Shiny Basics

Estimated time needed: **30** minutes

Objectives

After completing the lab you will be able to:

- Create a simple Shiny application
- Add title panel and slider input components
- Add a dynamic histogram

Dataset Used

You will use the mtcars (motor trend car road tests) dataset, which is built into R.

RStudio with Watson Studio

This lab will use Watson Studio to run RStudio so that you do not need to install anything locally. Before starting the lab, there will be instructions on how to set up your Watson Studio project to run RStudio.

Let's start creating the Shiny application

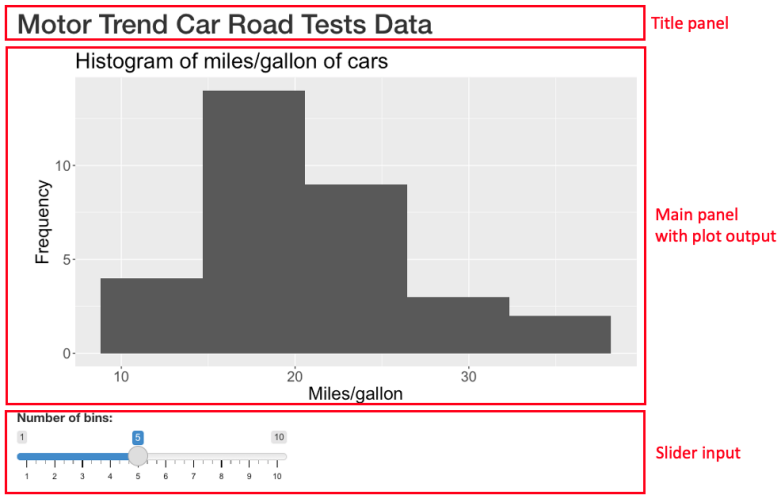
Goal

Create a dashboard that displays a histogram of the miles per gallons of cars in the `mtcars` dataset. The histogram will change depending on the number of bins a user inputs using a slider bar.

Expected Output

Below is the expected result from the lab. The dashboard application consists of three components:

- Title panel
- Main panel which contains the histogram
- Slider input that controls the bins in the histogram



This lab includes the following tasks:

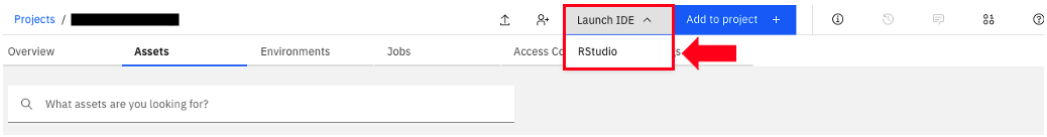
1. Add application title in the UI
2. Add plot output to main panel in UI
3. Create slider input in UI
4. Create plot and use input to change histogram in server
5. Run the application

Before starting these tasks, you will first set up Watson Studio to run RStudio. Then, you will download and become familiar with the starter code. The next two sections will guide you with these.

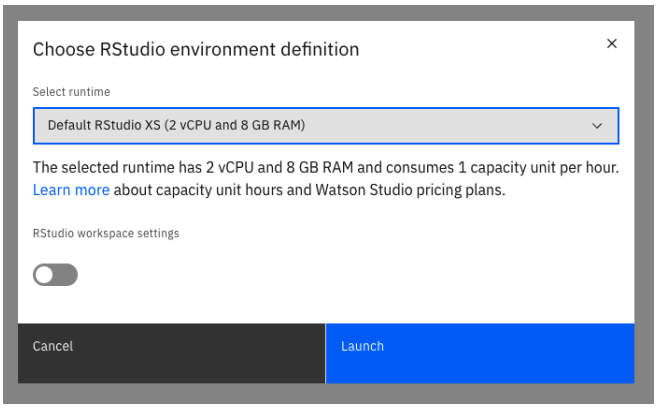
Launch RStudio in IBM Watson Studio

RStudio is the perfect development tool for R Shiny. Thus, in this project, you will be building the R Shiny app using RStudio, hosted by IBM Watson Studio.

