An Introduction to R Shiny

A Tool Demonstration by Group 6

What is Shiny?

- Code-based tool for data visualization.
- R Shiny is a web application framework for R.
- Allows creation of interactive web applications directly from R code.
- Build web apps with R Shiny without knowing HTML, CSS, or JavaScript.
- Open source and free to use.
- Has a large and supportive community of users and developers.
- Widely used in data science, finance, healthcare, and other fields for creating data-driven web applications.

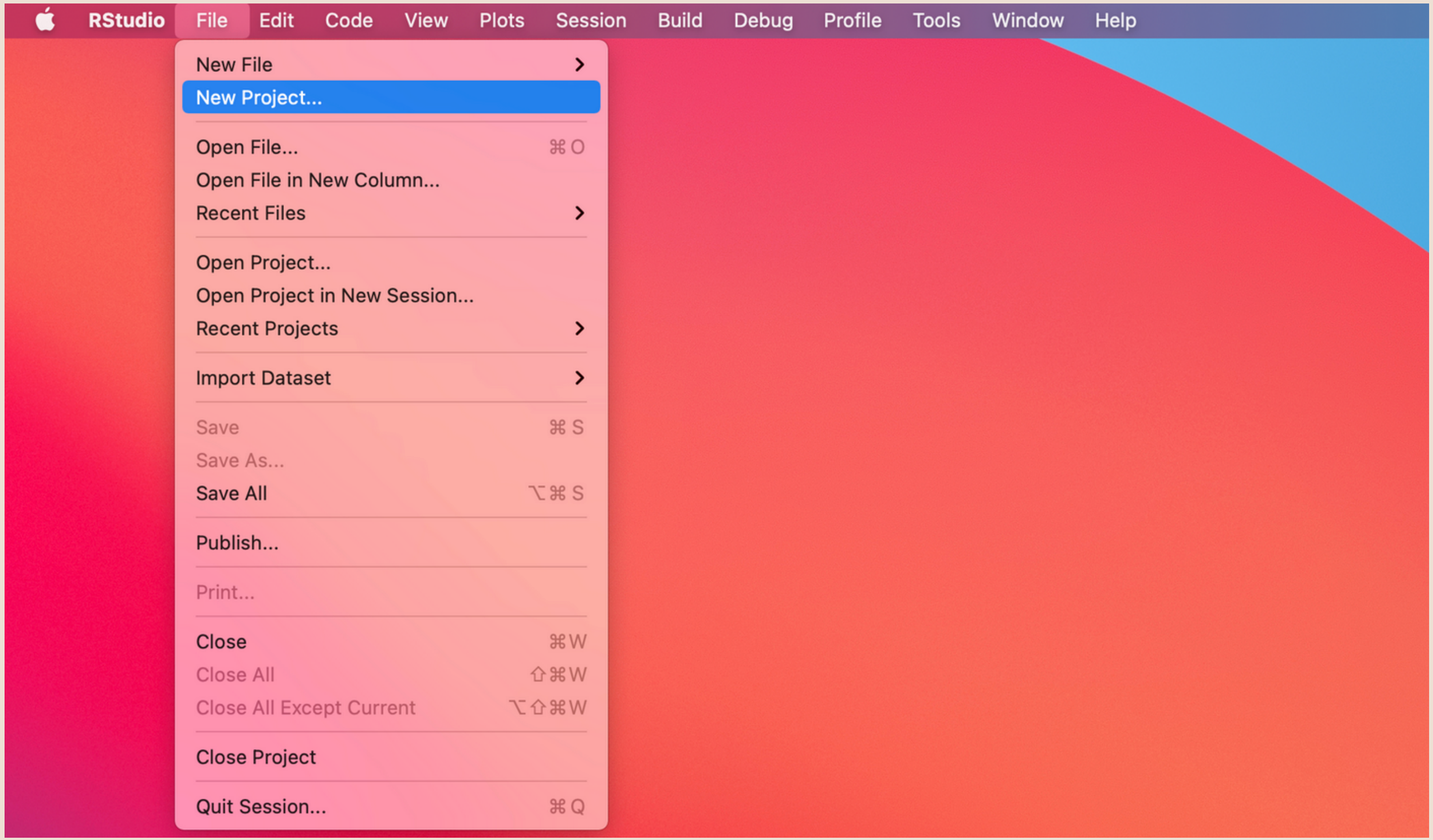
Features

- Supports reactive programming, which means that your application will update in real-time as the user interacts with it.
- Wide range of input widgets, such as sliders, dropdowns, text boxes, and buttons, that can be used to build interesting UIs.
- Built-in support for data processing and manipulation, so you can perform calculations and analysis within your application.
- Create complex layouts and data visualizations, including interactive maps and charts.
- Integrates with other web technologies, such as databases, APIs, and web services.

