.rstudio.com](http://shiny.rstudio.com)

**R Studio Basics**

A Shiny app is a web page (UI) connected to a computer running an R session (Server).  
 Users can manipulate the UI, which will cause the server to update the UI displays (by running R code).  
 A template  
 Begin writing a new app with this template. Preview the app by running the code at the command line.  
 • ui - nested R functions that assemble an HTML user interface for your app  
 • build and rebuild the R objects displayed in the UI  
 • shinyApp - combines ui and server into a functioning app. Wrap with runApp() if calling from a source script or inside a function.  
 Share your app  
 The easiest way to share your app is to host it on shinyapps.io, a cloud based service from RStudio.  
 1. Create a free or professional account at <http://shinyapps.io>.  
 2. Click the Publish icon in the RStudio IDE ( $\geq 0.99$ ) or run: rconnect::deployApp("path to directory")  
 Build or purchase your own Shiny Server at [www.rstudio.com/products/shiny-server/](http://www.rstudio.com/products/shiny-server/)

**Building an App** - Complete the template by adding arguments to fluidPage() and a body to the server function.

Add inputs to the UI with `input()` functions  
 Add outputs with `output()` functions  
 Tell server how to render outputs with R in the server function. To do this:  
 1. Refer to outputs with `output$<id>`  
 2. Refer to inputs with `input$<id>`  
 3. Wrap code in a `render()` function before saving to output

Save your template as `app.R`. Alternatively, split your template into two files named `ui.R` and `server.R`.

library(shiny)  
 ui <- fluidPage(  
 numericInput("n", "Number of observations", value = 25,  
 plotOutput("hist"))  
 server <- function(input, output){  
 output\$hist <- renderPlot({  
 hist(rnorm(input\$n), main = "Histogram of rnorm()")  
})  
 shinyApp(ui = ui, server = server)}

Save each app as a directory that contains an `app.R` file (or a `server.R` file and a `ui.R` file) plus optional extra files.

• app-name  
 app.R  
 DESCRIPTION  
 README  
 other files  
 www

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

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

renderImage(expr, env, quoted, deleteFile) works with imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

renderPlot(expr, width, height, res, ..., env, quoted, func) works with plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

renderPrint(expr, env, quoted, func, width) works with verbatimTextOutput(outputId)

renderTable(expr, ..., env, quoted, func) works with tableOutput(outputId)

renderText(expr, env, quoted, func) works with textOutput(outputId, container, inline)

renderUI(expr, env, quoted, func) works with 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/](https://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

A large, light gray checkmark icon inside a circle, positioned in the top-left corner.

# 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

A large, light gray checkmark icon inside a circle, positioned in the top-left corner.

