+ -
Helvetica N...
▼
Step 1 of 14

# Shiny Components

Estimated time needed: **45** minutes

### Objectives

After completing the lab you will be able to:

- Create a Shiny application with a sidebar layout
- Add HTML components like `br`, `h3`, and `p`
- Use different input widgets like select, numeric, and radio buttons
- Add several graphs using tab panels

### 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 explores the `mtcars` dataset more. You will use a sidebar layout that has three tab panels in the main panel. The first tab contains a histogram and boxplot to display numeric/continuous variables, the second tab contains a bar chart to display categorical variables, and the third tab has a scatter plot that shows the numeric and categorical variables. You will use several input widgets to change the variables being displayed.

### Expected Output

Below is the expected result from the lab. The layout of the application contains three parts (in red in the image):

- Title panel
- Main panel which contains the tab panels that hold the output graphs
- The sidebar panel which contains the input widgets

Additionally, this lab will use several Shiny widgets and components (in blue in the image) such as:

- Tab panels
- Headers (h3, h4)
- Select inputs
- Numeric inputs
- Radio button inputs
- Paragraph tag (p)

### This lab includes the following tasks:

1. Add application title in the UI
2. Add continuous and categorical select inputs in the UI
3. Create histogram settings using numeric and radio button inputs in the UI
4. Add variable map guide using paragraph tag in the UI
5. Add tab panels in the UI
6. Create histogram in the server
7. Create boxplot in the server
8. create bar chart in the server
9. Create scatter plot in the server

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/Lab5_shiny_components/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/Lab5_shiny_components/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 5, then move on to the server code. 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 can map the starter UI code to the final UI:

The inputs widgets you will use:

- Variable select inputs:
  - The continuous variable input will change the histogram, boxplot, and scatter plot
  - The categorical variable input will change the bar chart and the coloring of the scatter plot
- Numeric input: The number of bins input will change the bins in the histogram
- Radio button input: The fill option will change the fill of the histogram

### server.R

Then you will go on to the server.R file in tasks 6 to 9. Familiarize yourself with this starter code. The code is broken into four sections, one for each graph you will be creating. You can map the starter server code to the final dashboard:

## TASK 1 - Add application title in the UI

In the ui.R file, set the title to `"Data Exploration Using Shiny"`. Under the "TASK 1" comment, your code should look like:

```
titlePanel(title = "Data Exploration Using Shiny"),
```

## TASK 2 - Add continuous and categorical select inputs in the UI

Next, you will create the "Explore mtcars" section of the sidebar panel. There are three parts to add:

- Add `"Explore mtcars"` as the text for the header tag `h3`
- Add an input for selecting a continuous variable with `varSelectInput()` using the parameters:
  - `inputId`: the input ID (`"continuous_variable"`) to be used in the server
  - `label`: the label displayed in the UI
  - `data`: the filtered data, so here it contains the continuous variable columns using `select(mtcars, -categorical_variables)`. Note that `select()` is from [dplyr](#) which is in the tidyverse package.
  - `selected`: parameter is the default variable selected. Below is the code:

```
varSelectInput("continuous_variable",
               "Select Continuous Variable",
               data = select(mtcars, -categorical_varibles),
               selected = "mpg")
```

- Now try adding an input for selecting a categorical variable with `varSelectInput()`:
  - Set the `inputId` to `"categorical_variable"`
  - Set the `label` to `"Select Categorical Variable"`
  - Set the `data` as just the categorical variables, `mtcars[categorical_variable]`
  - Set the `selected` as `"cyl"`

The final code under the "TASK 2" comment should be:

```
# TASK 2: Add h3 and variable select inputs for continuous/categorical
h3('Explore mtcars'),
varSelectInput("continuous_variable",
               "Select Continuous Variable",
               data = select(mtcars, -categorical_varibles),
               selected = "mpg"),
varSelectInput("categorical_variable",
               "Select Categorical Variable",
               data = mtcars[categorical_varibles],
               selected = "cyl"),
```

## TASK 3 - Create histogram settings using numeric and radio button inputs in the UI

This part will add a numeric input to change the bins in the histogram and radio button inputs to change the fill color of the histogram (default color or blue). There is one heading and two input widgets to add:

- In `h3()`, add `"Histogram settings:"`
- The numeric input with `numericInput()` with the following parameters:
  - `inputId`: the input ID to be used in the server. Set it to `"bins"`
  - `label`: the label displayed in the UI, set it to `"Number of bins"`
  - `min`: the minimum allowed input number, set it to `2`
  - `max`: the maximum allowed input number, set it to`20`
  - `value`: the default number, set it to `10`
- The radio buttons with `radioButtons()`
  - `inputId`: the input ID to be used in the server, set it to `"bins"`
  - `label`: the label displayed in the UI, set to `"Number of bins"`
  - `choices`: the radio button choices, set to `c("default", "blue")`. So there will be two choices, the default color and blue.