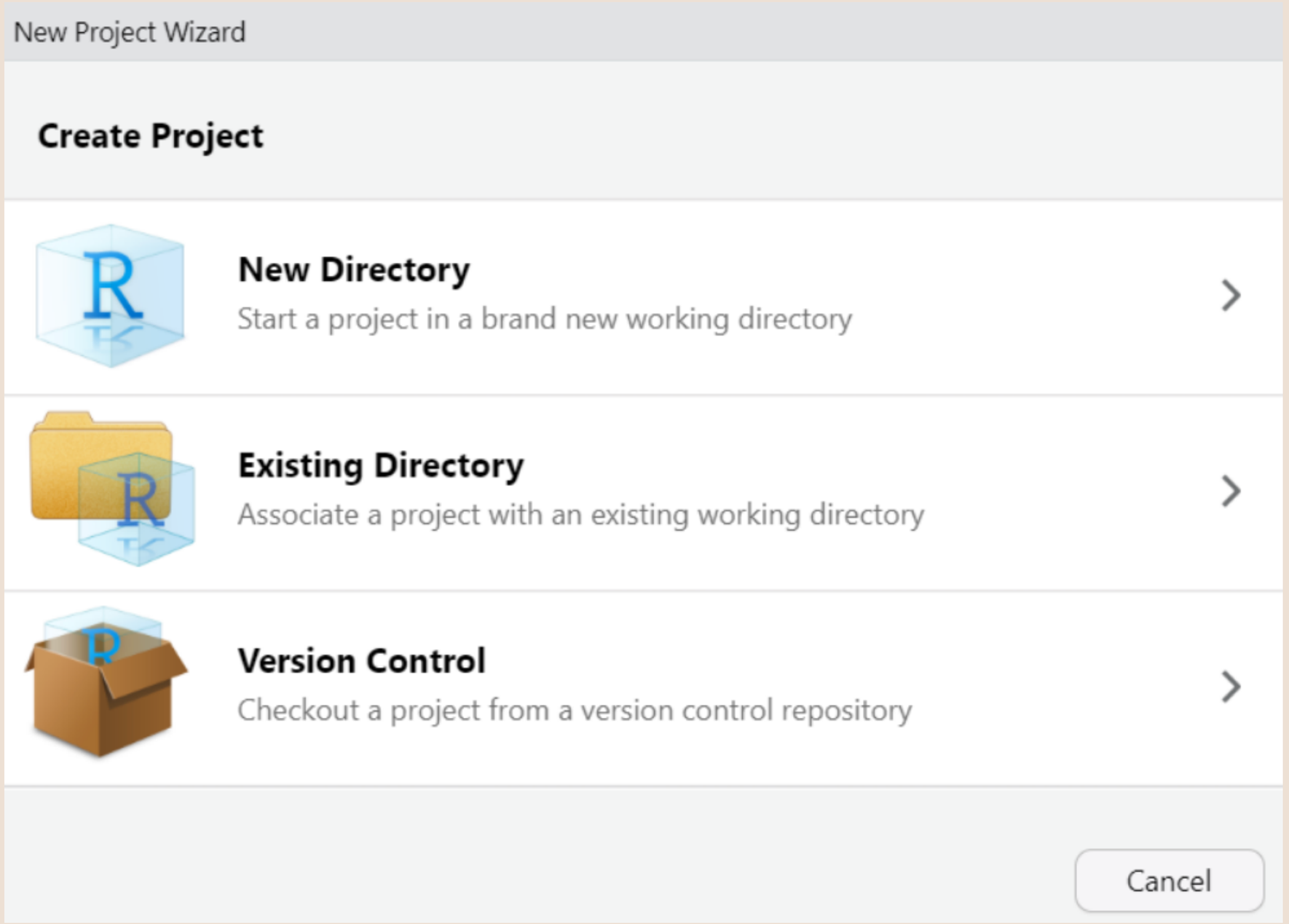
Demonstration

Setting up a new project in Rstudio

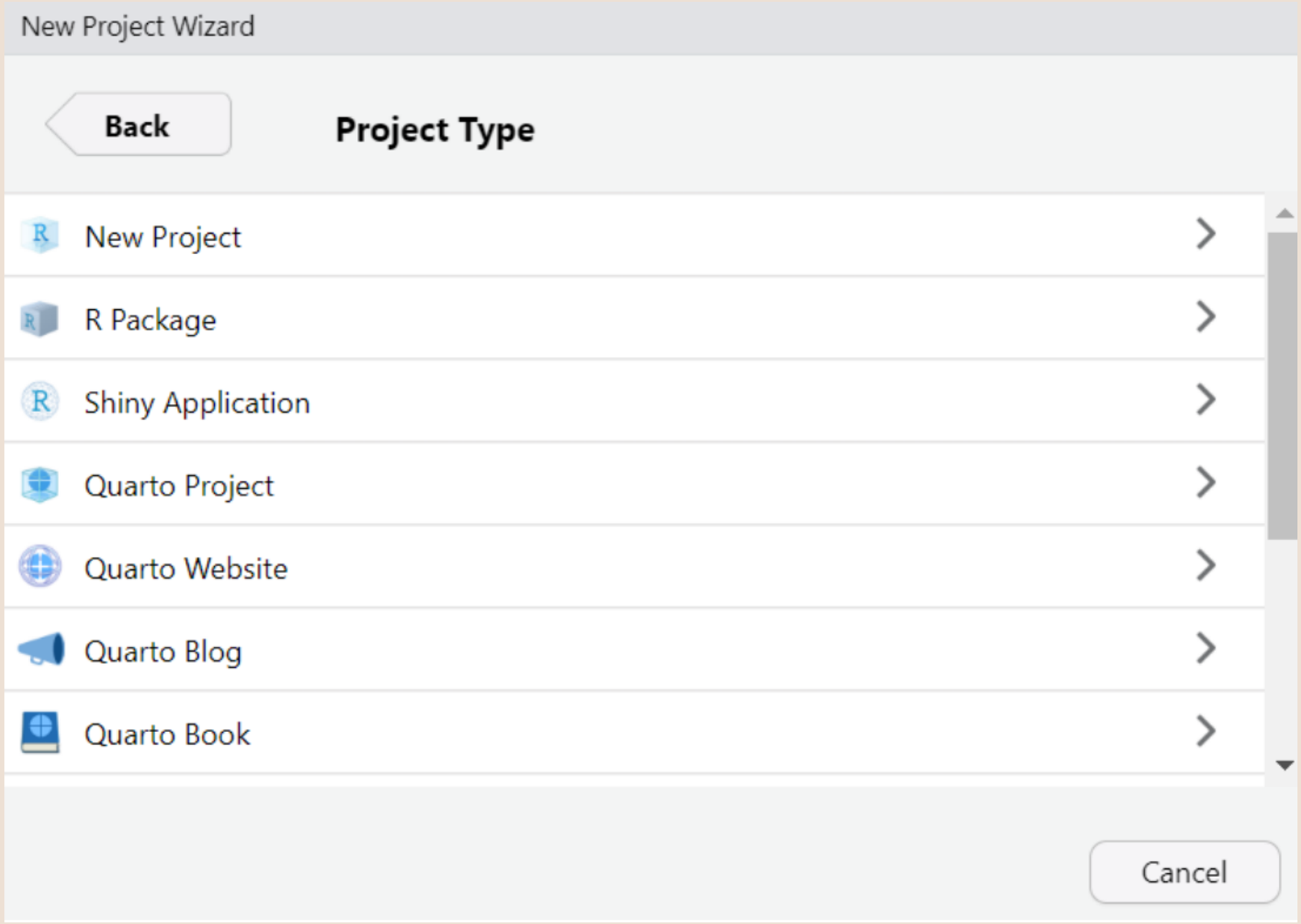
Step 1- Select **File** > **New Project**.. from the menu bar.



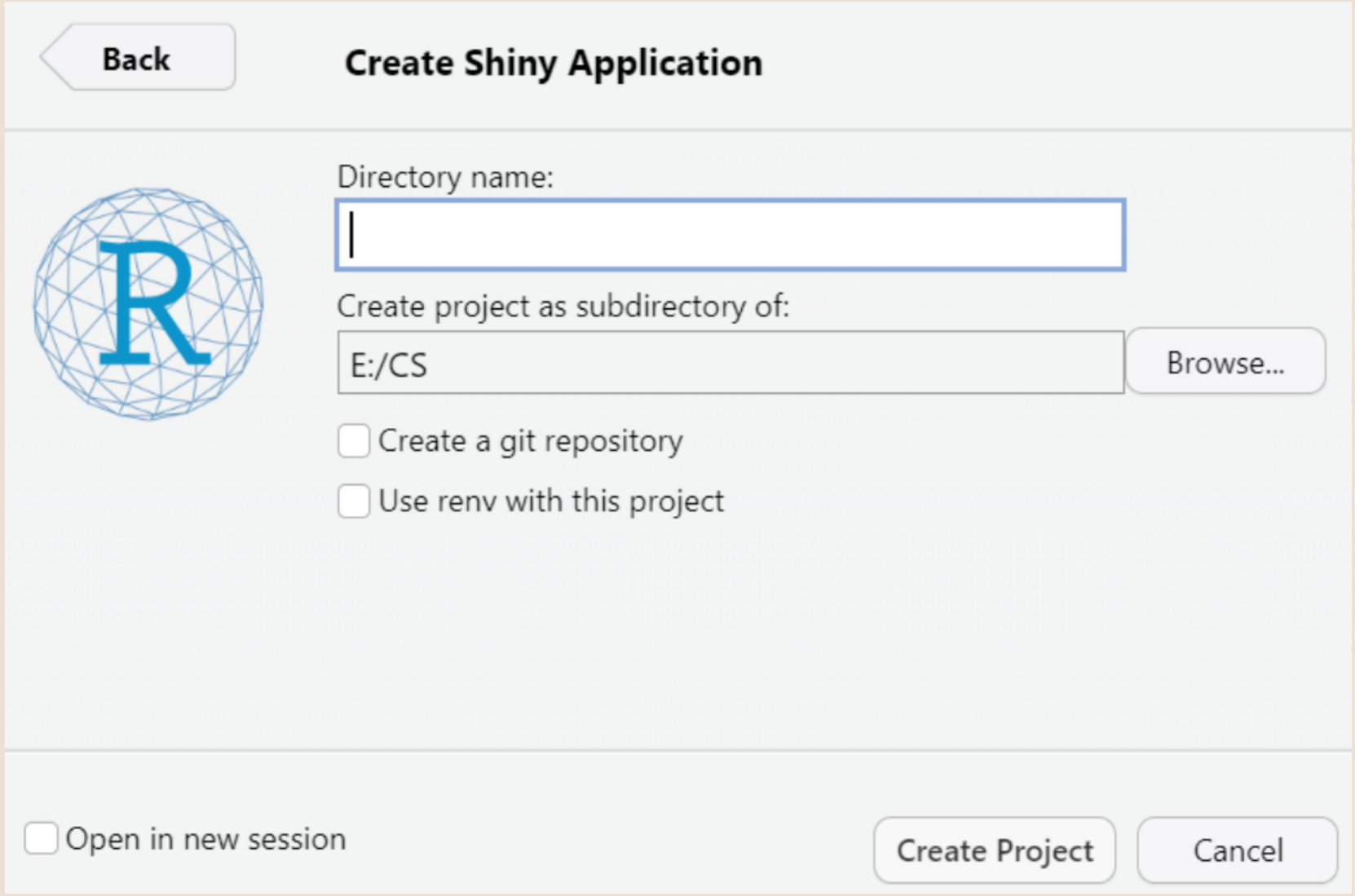
Step 2- Select **New Directory** from the pop-up window



Step 3- Select Shiny Application



Step 4- Pick a name for your Project Directory, and select a subdirectory to store your project.



Step 5- Tick the **open in new session**, and click **Create Project** to open your R project in a new R studio window.

Creating an App with R Shiny

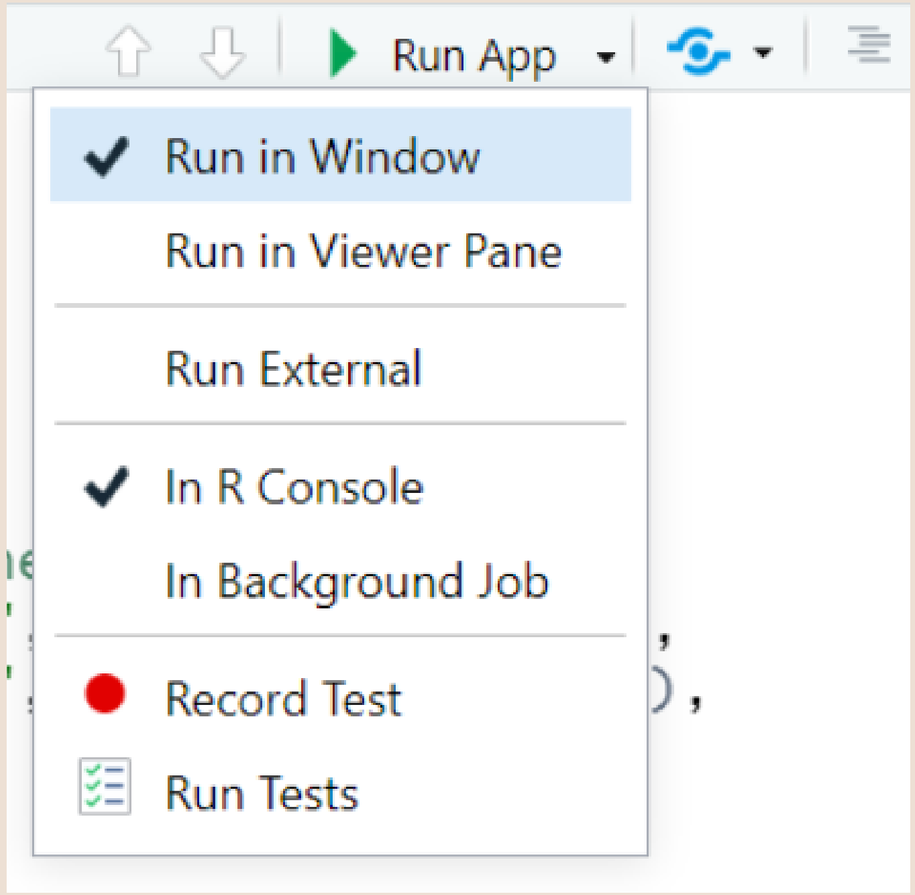
A shiny web applications requires a computer/server with **R**