# SOLUTION

## Solution to the previous exercise

`movies_05.R`

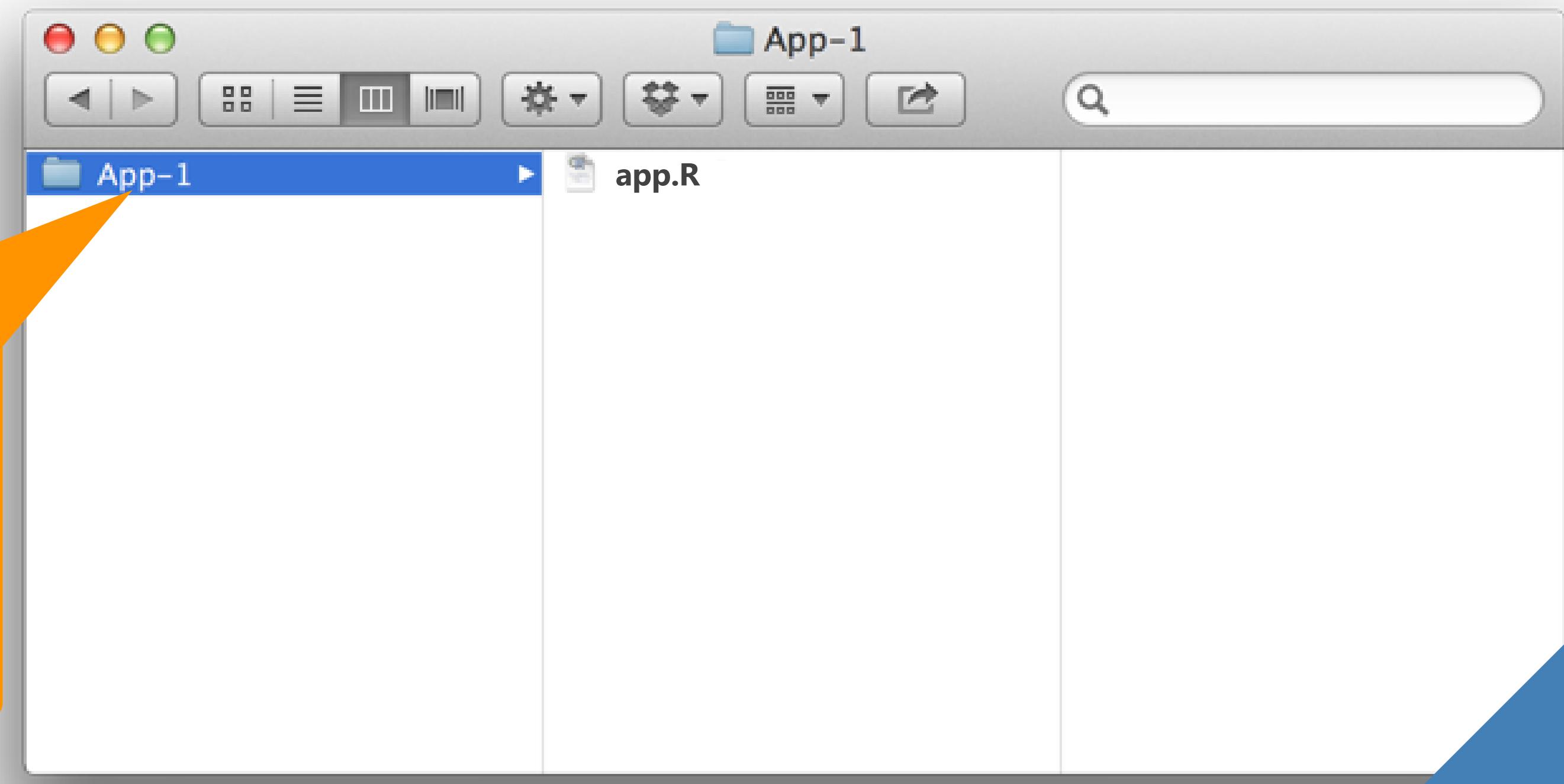
# File structure

# 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

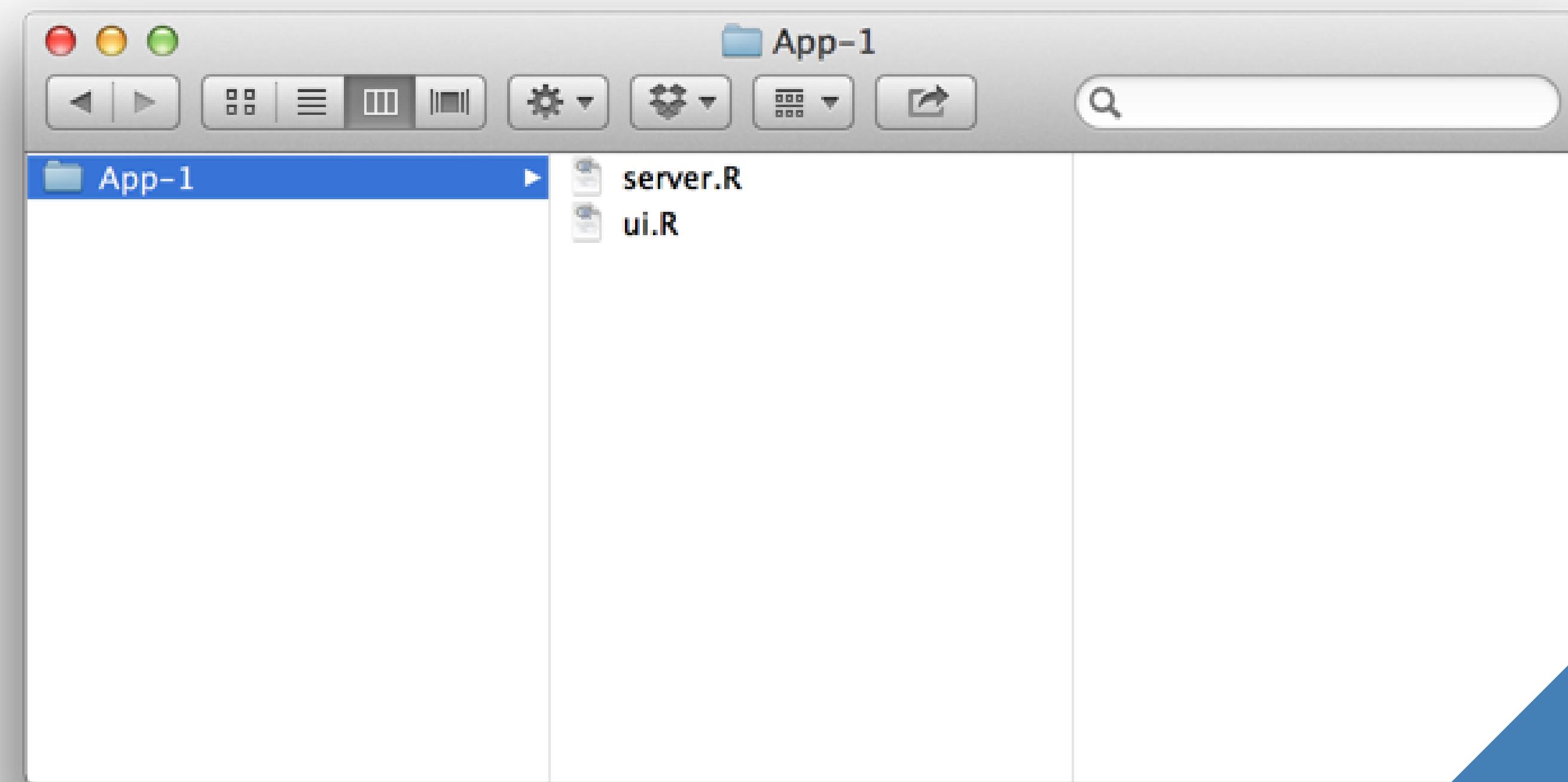