1. First, make sure you have IBM Watson Studio setup and create a project, follow this guide for help.
2. In the Watson Studio project, click on "New Data asset" to add the two starter code files (ui.R and server.R) to the project.
3. To launch RStudio, click on the `Launch IDE` drop down menu, then click on `RStudio`.



4. You will then be asked to choose a run time. You can select any. Click on "Launch".



You will be directed to the RStudio IDE. Now you are all set up in Watson Studio to get started on the lab.

Download and understand the starter code

As you learned, a Shiny app contains two parts: the UI and the server. Open up the starter codes ui.R and server.R, let's look through them before starting the tasks. The `...` will be where you modify the code.

Download starter code

In the RStudio console, run the commands to download the two files to start.

```
# UI starter code
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DV0151EN-SkillsNetwork/labs/module_3/Lab4_shiny_basics/starter_code/ui.R"
download.file(url, destfile = "ui.R")

# Server starter code
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DV0151EN-SkillsNetwork/labs/module_3/Lab4_shiny_basics/starter_code/server.R"
download.file(url, destfile = "server.R")
```

Once these files have downloaded, click on these files to open them.

ui.R

You will work on the UI code first in tasks 1 to 3, then move on to the server code. Familiarize yourself with the starter code in ui.R. We'll go through it step by step:

- First, it loads the shiny library, if you want to run this locally make sure to install first:

```
install.packages("shiny")
```

- Then the UI is initialized with a `fluidPage()` in `shinyUI()`.
- There are three main components to the layout of this application:
    1. application title
    2. main panel which will have the histogram **output**
    3. a slider **input** to control the number of bins

server.R

After tasks 1 to 3, you will go on to the server.R file. Familiarize yourself with this starter code. We'll go through it step by step:

- First, it imports the `shiny` and `ggplot2` library. Notice how shiny should be imported in both ui.R and server.R
- The Shiny server function has two parameters, `input` and `output`, which can access objects from the UI. For example, `input$bins` and `output$histPlot`.
- Next, `renderPlot()` is used to change the output plot in the UI
    - Inside, you can use objects stored in `input` to change the `ggplot`.

Now that you understand the UI and server starter code, let's get started.

TASK 1 - Add application title in the UI

In the ui.R file, add the application title using `titlePanel()`. Set the `title` to `"Motor Trend Car Road Tests Data"`.

The code in this task should be:

```
titlePanel("Motor Trend Car Road Tests Data")
```

TASK 2 - Add plot output to main panel in UI

In the ui.R file, in `plotOutput()` add the `outputId` as `"histPlot"`. This will save an output plot named "histPlot" and can be accessed in the server. In task 6 you will use it with `output$histPlot`.

The code for this task should look like:

```
plotOutput("histPlot")
```

TASK 3 - Create slider input in UI

Next in the UI, you will add the slider input to change the number of bins the histogram has using `sliderInput()`. Try updating the slider input with the following required parameters:

- `inputId` - the input ID should be set to `"bins"`, in the server can be accessed with `input$bins` (see task 6)
- `label` - the label name is `"Number of bins:"`
- `min` - the minimum value is `1`
- `max` - the maximum value is `10`
- `value` - the default value is `5`

Your final code for this part should look similar to:

```
sliderInput(
      inputId = "bins",
      label = "Number of bins:",
      min = 1,
      max = 10,
      value = 5
    )
```

Note that you do not need to name these parameters (e.g. `sliderInput("bins", "Number of bins:", 1, 10, 5)` would also work) since they are required. Our sample code includes it since you are just learning these concepts and may need a reminder on what the parameters are.

TASK 4 - Create plot and use input to change histogram in server

You will fill in the `...` in the starter code. This is creating a histogram that you should be familiar with from previous lessons. The main difference is that you will use the dynamic input to change the bin numbers.