1. Basic R script for structure of a shiny app:

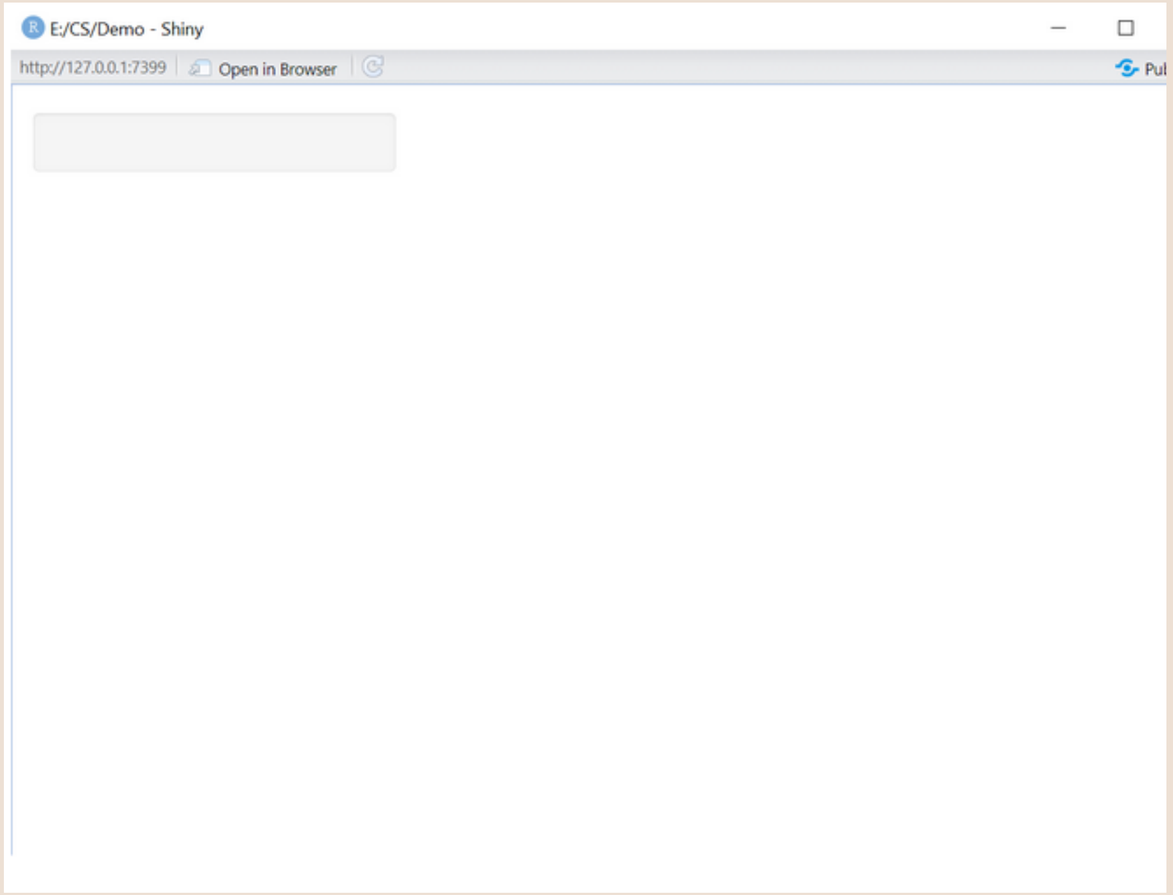
Run App

```r
# Importing Packages
library(shiny)
library(ggplot2)
library(dplyr)
library(agridat)

# defining the ui
ui <- fluidPage(
  titlePanel(""),  # Add a title panel
  sidebarLayout(  # Make the a sidebar
    sidebarPanel(),  # add a sidepanel inside a sidebar
    mainPanel()  # add a main panel
  )
)

# defining server
server <- function(input, output) {}

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

2. To run the app, click **Run App** at the top right of the script window.



3. Output after running the script from the previous slide.



4. Without the connection to the server and app, UI is just HTML content

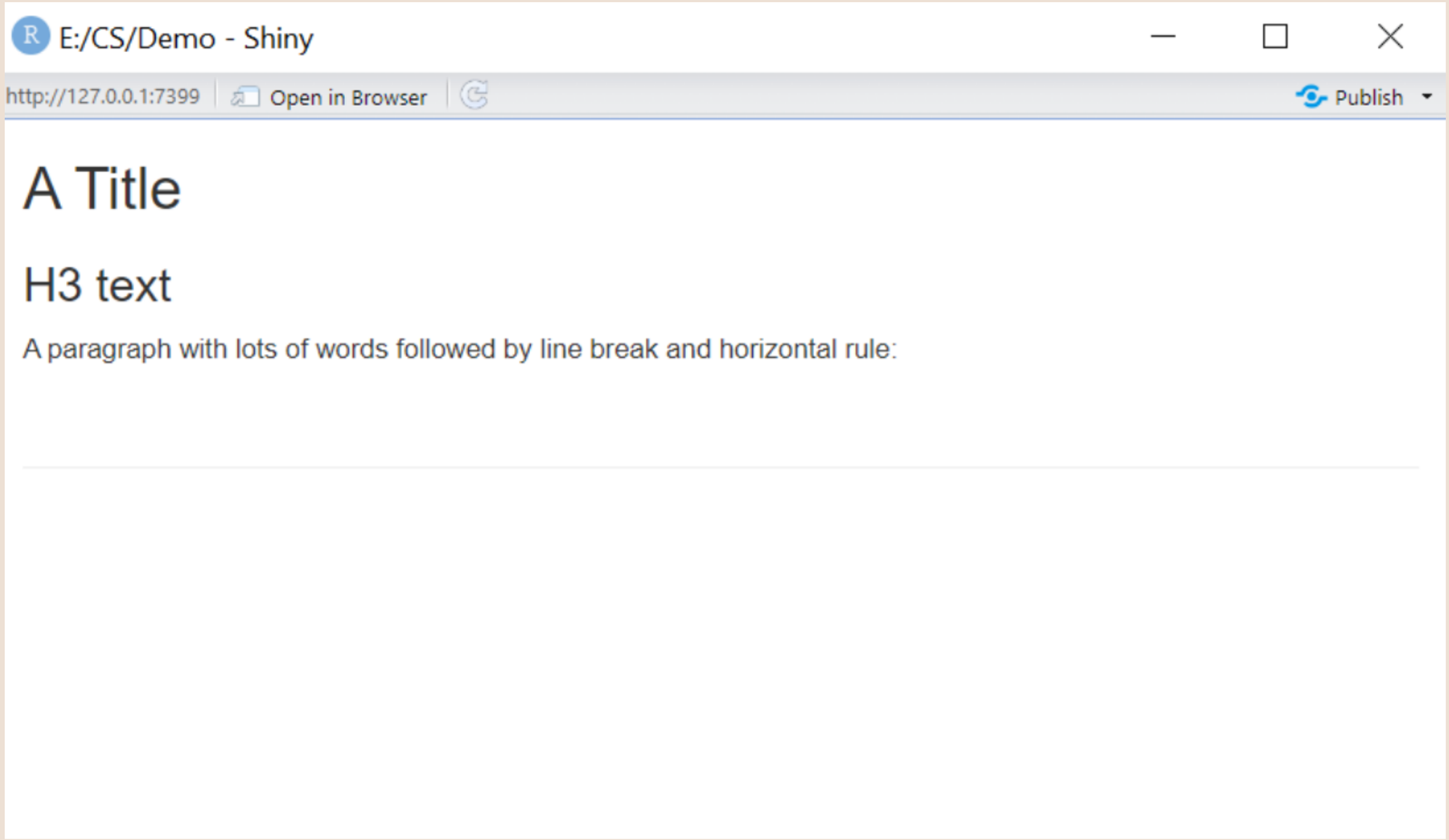
```
> fluidPage(
+     titlePanel(""),  # Add a title panel
+     sidebarLayout(  # Make the a sidebar
+         sidebarPanel(),  # add a sidepanel inside a sidebar
+         mainPanel()  # add a main panel
+     )
+ )
<div class="container-fluid">
  <h2></h2>
  <div class="row">
    <div class="col-sm-4">
      <form class="well" role="complementary"></form>
    </div>
    <div class="col-sm-8" role="main"></div>
  </div>
</div>
>
```

5. We can add HTML tags in the ui.

Run App

```
# Importing Packages
library(shiny)
library(ggplot2)
library(dplyr)
library(agridat)

# defining the ui
ui <- fluidPage(
      titlePanel("A Title"),
      h3("H3 text"),
      p("A paragraph with lots of words followed by line break and horizontal rule:"),
      br(),
      hr(),

)

# defining server
server <- function(input, output) {}

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

6. The very basic shiny app you'll see when you run the code in previous slide



7. R script and Error on running the app without defining the server

```r
# importing the library
library(shiny)

# defining the UI
ui <- fluidPage(
  sliderInput(inputId = "num", label = "Choose a number",
              value = 25, min = 1, max = 100),
  plotOutput("hist")
)

# running the application
shinyApp(ui = ui)

```

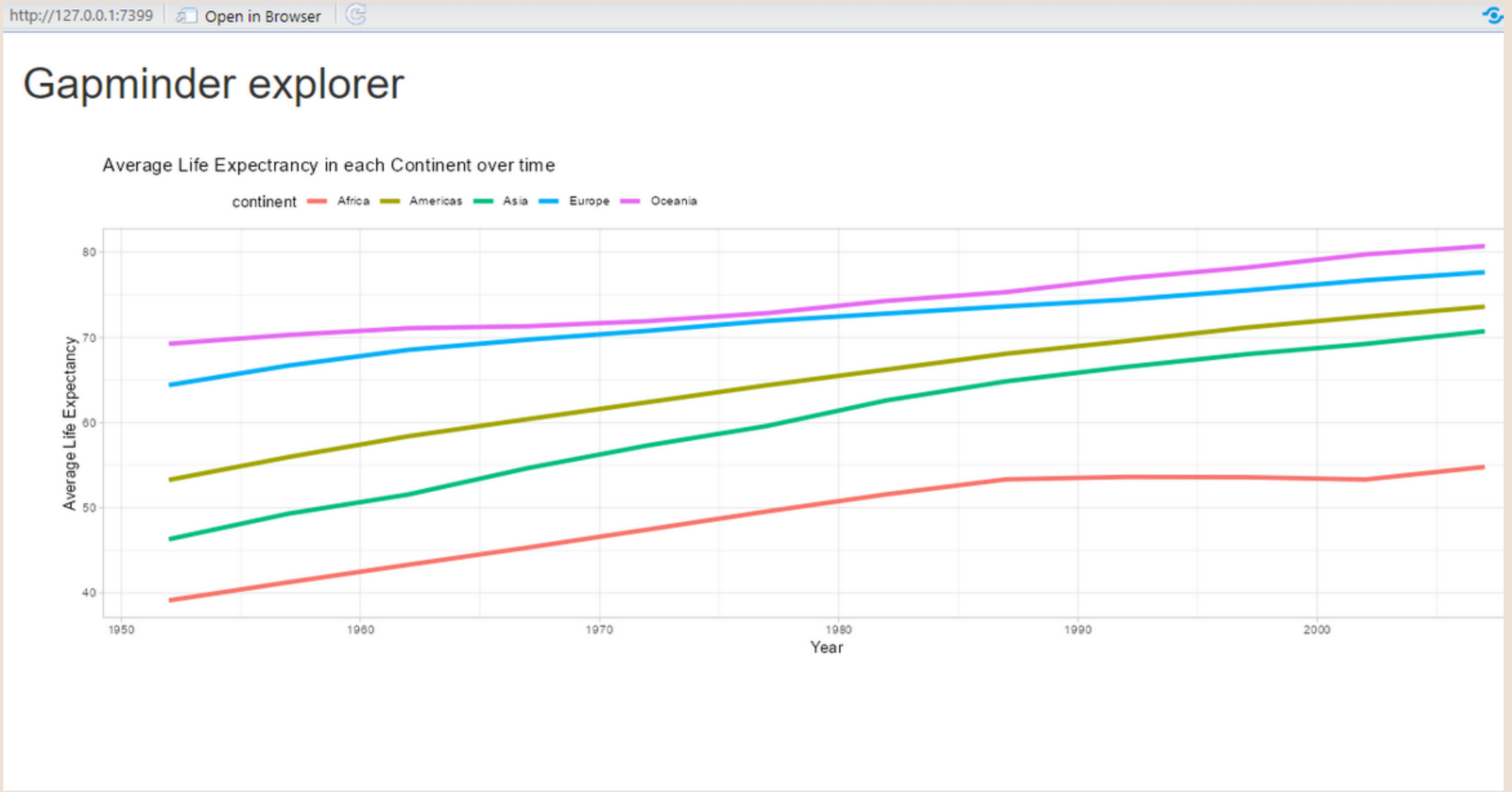
```
> runApp('app2.R')
Error in shinyApp(ui = ui) :
  argument "server" is missing, with no default
>
```