You must use this  
exact name (`app.R`)  
for deploying the app

# 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

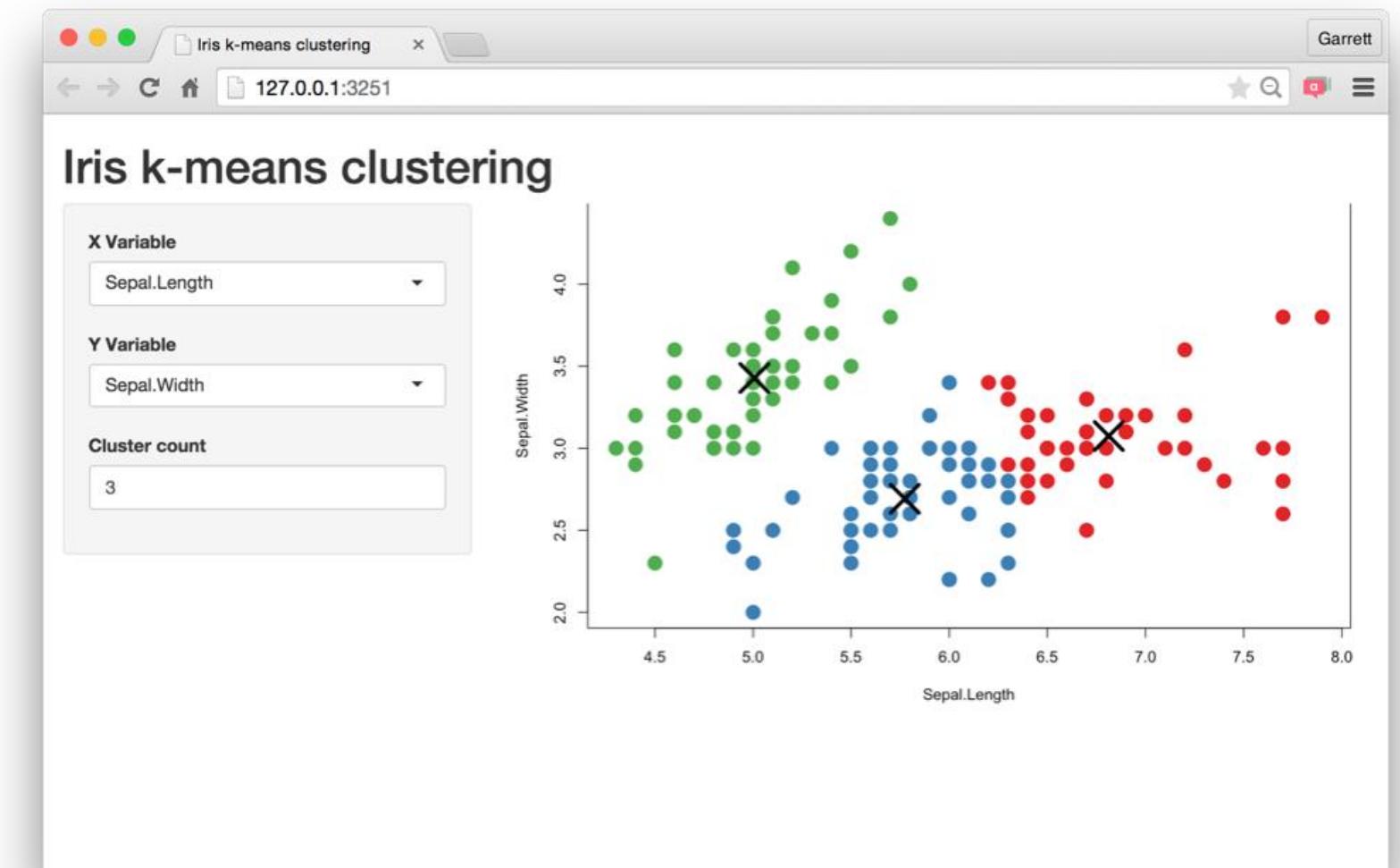
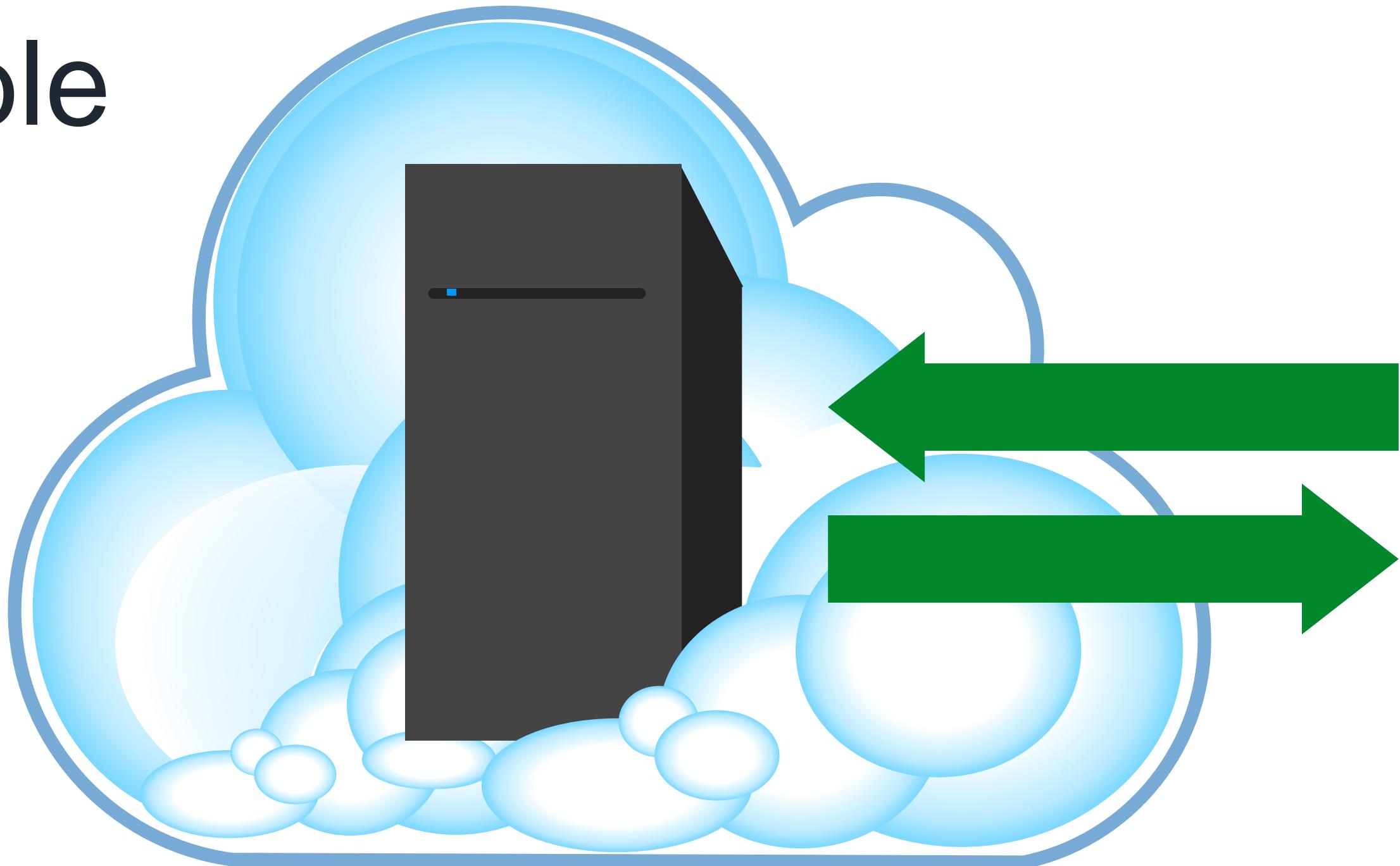
# Sharing your app

