Shiny Basics

Estimated time needed: **30** minutes

Objectives

In this lab, you will learn to use the Shiny package to build interactive dashboard apps. Shiny is an R package for building interactive web applications that can help you with effective data storytelling. A Shiny app consists of two parts: the server and the user interface (or UI.) The server powers the app and holds the app logic. Both UI and the server work together to generate visualizations based on user input.

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 a pre-installed dataset in the R programming environment. This dataset is from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). The variables you will focus on are:

- `mpg` Miles/(US) gallon
- `cyl` Number of cylinders
- `disp` Displacement (cu.in.)
- `hp` Gross horsepower
- `drat` Rear axle ratio
- `wt` Weight (1000 lbs)
- `qsec` 1/4 mile time
- `vs` Engine (0 = V-shaped, 1 = straight)
- `am` Transmission (0 = automatic, 1 = manual)
- `gear` Number of forward gears

R-Studio on Coursera Labs

In this project, you will be building the R-Shiny app using R-Studio, hosted on Coursera.

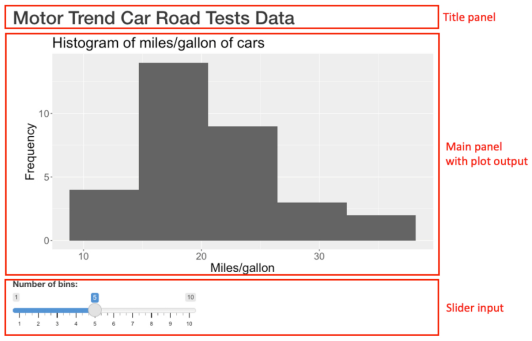
Goal

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:

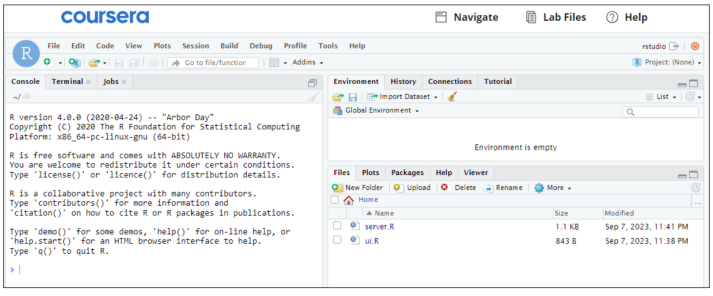
- Title panel
- Main panel which contains the histogram
- Slider input that controls the bins in the histogram



Getting Started

Step 1: Launch RStudio in Coursera

Launch RStudio lab using the `Launch App` button. You will be directed to the RStudio IDE as shown in the screenshot below. Now you are all set to get started on the lab.



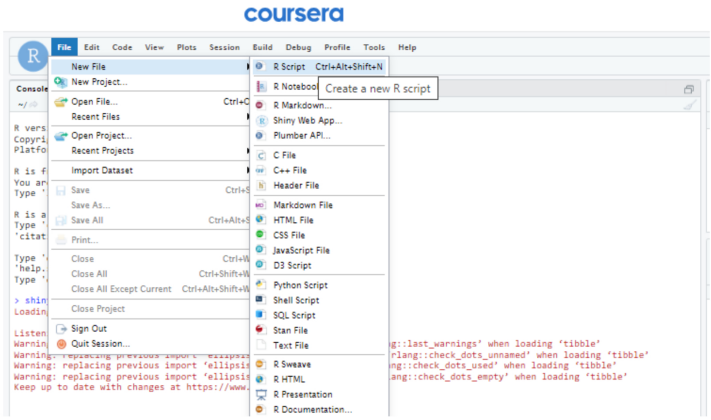
Step 2: Get Familiar with Starter codes (ui.R and Server.R)
This lab includes the following tasks:

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

Before starting these tasks, you need to become familiar with the UI and Server R scripts. The next section will guide you with this.

As you have learnt before, a Shiny app contains two parts: the **ui.R** and the **Server.R** script files. This lab comes with the pre-loaded starter UI and Server R scripts within lab environment. These files are available under `DV0151EN_Week3_CreateShinyApp_StarterCodes` directory when you launch the lab.

If these files are not available within the lab, create two new .R script files in the lab, copy and paste the starter codes given below.Click **File** > **New File** > **R Script** to create new script files as shown in the screenshot below.



▶ Click here for starter code for ui.R file

▶ Click here for starter code for server.R file

Step 3: Edit Starter codes - ui.R and server.R files

Before we begin, let's review the code. Some sections of the code within these files are marked as `...`, and it's essential to replace these placeholders with the correct code to obtain the desired output.

Note that this lab requires `shiny` and `ggplot2` package, which are already pre-installed in this lab environment. Hence, first line of the script directly loads the shiny library. If you want to run these scripts in your local RStudio, ensure you install these packages prior to loading the library. You will work on the UI code first for performing Tasks 1 to 3, then move on to the server code. You will perform Tasks 4 and 5 in the server.R file.

ui.R

Familiarize yourself with the starter code in ui.R. There are comments like "TASK 1" to show you which part of the code you will be working on. You will fill in the `...` in the starter code. We'll go through it step by step:

- First, load the `shiny` library.
- The `fluidPage()` in `shinyUI()` functions are used to initialize the UI.
- There are three main components to the layout of this application:
  i. application title
  ii. main panel which will have the histogram **output**
  iii. a slider **input** to control the number of bins

server.R

Familiarize yourself with this starter code. We'll go through it step by step:

- First, load the `shiny` and `ggplot2` library.
- 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`.

Exercises

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 in Server

You will fill in the `...` in the starter code to create a histogram plot that you are already familiar with labs 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.

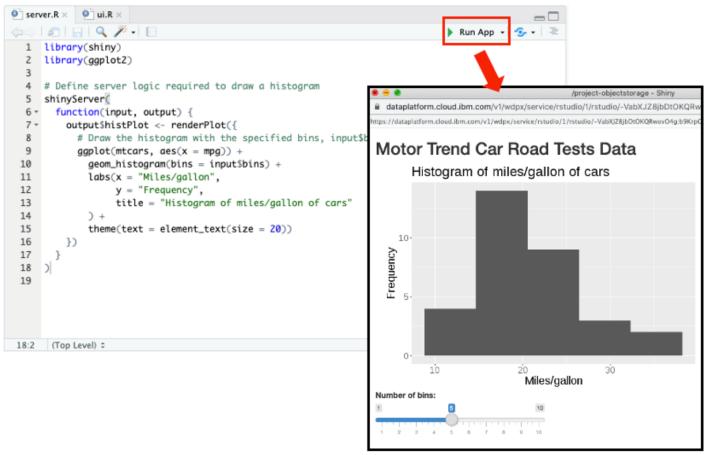
The final server code should look like:

```
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

If you were stuck on any section, you can review the solution codes given below. 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.

- 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!

Author(s)

Tiffany Zhu

Saishruthi Swaminathan

Skills Network

IBM