8. Script for Plotting a Static Line Graph.

Reload App

```
1  library(gapminder)
2  library(ggplot2)
3  library(dplyr)
4  library(shiny)
5
6  ui <- fluidPage(
7    titlePanel("Gapminder explorer"),
8    plotOutput("line")
9  )
10
11 server <- function(input, output) {
12   output$line <- renderPlot({
13     data <- gapminder %>%
14       group_by(year, continent) %>%
15       summarise(avgLifeExp = mean(lifeExp))
16
17     ggplot(data, aes(x = year, y = avgLifeExp, color = continent)) +
18       geom_line(size = 1.5) +
19       ggtitle("Average Life Expectrancy in each Continent over time") +
20       labs(x = "Year", y = "Average Life Expectancy") +
21       theme_light() +
22       theme(
23         plot.margin = unit(c(2, 1, 1, 1), "cm"),
24         plot.title = element_text(vjust = 13),
25         legend.position = c(0.25, 1.07), legend.direction = "horizontal"
26       )
27   })
28 }
29
30 shinyApp(ui = ui, server = server)
```

9. Visuals of graph plotted by script from the previous page.



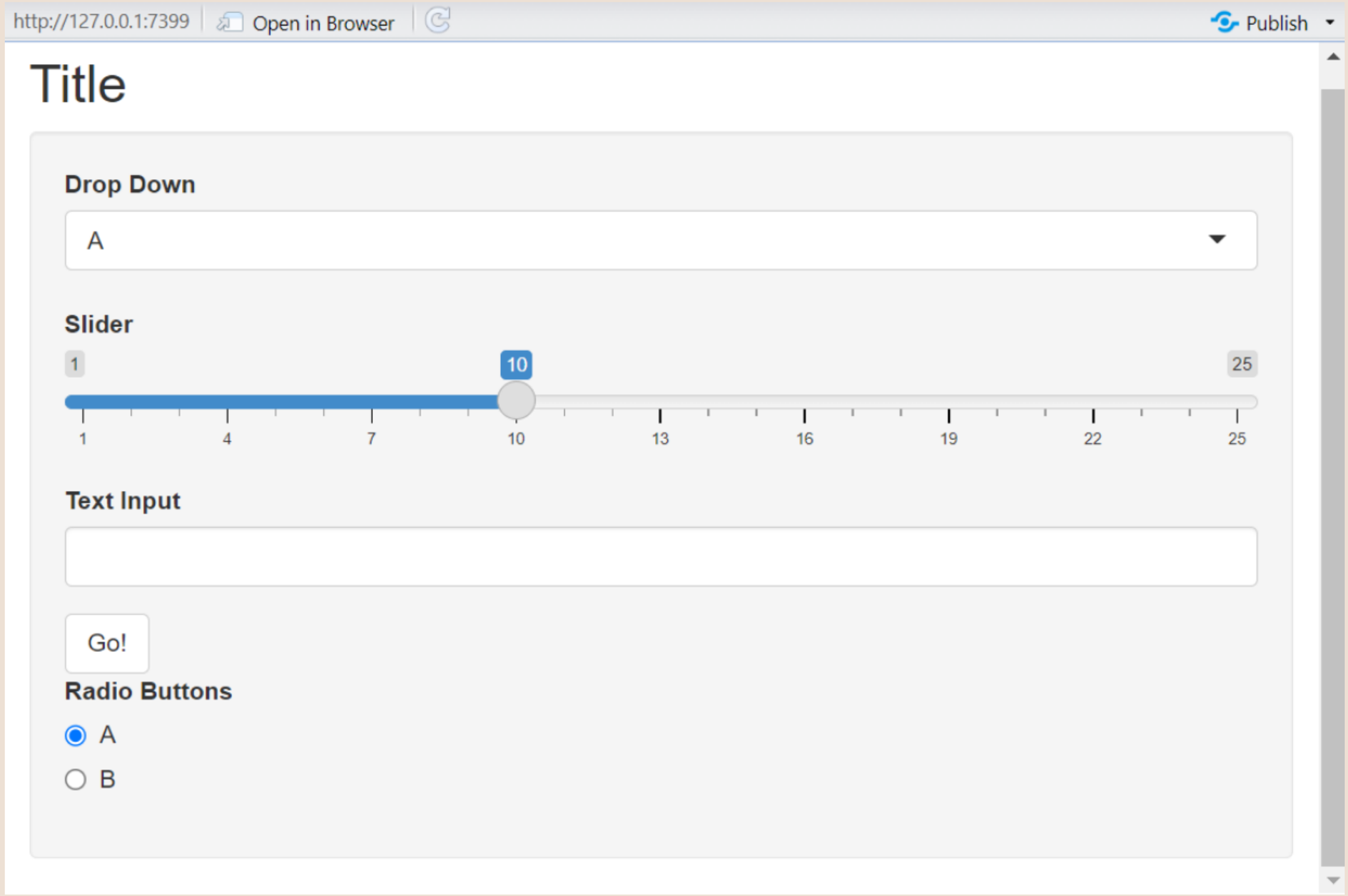
Some basic interactive visualizations using R Shiny

Example 1. R script showing how to take inputs for visualizing data.

Run App

```
1  library(shiny)
2
3  ui <- fluidPage(
4    titlePanel("Title"),
5    sidebarLayout(
6      sidebarPanel(
7        selectInput(inputId = "one",
8                    label = "Drop Down",  # Give the input a label
9                    choices = c("A" = "a","B" = "b","C" = "c","D" = "d",
10                               "E" = "e","F" = "f","G" = "g","H" = "h"),
11                   selected = "a"),
12       sliderInput(inputId = "two",
13                   label = "Slider",
14                   min=1, max=25, value= c(10)),
15       textInput(inputId = "text",
16                  label = "Text Input", ""),
17       actionButton(inputId = "action", label = "Go!"),
18       radioButtons(inputId = "radio", label = "Radio Buttons",
19                    choices = c("A", "B")),
20
21     ),
22     mainPanel()
23   )
24 )
25
26 server <- function(input, output) {}
27 shinyApp(ui = ui, server = server)
28
```

Example 1. Output Visual for the R script from the previous slide.

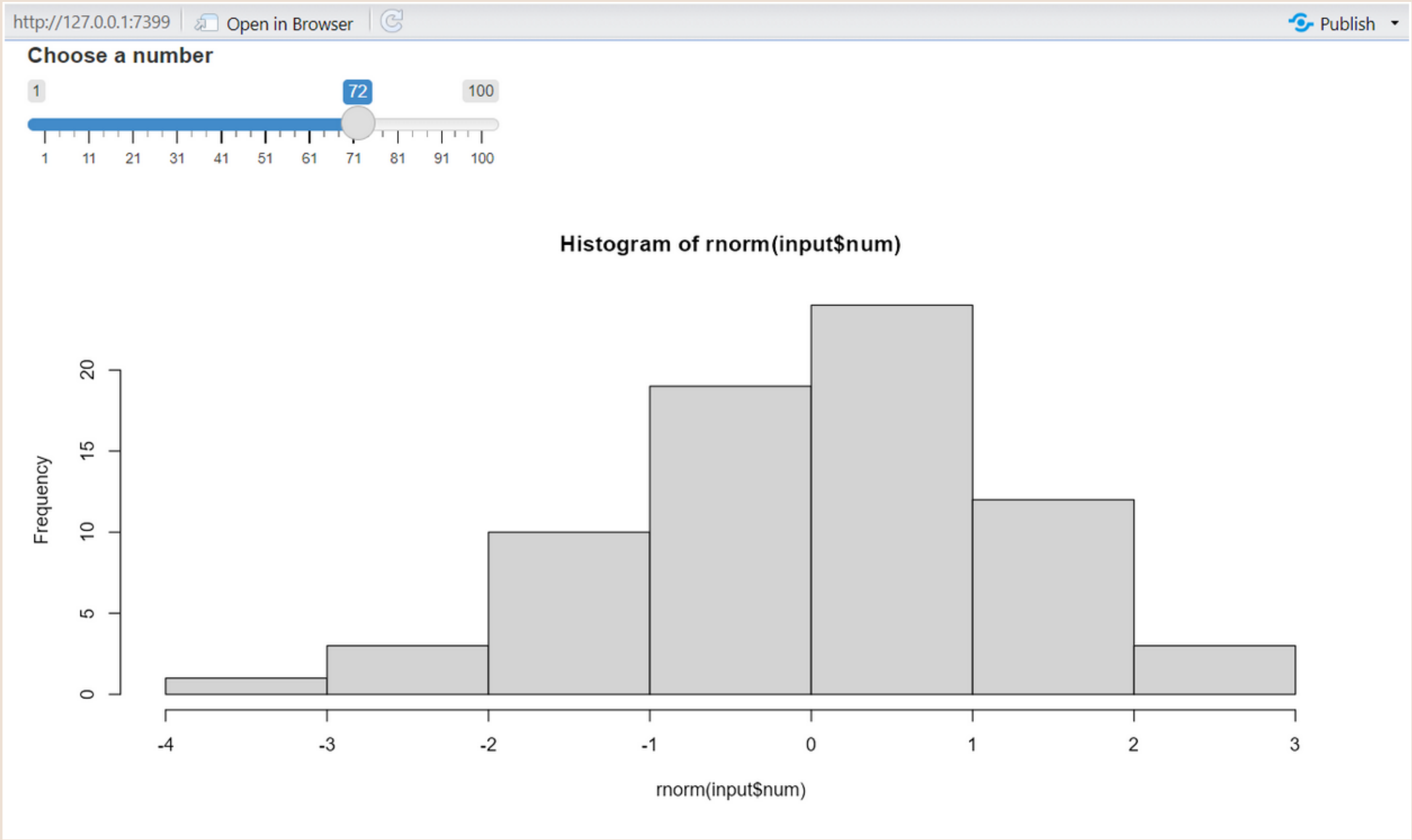


Example 2. R script to create a simple normal distribution histogram.