shinyapps.io



## A server maintained by RStudio

- ▶ easy to use
- ▶ secure
- ▶ scalable



# HASSE-FREE CLOUD HOSTING FOR SHINY

shinyapps.io by RStudio

Home

Features

Pricing

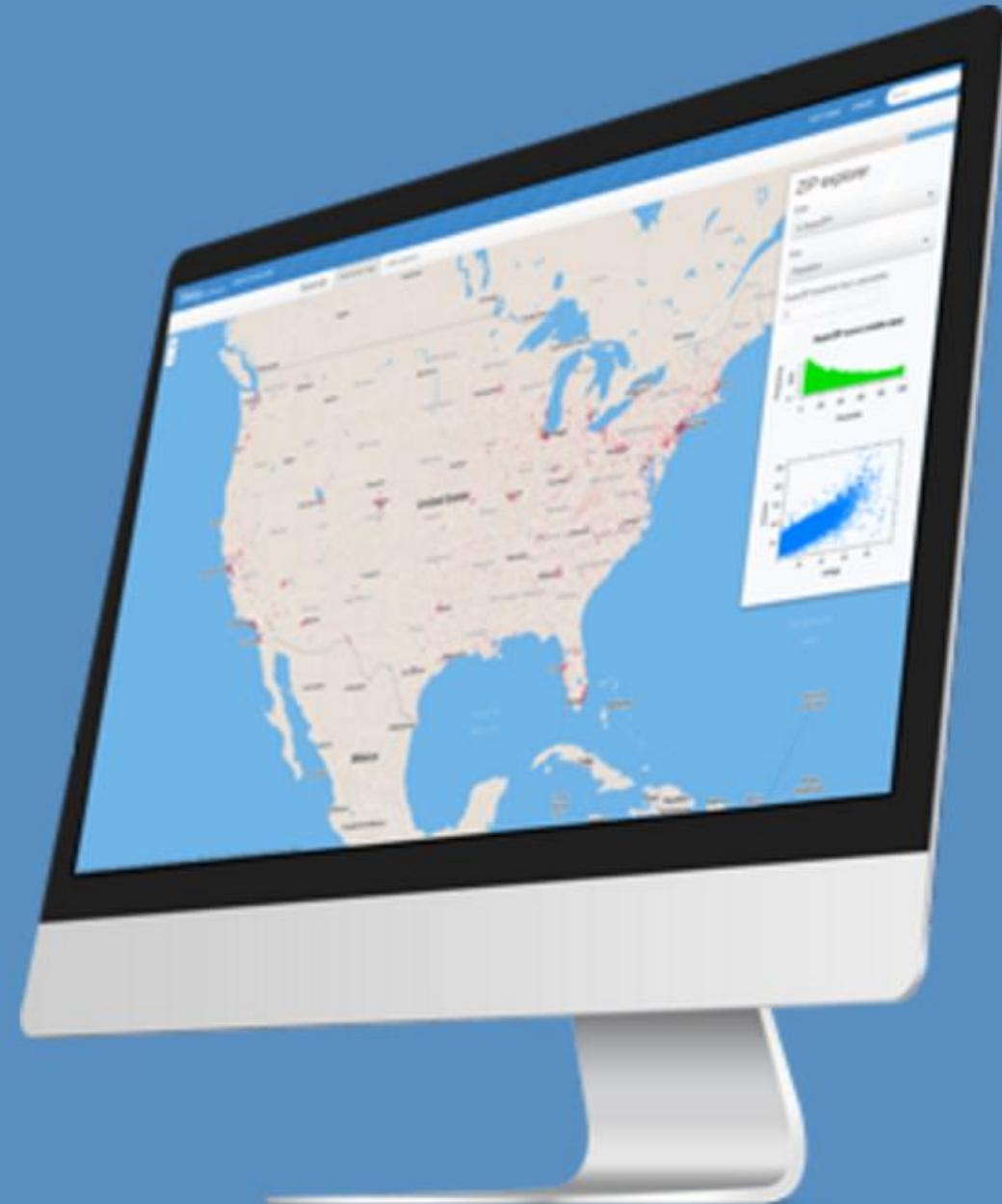
Support

Log In

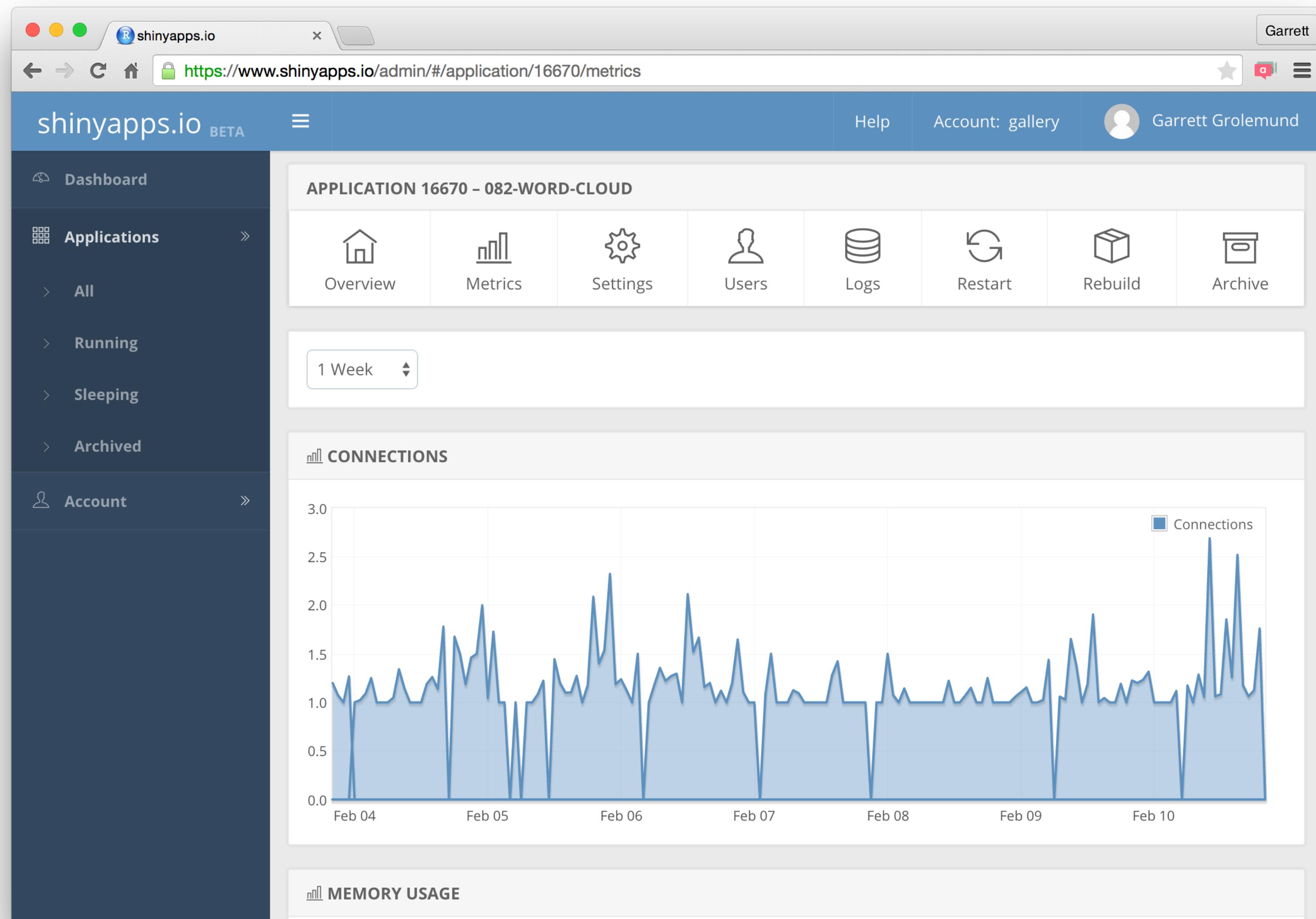
Share your Shiny  
Applications Online

Deploy your Shiny applications on the Web in minutes

Sign Up



# WITH BUILT-IN METRICS



# MEMBERSHIP PRICING

FREE

**\$ 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

Build your own  