- In `aes()`, set `x` to `mpg`.
- In `geom_histogram`, set `bins` to the dynamic number of bins `input$bins`
- In `labs()`, set `x` to `"Miles/gallon"`, set `y` to `"Frequency"`, and set the `title` to `"Histogram of miles/gallon of cars"`

The `theme()` is changing the text size to be larger than the default. In the starter code it is setting the size to `20`, feel free to increase or decrease the `size` value.

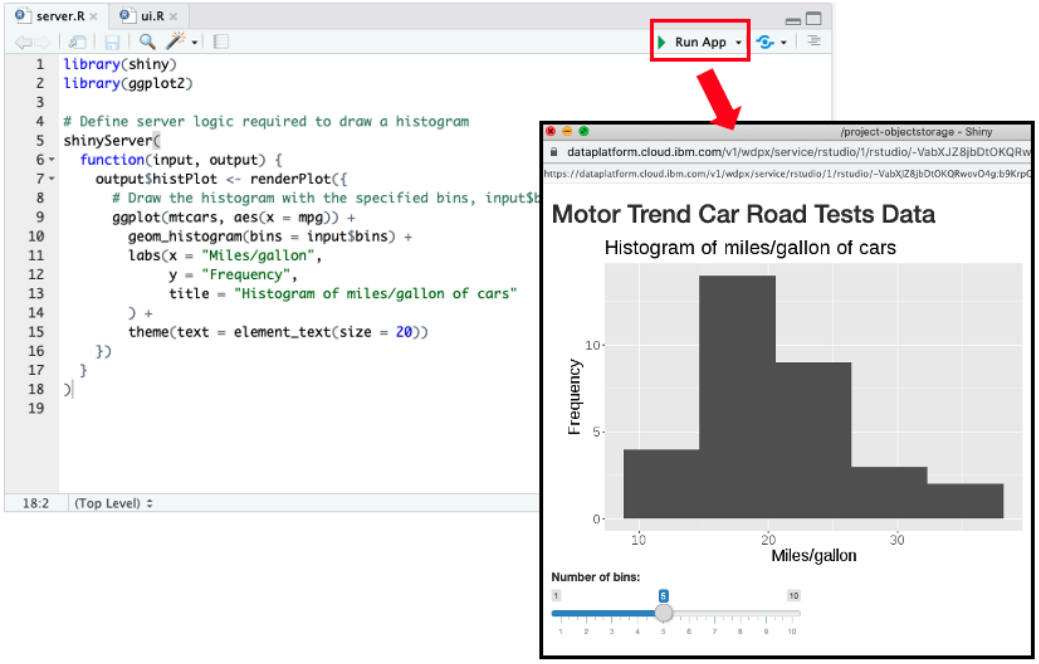
The final server code should look like:

```
library(shiny)
library(ggplot2)

# Define server logic required to draw a histogram
shinyServer(
  function(input, output) {
    output$histPlot <- renderPlot({
      # Draw the histogram with the specified  bins, input$bins
      ggplot(mtcars, aes(x = mpg)) +
        geom_histogram(bins = input$bins) +
        labs(x = "Miles/gallon",
             y = "Frequency",
             title = "Histogram of miles/gallon of cars"
        ) +
        theme(text = element_text(size = 20))
    })
  }
)
```

TASK 5 - Run application

You have completed both the UI and server files. You can test your app by clicking "Run App" on the top right. A new window should pop up with the completed Shiny application. It may take a few seconds for the histogram to load initially.



Once the graph has loaded, use the slider. You should see the histogram changing as you change the number of bins.

Solution

- Click here for the final UI code.
- Click here for the final server code.

Next Steps

Congratulations, you have successfully created your first Shiny application! If you were stuck on any section, you can review the solution code (UI and server). You can play around with the code and add any modifications you would like. In the next lab, you will use more Shiny widgets and components to make even more sophisticated dashboards.

Author(s)

Tiffany Zhu

Saishruthi Swaminathan

Changelog

Date	Version	Changed by	Change Description
2021-05-11	1.0	Tiffany Zhu	Created the initial version

Previous Continue