Reload App

```
1  # importing the library
2  library(shiny)
3
4  # defining the UI
5  ui <- fluidPage(
6    sliderInput(inputId = "num", label = "Choose a number",
7                value = 25, min = 1, max = 100),
8    plotOutput("hist")
9  )
10
11 # defining server
12 server <- function(input, output) {
13   output$hist <- renderPlot({
14     hist(rnorm(input$num))  #n random samples distr. with mean=0 sd=1
15   })
16 }
17
18 # running the application
19 shinyApp(ui = ui, server = server)
20
21
```

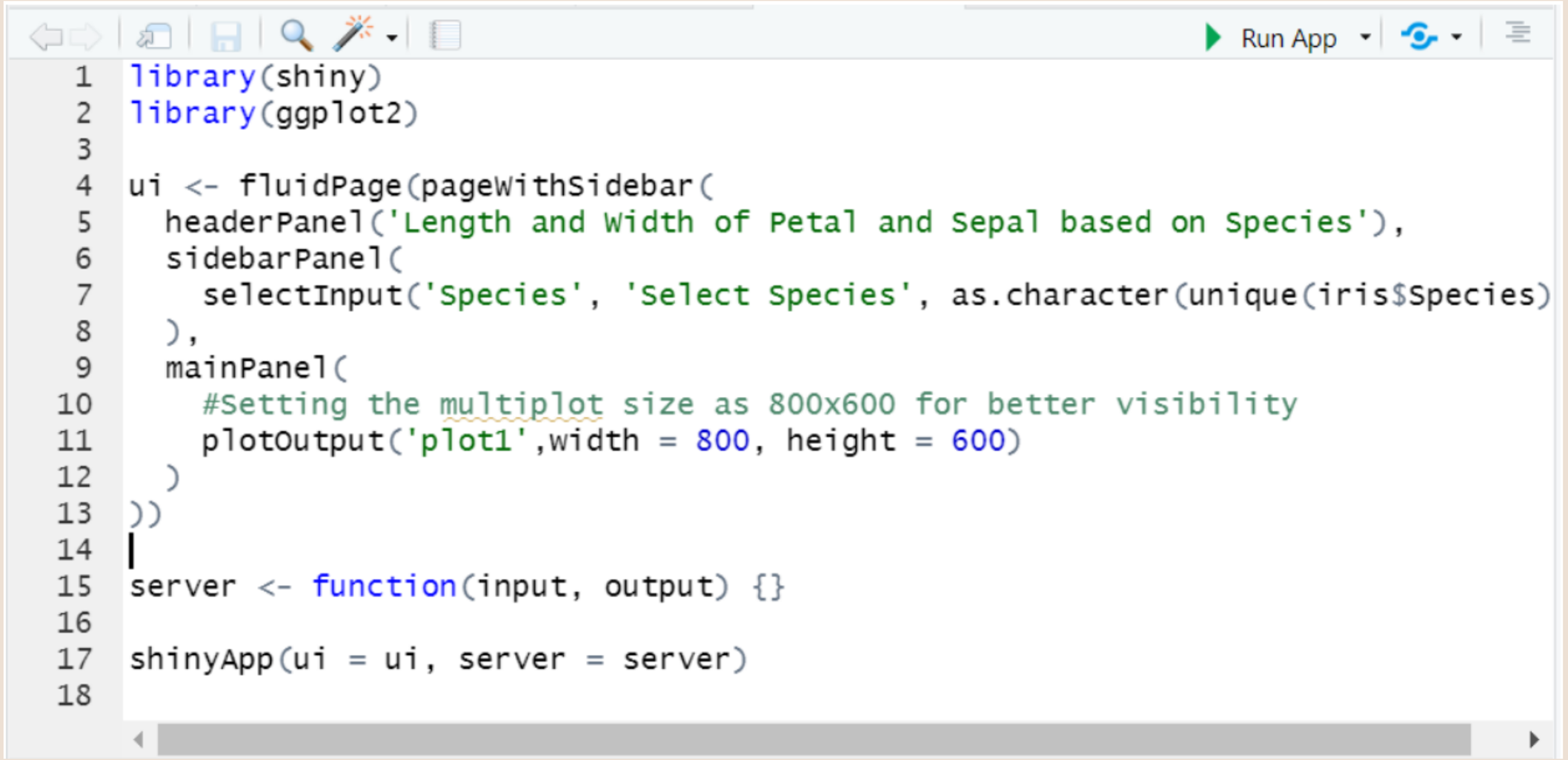
Example 2. Snapshot of Resulting visualization for the script in the previous slide.



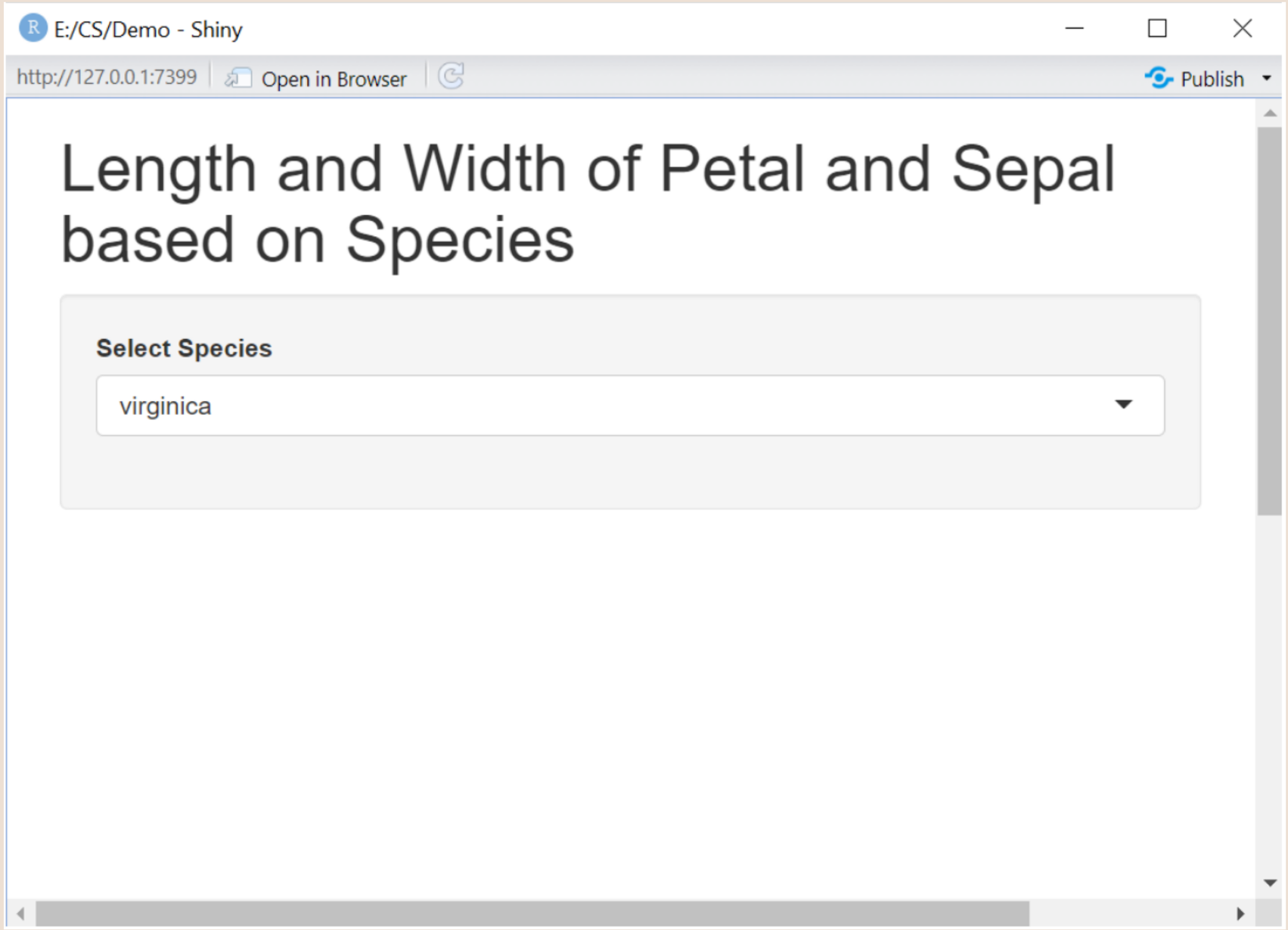
Visualizing a Dashboard with R Shiny: Iris Dataset

- The dataset used for this project is IRIS which is available in R.

- Creating the UI:



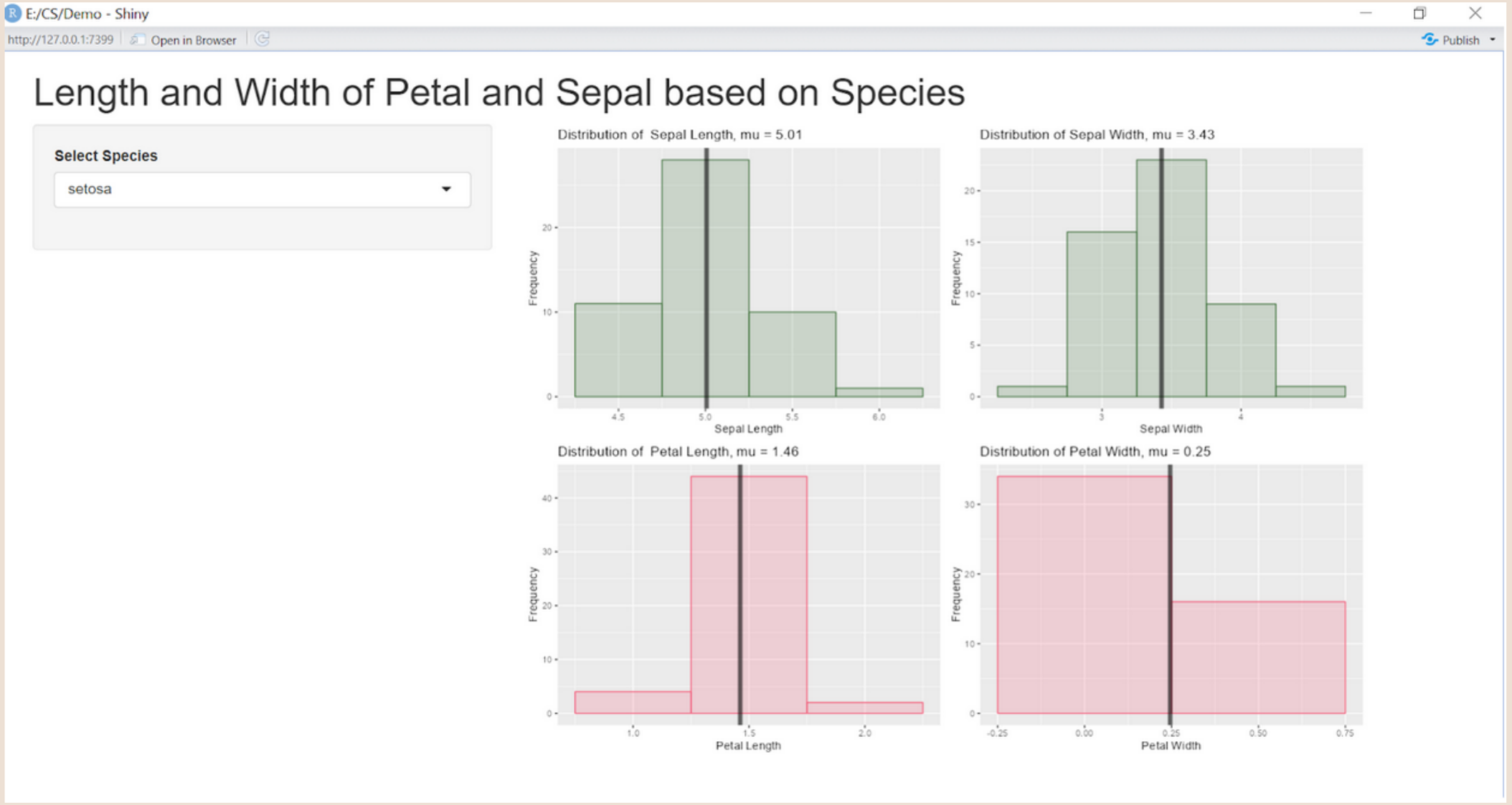
- Resulting Visual of Step 1.



- Creating the Server Logic (visualizations are made in ggplot):

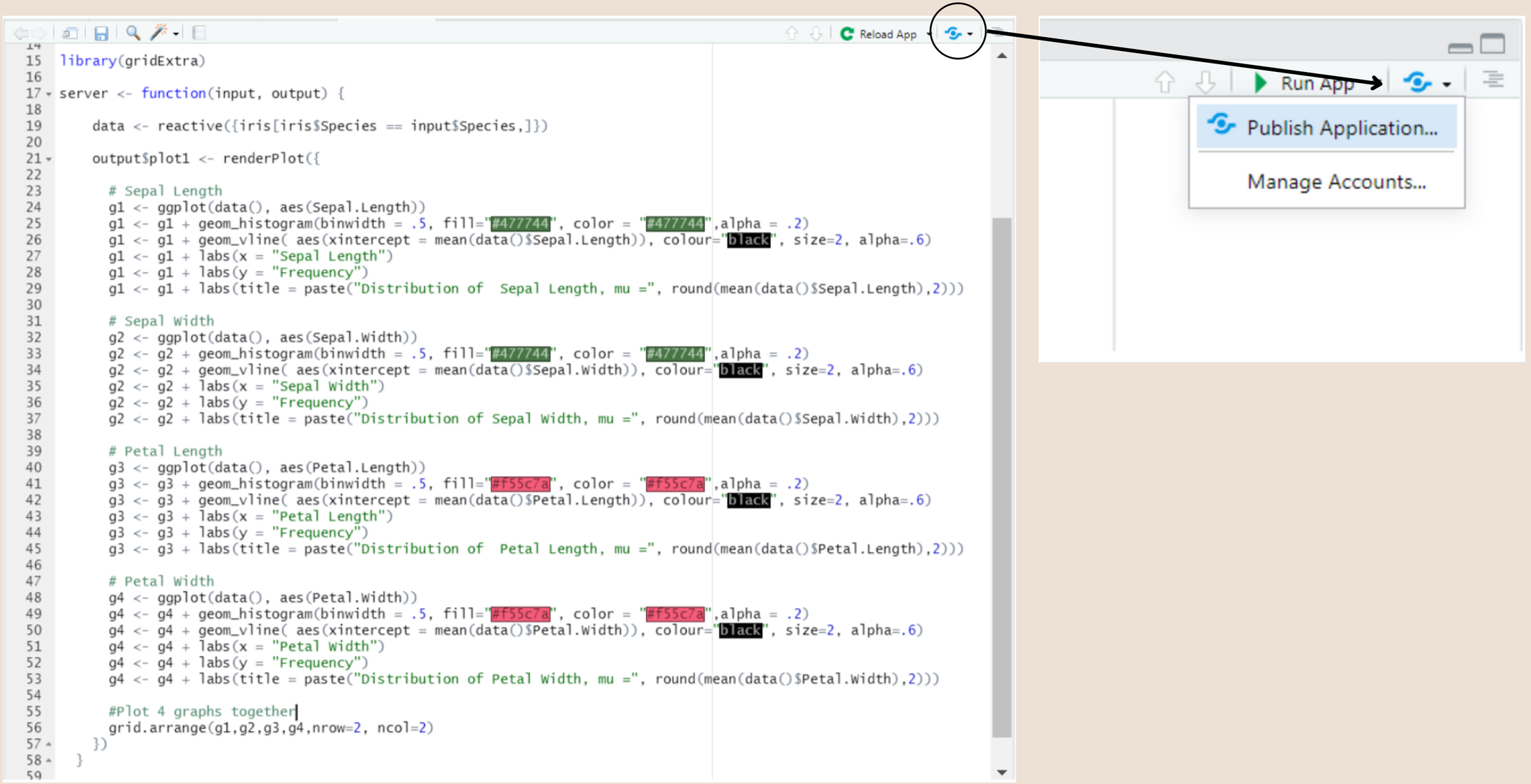
```
library(gridExtra)

server <- function(input, output) {

    data <- reactive({iris[iris$Species == input$Species,]})

    output$plot1 <- renderPlot({

      # Sepal Length
      g1 <- ggplot(data(), aes(Sepal.Length))
      g1 <- g1 + geom_histogram(binwidth = .5, fill="#477744", color = "#477744",alpha = .2)
      g1 <- g1 + geom_vline( aes(xintercept = mean(data()$Sepal.Length)), colour="black", size=2, alpha=.6)
      g1 <- g1 + labs(x = "Sepal Length")
      g1 <- g1 + labs(y = "Frequency")
      g1 <- g1 + labs(title = paste("Distribution of  Sepal Length, mu =", round(mean(data()$Sepal.Length),2)))

      # Sepal Width
      g2 <- ggplot(data(), aes(Sepal.Width))
      g2 <- g2 + geom_histogram(binwidth = .5, fill="#477744", color = "#477744",alpha = .2)
      g2 <- g2 + geom_vline( aes(xintercept = mean(data()$Sepal.Width)), colour="black", size=2, alpha=.6)
      g2 <- g2 + labs(x = "Sepal Width")
      g2 <- g2 + labs(y = "Frequency")
      g2 <- g2 + labs(title = paste("Distribution of Sepal Width, mu =", round(mean(data()$Sepal.Width),2)))

      # Petal Length
      g3 <- ggplot(data(), aes(Petal.Length))
      g3 <- g3 + geom_histogram(binwidth = .5, fill="#f55c7a", color = "#f55c7a",alpha = .2)
      g3 <- g3 + geom_vline( aes(xintercept = mean(data()$Petal.Length)), colour="black", size=2, alpha=.6)
      g3 <- g3 + labs(x = "Petal Length")
      g3 <- g3 + labs(y = "Frequency")
      g3 <- g3 + labs(title = paste("Distribution of  Petal Length, mu =", round(mean(data()$Petal.Length),2)))

      # Petal Width
      g4 <- ggplot(data(), aes(Petal.Width))
      g4 <- g4 + geom_histogram(binwidth = .5, fill="#f55c7a", color = "#f55c7a",alpha = .2)
      g4 <- g4 + geom_vline( aes(xintercept = mean(data()$Petal.Width)), colour="black", size=2, alpha=.6)
      g4 <- g4 + labs(x = "Petal Width")
      g4 <- g4 + labs(y = "Frequency")
      g4 <- g4 + labs(title = paste("Distribution of Petal Width, mu =", round(mean(data()$Petal.Width),2)))

      #Plot 4 graphs together
      grid.arrange(g1,g2,g3,g4,nrow=2, ncol=2)
    })
  }
```