server

# SHINY SERVER

[rstudio.com/products/shiny/shiny-server/](http://rstudio.com/products/shiny/shiny-server/)

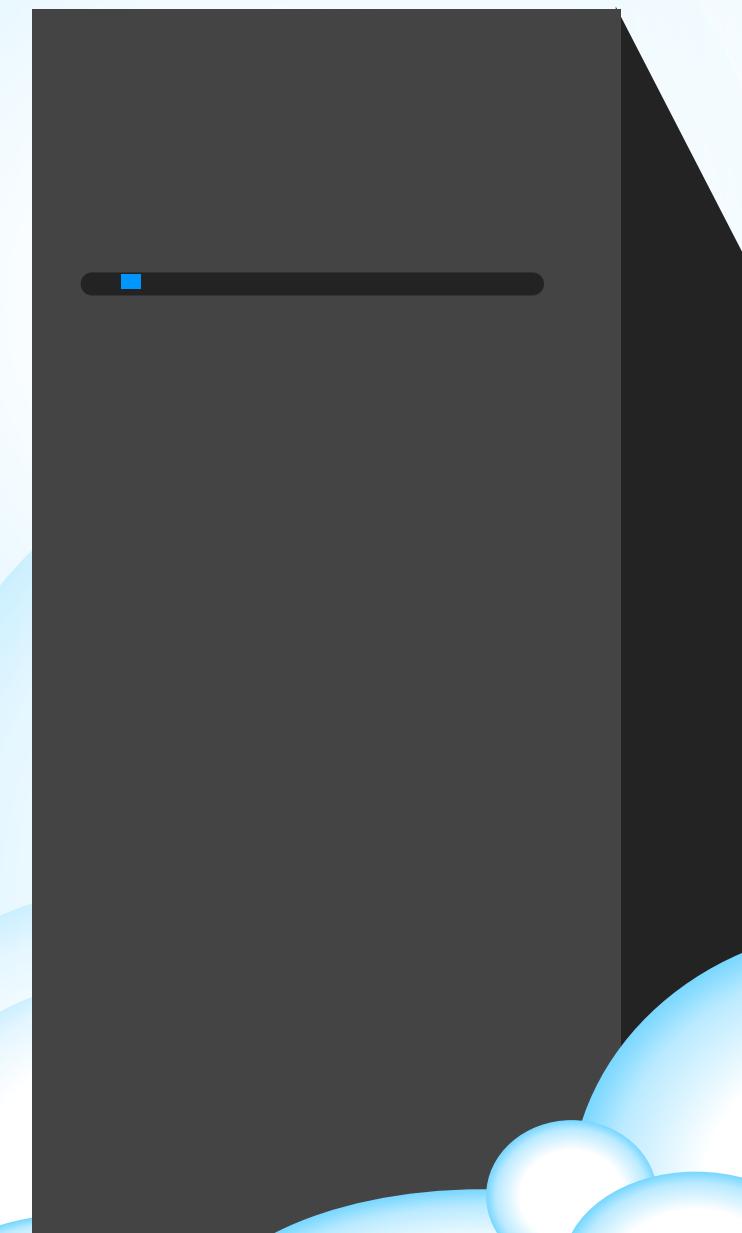


- ✓ Deploy Shiny apps to the internet
- ✓ Run on-premises  
move computation closer to the data
- ✓ Host multiple apps on one server
- ✓ Deploy inside the firewall
- ✓ xcopy deployment



# SHINY SERVER PRO

[rstudio.com/products/shiny/shiny-server/](https://rstudio.com/products/shiny/shiny-server/)



✓ **Secure access**

LDAP, GoogleAuth, SSL, and more

✓ **Performance**

fine tune at app and server level

✓ **Management**

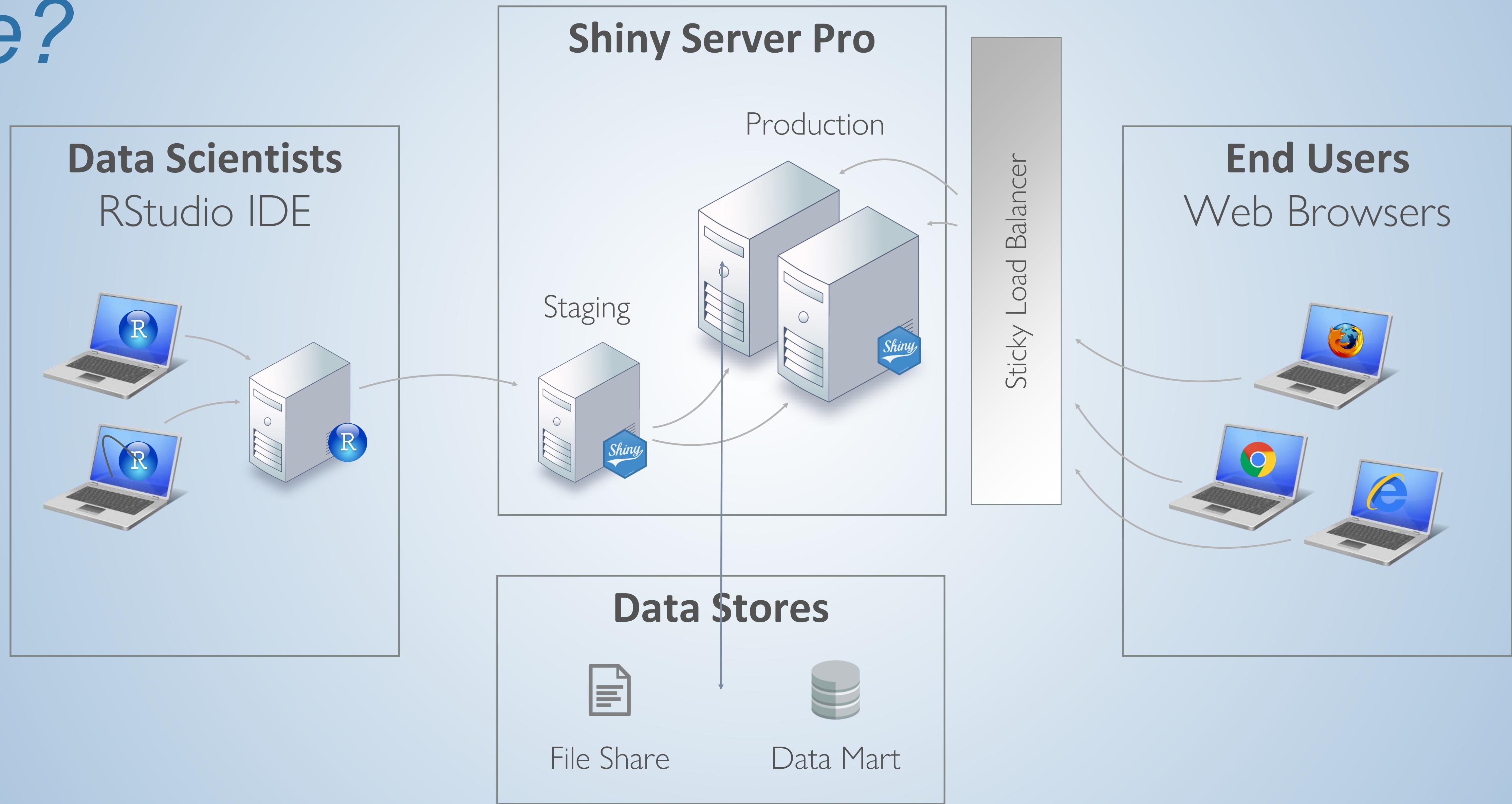
monitor and control resource use

✓ **Support**

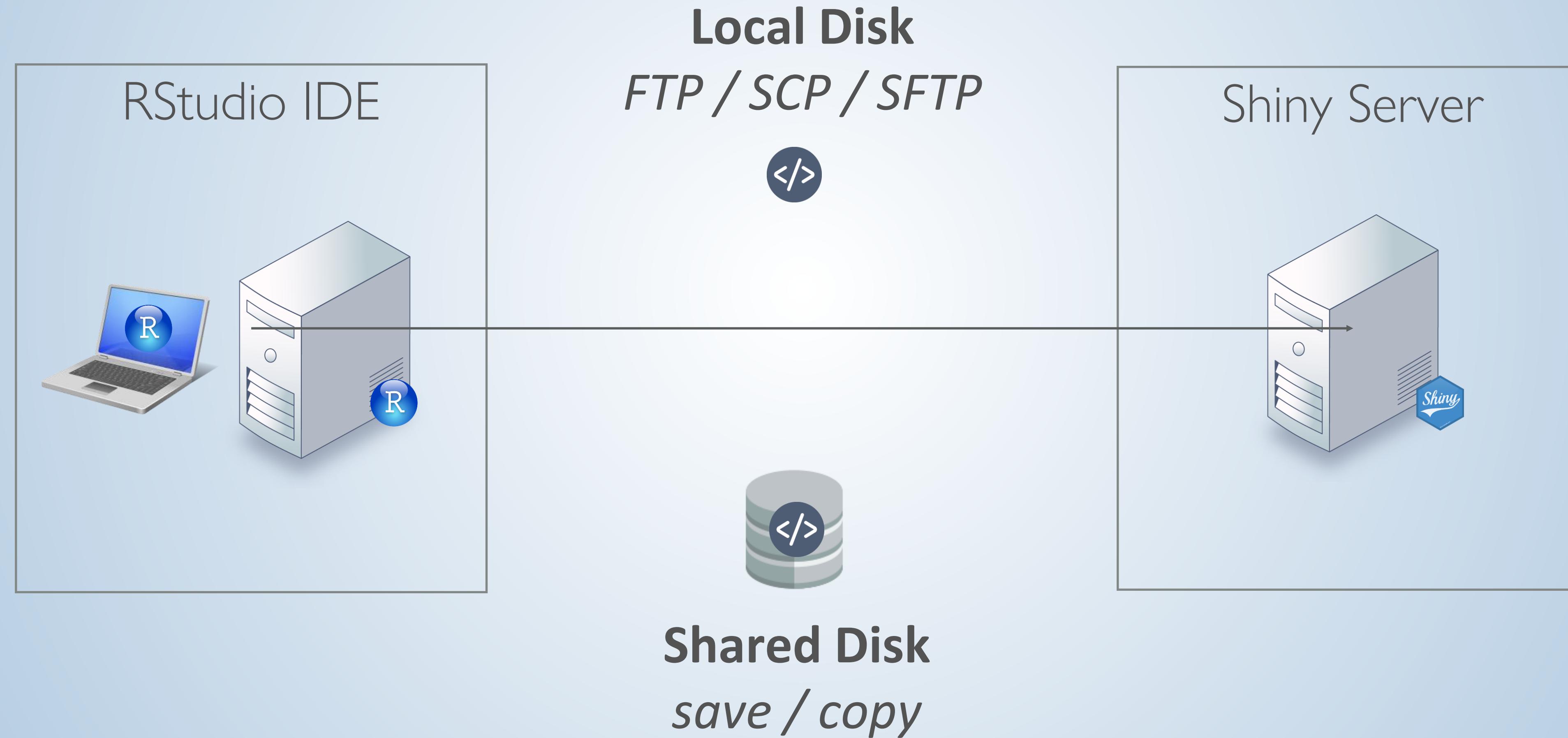