- Final Visualization



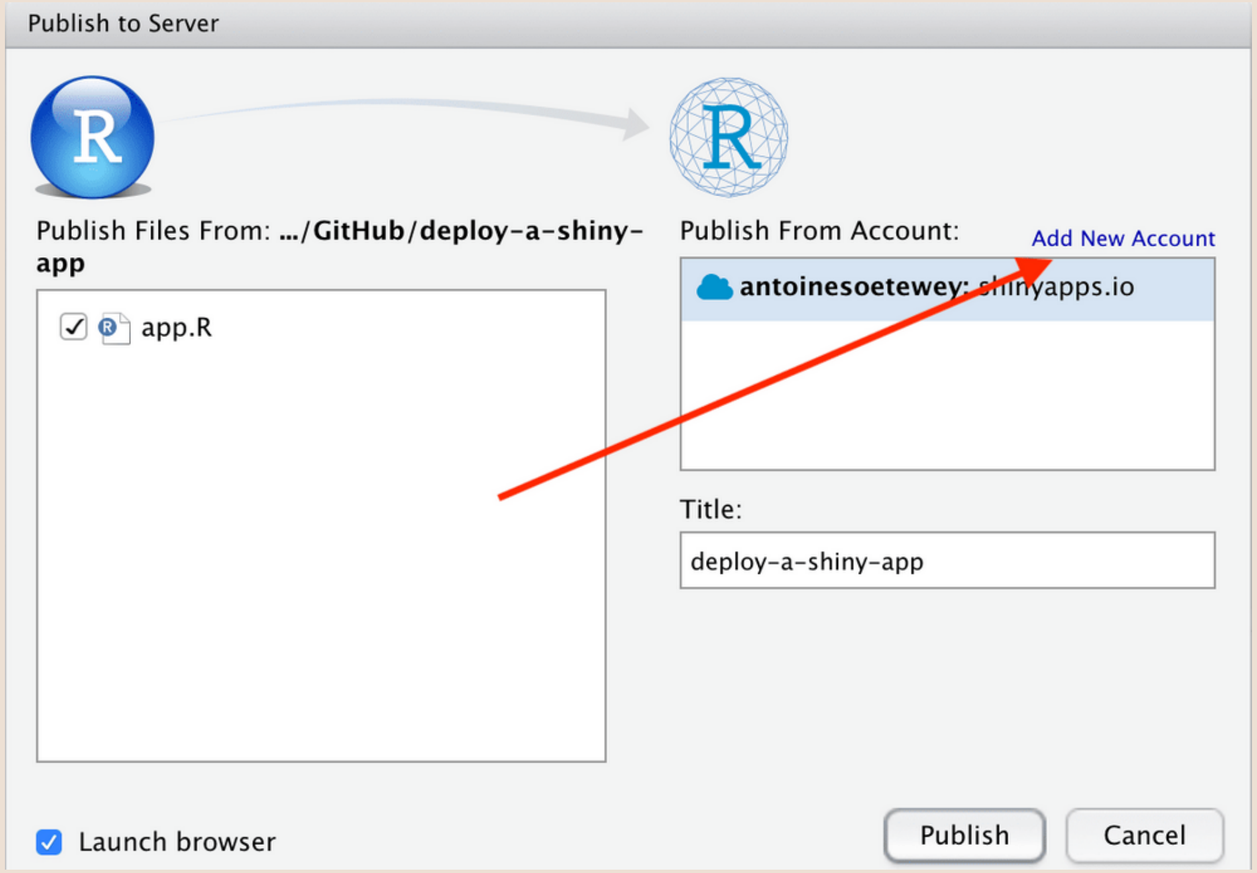
Publishing an R Shiny App

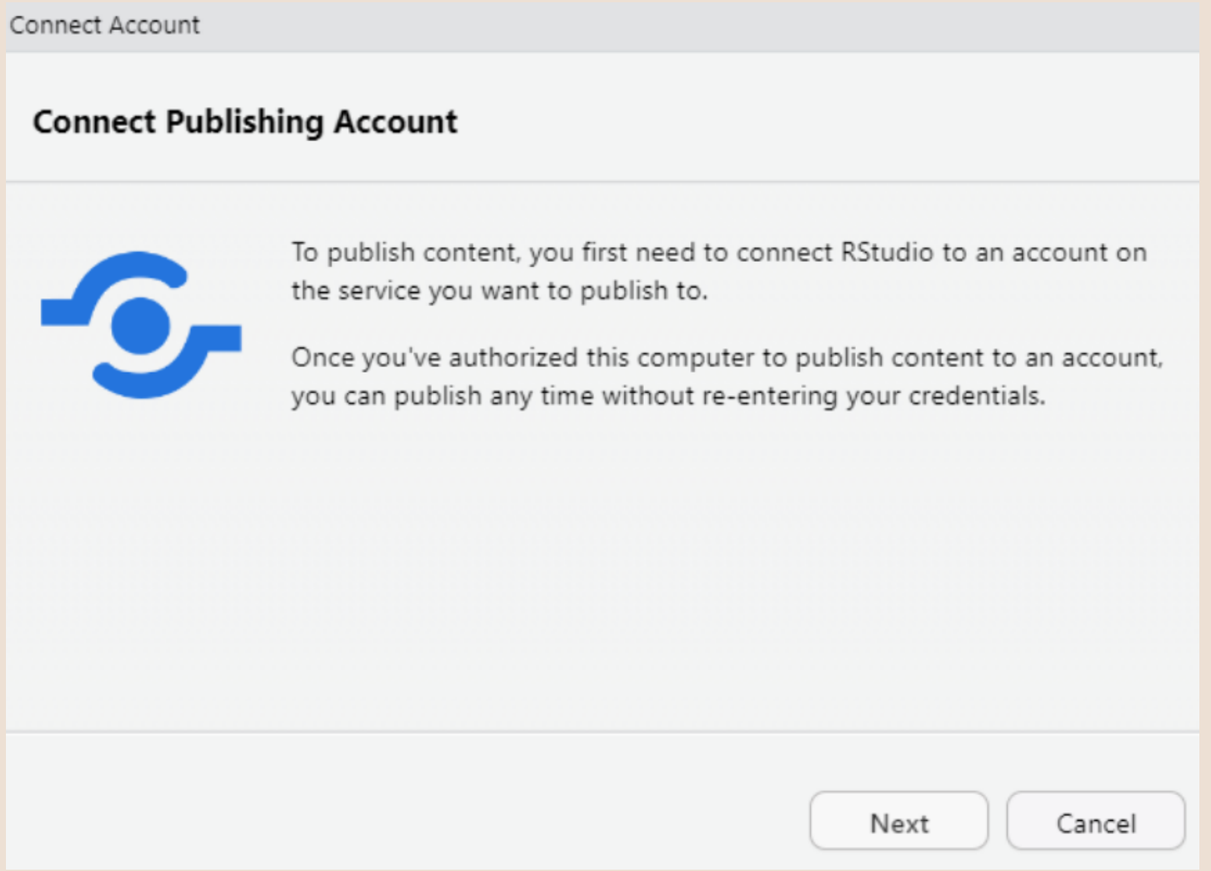
Step 1- Click the button shown in the figure below in the right corner of the window



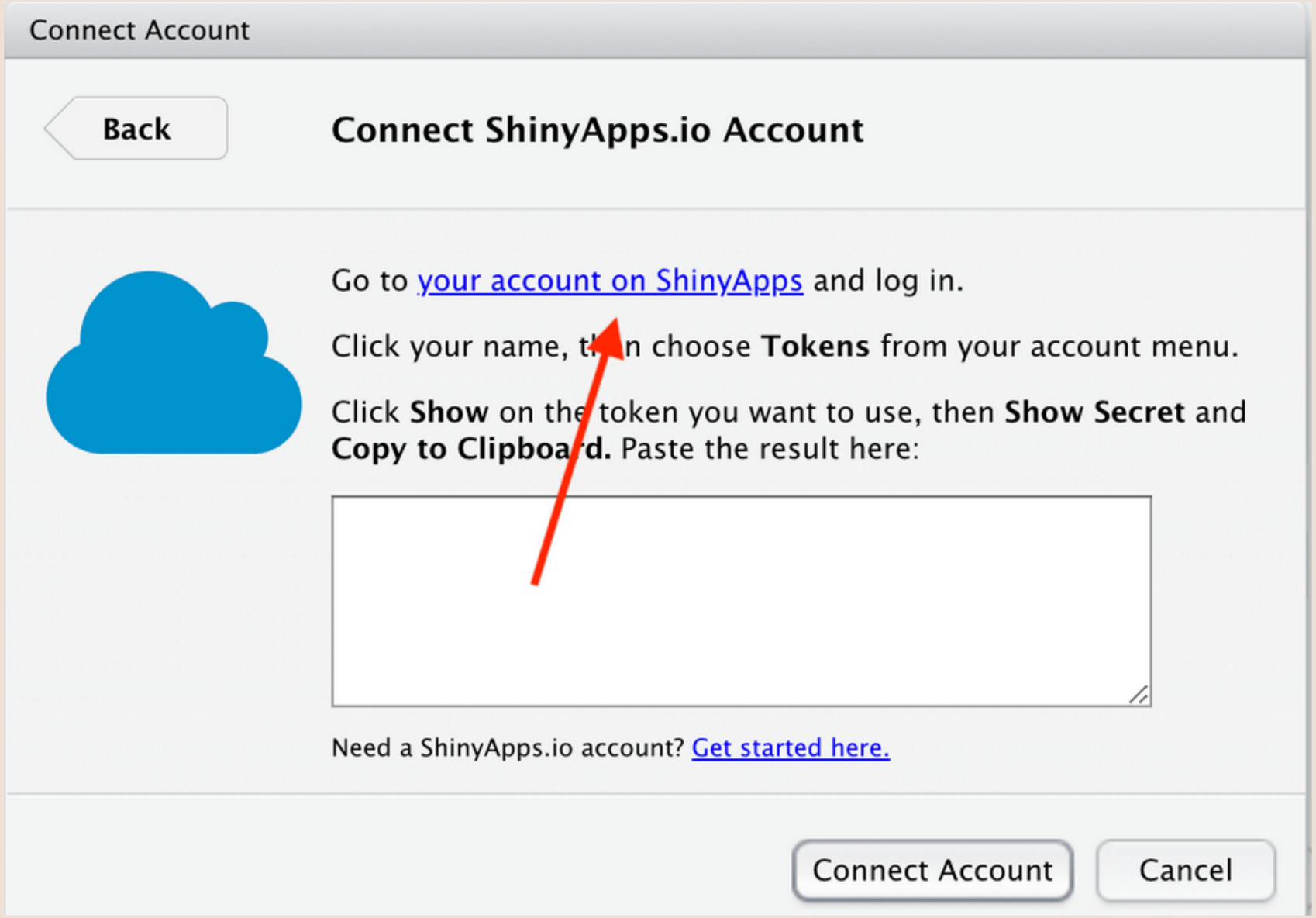

```
library(gridExtra)

server <- function(input, output) {

    data <- reactive({iris[iris$Species == input$Species,]})

    output$plot1 <- renderPlot({

      # Sepal Length
      g1 <- ggplot(data(), aes(Sepal.Length))
      g1 <- g1 + geom_histogram(binwidth = .5, fill="#477744", color = "#477744",alpha = .2)
      g1 <- g1 + geom_vline( aes(xintercept = mean(data()$Sepal.Length)), colour="black", size=2, alpha=.6)
      g1 <- g1 + labs(x = "Sepal Length")
      g1 <- g1 + labs(y = "Frequency")
      g1 <- g1 + labs(title = paste("Distribution of  Sepal Length, mu =", round(mean(data()$Sepal.Length),2)))

      # Sepal Width
      g2 <- ggplot(data(), aes(Sepal.Width))
      g2 <- g2 + geom_histogram(binwidth = .5, fill="#477744", color = "#477744",alpha = .2)
      g2 <- g2 + geom_vline( aes(xintercept = mean(data()$Sepal.Width)), colour="black", size=2, alpha=.6)
      g2 <- g2 + labs(x = "Sepal Width")
      g2 <- g2 + labs(y = "Frequency")
      g2 <- g2 + labs(title = paste("Distribution of Sepal Width, mu =", round(mean(data()$Sepal.Width),2)))

      # Petal Length
      g3 <- ggplot(data(), aes(Petal.Length))
      g3 <- g3 + geom_histogram(binwidth = .5, fill="#f55c7a", color = "#f55c7a",alpha = .2)
      g3 <- g3 + geom_vline( aes(xintercept = mean(data()$Petal.Length)), colour="black", size=2, alpha=.6)
      g3 <- g3 + labs(x = "Petal Length")
      g3 <- g3 + labs(y = "Frequency")
      g3 <- g3 + labs(title = paste("Distribution of  Petal Length, mu =", round(mean(data()$Petal.Length),2)))

      # Petal Width
      g4 <- ggplot(data(), aes(Petal.Width))
      g4 <- g4 + geom_histogram(binwidth = .5, fill="#f55c7a", color = "#f55c7a",alpha = .2)
      g4 <- g4 + geom_vline( aes(xintercept = mean(data()$Petal.Width)), colour="black", size=2, alpha=.6)
      g4 <- g4 + labs(x = "Petal Width")
      g4 <- g4 + labs(y = "Frequency")
      g4 <- g4 + labs(title = paste("Distribution of Petal Width, mu =", round(mean(data()$Petal.Width),2)))

      #Plot 4 graphs together
      grid.arrange(g1,g2,g3,g4,nrow=2, ncol=2)
    })
  }
```

Step 2- If it is your first Shiny app, the box “Publish From Account” should be empty. Click on **“Add New Account”** to link the shinyapps.io account you just created, else connect to your existing Publishing Account.