direct priority support



# What does a typical setup with SSP look like?

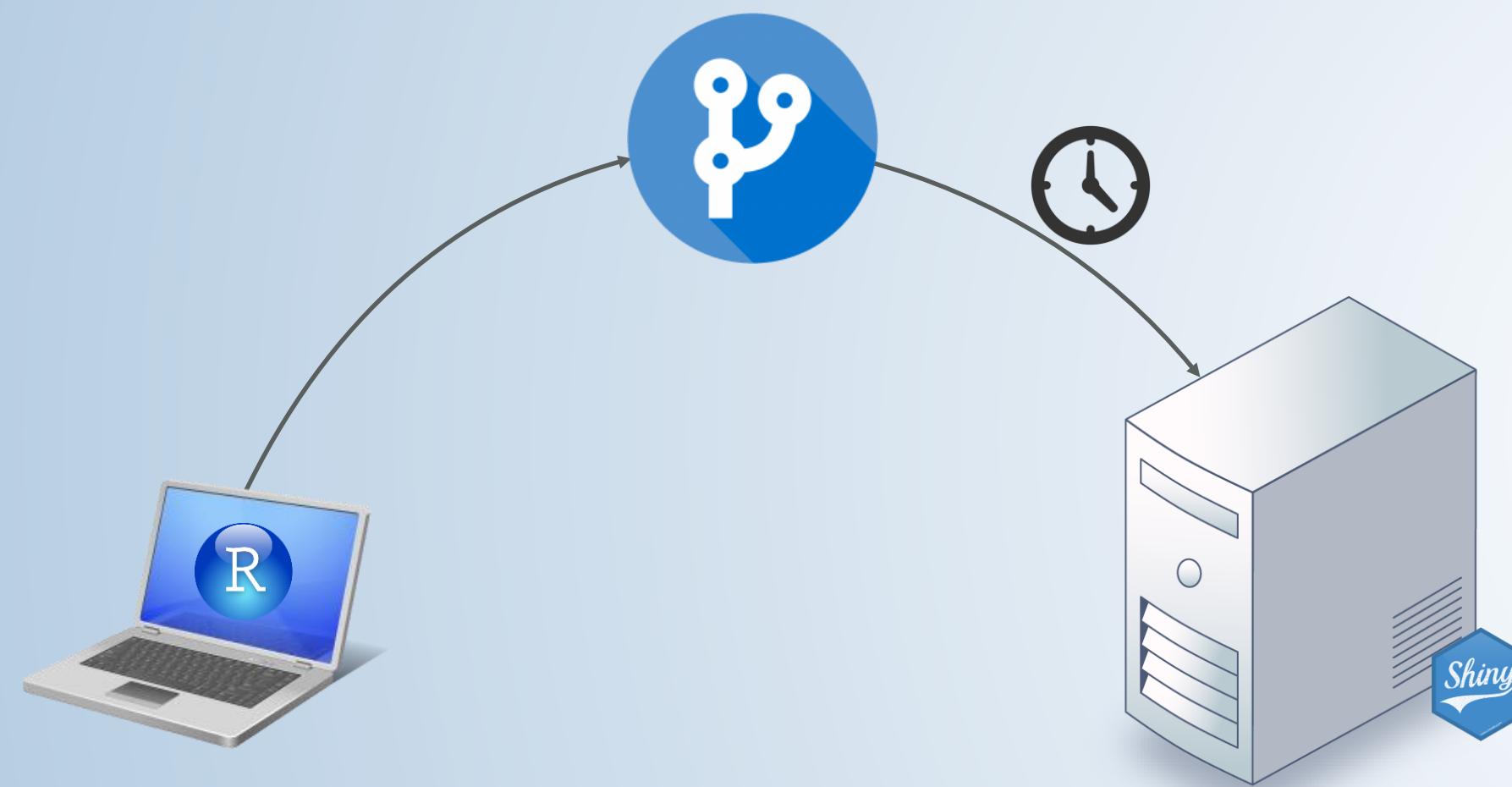


# How do I deploy apps?



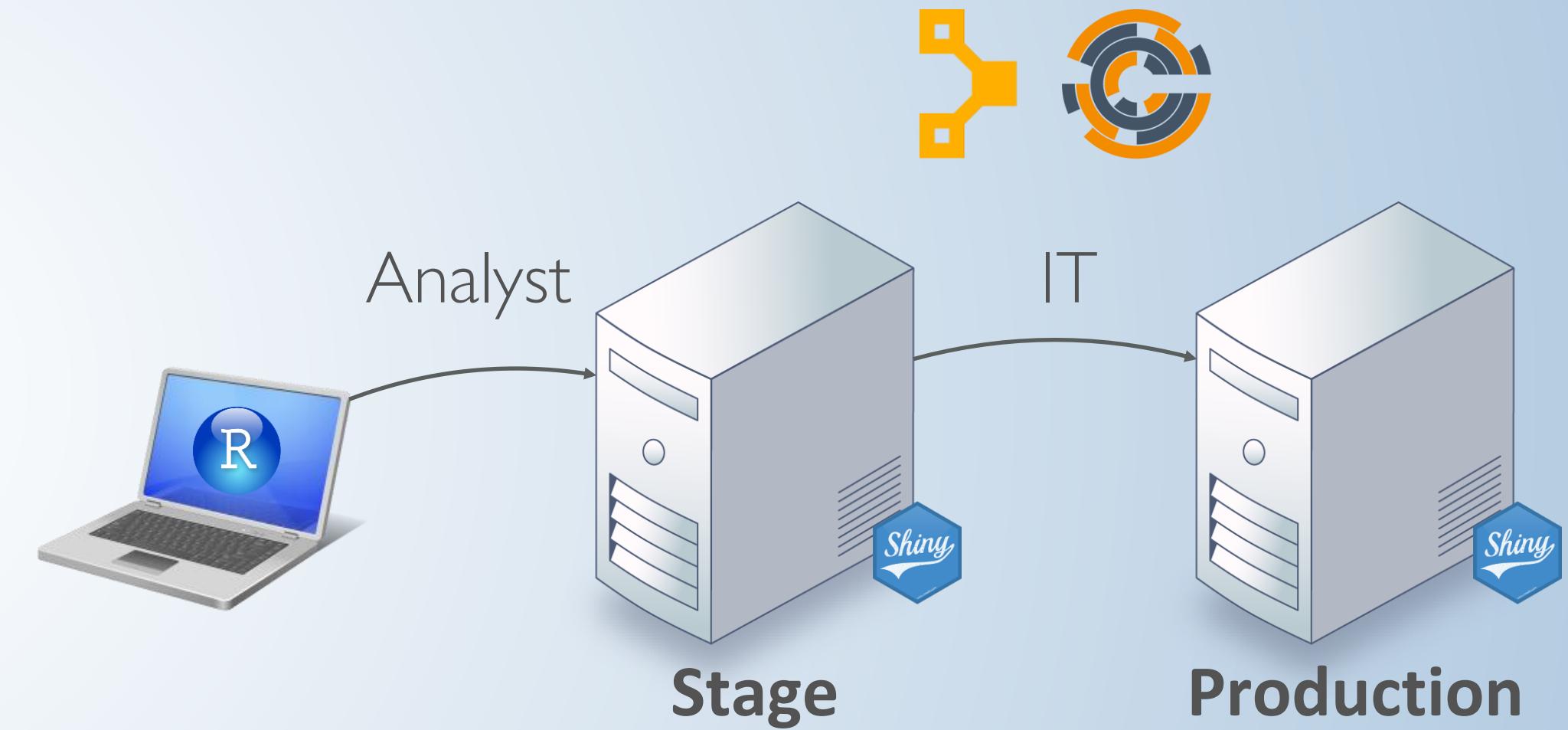
# How do I deploy apps in production?