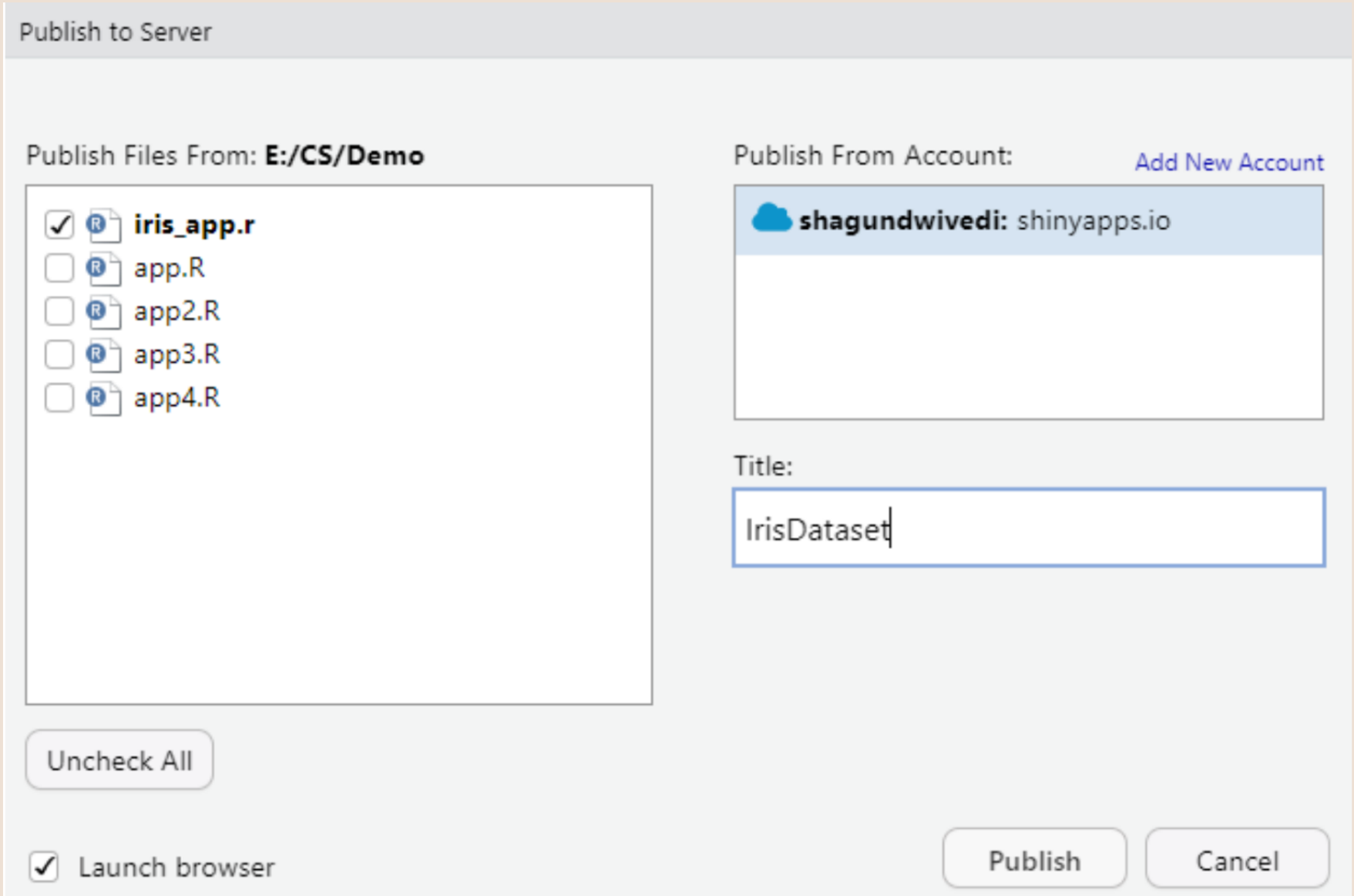




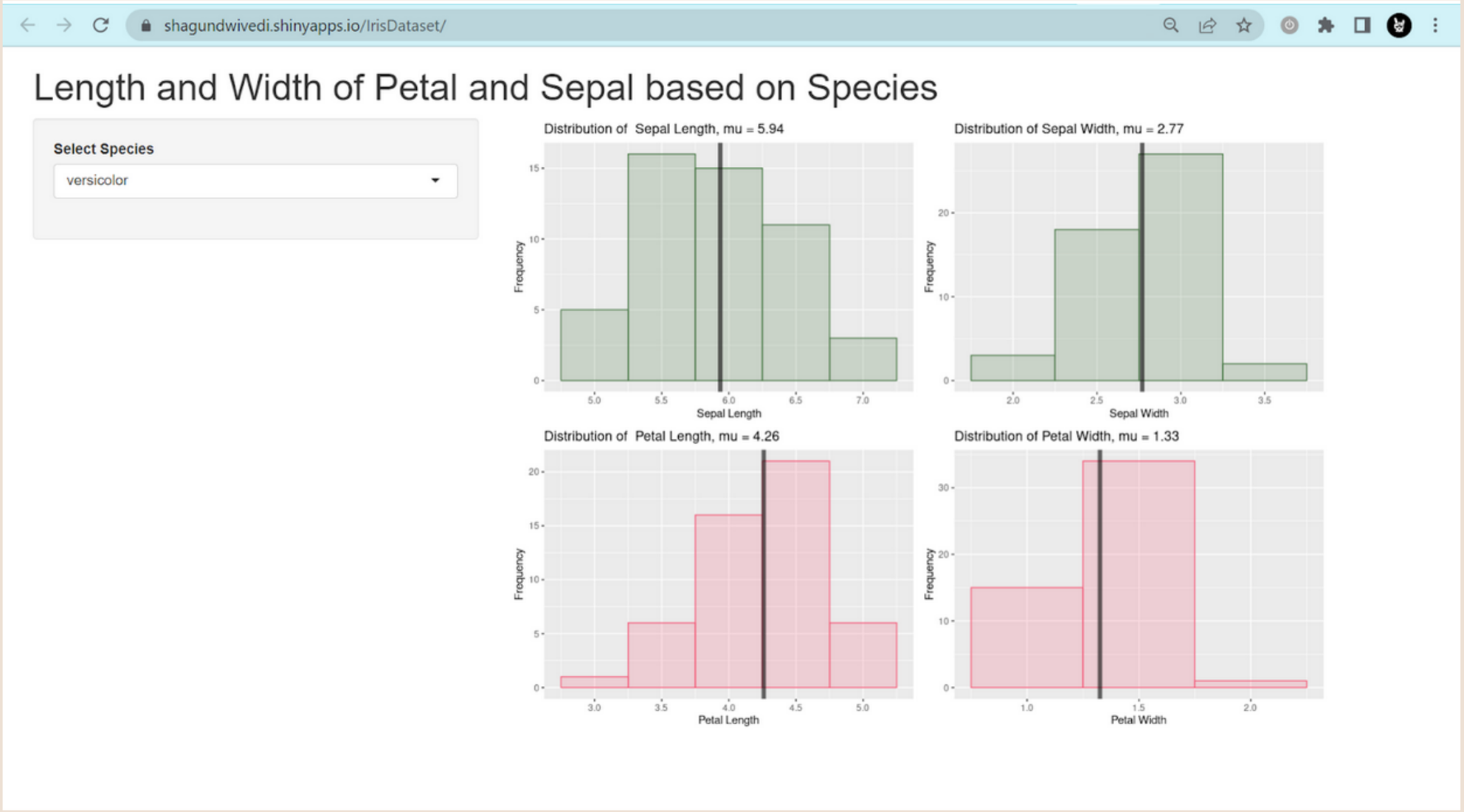
Step-3. Click on the link to your ShinyApps account:



Step-4. Choose a title (without space) and click on the Publish button:



Step-5. After a while, we get redirected to the shinyapps.io webpage:



Personal insights about R Shiny Strengths and Weaknesses

Insights

- Relatively easy to learn if you know R
- Results can be published in a straightforward manner
- Using RStudio makes work smoother

PROS (Being Code Based):

- highly customizable
- good for creating dashboards
- can use any R library like ggplot, tidyverse to create visualizations
- can carry out data manipulation if needed

CONS (Being Code Based):

- requires knowledge of the R language
- more time and effort taking than UI-based tools like Tableau
- functionalities not present in R packages are hard to implement
- user needs to understand UI and server logic and how reactivity applies to the app

How can you learn R Shiny?

A list of resources for you to get started with R Shiny!

- https://lrouviere.github.io/VISU/pres_shiny.pdf
- https://cl.indiana.edu/~obscrivn/Shiny-Introduction.pdf
- https://deanattali.com/blog/building-shiny-apps-tutorial/
- https://ibiostat.be/seminar/uploads/introdcution-r-shiny-package-20160330.pdf

Contributors: Group 6

- Shagun Dwivedi
- Deeksha Chutani
- Param Chordiya
- Shaifali Vashistha

Thank You