## Version Control



Push to version control and  
automate clones via a scheduler

## Handoff



Analyst stage app  
IT deploys to production

# *More information*

- ▶ Sharing your apps tutorial:  
<https://shiny.rstudio.com/tutorial/written-tutorial/lesson7/>
- ▶ Administering Shiny Server Pro webinar:  
<https://www.rstudio.com/resources/webinars/administering-shiny-server-pro/>



# 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

# RSTUDIO CONNECT

[rstudio.com/products/connect/](https://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

# SHINYPROXY

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**

AD groups can restrict app access



# *More information*

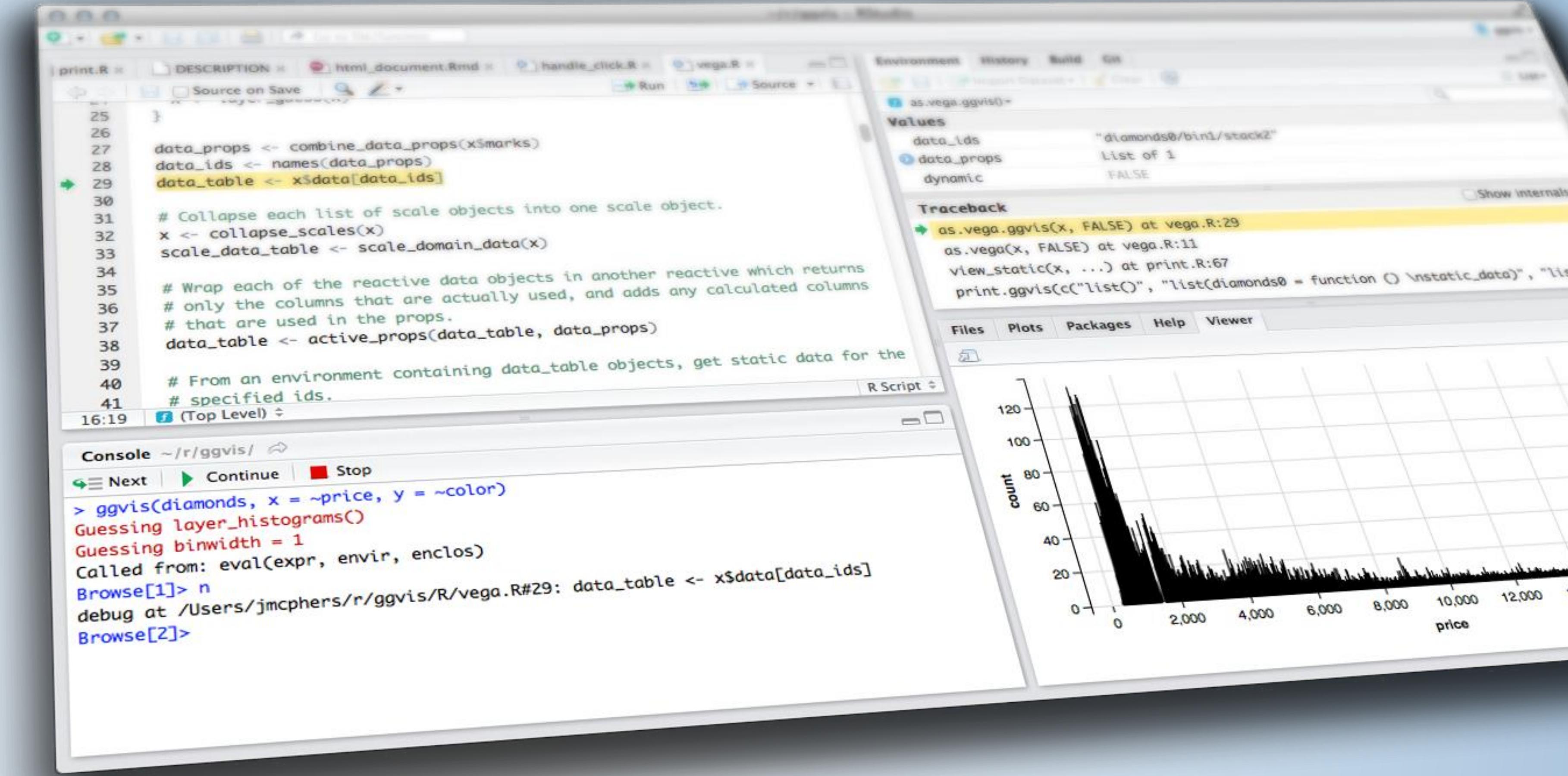
- ▶ Shinyproxy documentation:  
<https://www.shinyproxy.io/deploying-apps/>
- ▶ Shinyproxy github repo:  
<https://github.com/openanalytics/shinyproxy>

# How do I deploy apps?



# COURSE OVERVIEW

& INTRODUCTION TO GITHUB & SHINY



Shiny from  R Studio™