The final code in this task should be:

```
# TASK 3: Add numeric input for bins and radio buttons for fill
h3("Histogram settings:""),
numericInput("bins",
            "Number of bins",
            min = 2,
            max = 20,
            value = 10),
radioButtons("hist_fill",
             "Histogram fill:",
             choices = c("default", "blue")),
```

## TASK 4 - Add variable map guide using paragraph tag in the UI

In this task, you will use some functions that resemble HTML tags. This means that you will be generating some HTML to customize the UI.

Add some text that tells the user what each variable name means so that they can understand your graphs better. You will use the header tag, `h4`, and the paragraph tag, `p`.

- Set the `h4` text to be `"Plot Variable Map Guide"`
- For the `p` tag, use the below code. The `br()` adds line breaks.

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

## TASK 5 - Add tab panels in the UI

Next, you will add three tab panels to hold all the graphs.

- The first tab panel will contain two plots for numerical variables. In `tabPanel()`:
  - Add the `title` of the panel as `"Distribution of Numerical Variables"`
  - Add `plotOutput("p1")` for the histogram to be displayed here. `"p1"` can be accessed in the server code.
  - Add `plotOutput("p2")` for the boxplot to be displayed here.
- The second tab panel will contain one plot. In the next `tabPanel()`:
  - Add the `title` of the panel as `"Distribution of Categorical Variables"`
  - Add `plotOutput("p3")` for the bar chart to be displayed here.
- The third tab panel will contain one plot. In the final `tabPanel()`:
  - Add the `title` of the panel as `"Plots for Observing Data Correlation"`
  - Add `plotOutput("p4")` for the scatter plot to be displayed here.

The code for the three panels should be:

```
# TASK 5: Add three panels
tabsetPanel(
  tabPanel(
    "Distribution of Numerical Variables",
    plotOutput("p1"),  # histogram
    plotOutput("p2")   # boxplot
  ),
  tabPanel(
    "Distribution of Categorical Variables",
    plotOutput("p3")  # bar chart
  ),
  tabPanel(
    "Plots for Observing Data Correlation",
    plotOutput("p4"))  # scatter plot
  )
)
```

You have finished the UI, now let's move on to the server code. If you want to see what your UI looks like right now, you can comment out the code in the `shinyServer()` or delete it for now like:

```
shinyServer(function(input, output) {

})
```

Then `Run App`. The application will be missing the graphs which will be created in the server code.

## TASK 6 - Create histogram in the server

In the server.R file, you will complete the logic for creating all the graphs. The first graph is the histogram, or `output$p1`.

Under the "TASK 6" comment, you will see that a base ggplot object is created first. Remember that ggplot allows you to add components on to create a graph.

- For the first `...` (the `x` parameter), replace with the continuous variable from the input with `!!input$continuous_variable`. `input$continuous_variable` is a string, so `!!` is a special function from `rlang` that interprets it as the variable.
- The next `...` in `paste()` should be replaced with `input$continuous_variable`, notice that you just need the string so `!!` is not needed. The `paste()` functions pastes together all the strings.

Next in the code is a conditional statement. It is checking the radio button value `input$hist_fill`. Make the following changes to the code:

- If the value is `"default"`, you will just add `geom_histogram(bins = input$bins)` and not change the fill.
- Otherwise, you will add `geom_histogram(bins = input$bins, fill = "dodgerblue3")` to change the fill color.

The final code in this task is

```
# TASK 6
output$p1 <- renderPlot({
  # Base ggplot object
  p <- ggplot(mtcars, aes(x = !!input$continuous_variable)) +
    labs(y = "Number of Cars", title = paste("Trend of ", input$continuous_variable))

  # Based on radio button input, change histogram fill
  if (input$hist_fill == "default") {
    p + geom_histogram(bins = input$bins)
  }
  else {
    p + geom_histogram(bins = input$bins, fill = "dodgerblue3")
  }
})
```

## TASK 7 - Create boxplot in the server

The next graph is the boxplot, which is saved as `output$p2`. Replace the two `...` with either `!!input$continuous_variable` or `input$continuous_variable` based on what you learned from the previous task.

The final code in this task is

```
# TASK 7: Boxplot
output$p2 <- renderPlot({
  ggplot(mtcars, aes(y = !!input$continuous_variable)) +
    geom_boxplot() +
    labs(title = paste("How", input$continuous_variable, "value is spread")) +
    coord_flip()
})
```

You should be familiar with all the components to create a boxplot yourself. One new concept is the `coord_flip()` that is added at the end, it flips the coordinates so that the boxplot is displayed horizontally instead of vertically.

## TASK 8 - create bar chart in the server

The next graph is the bar chart, which is saved as `output$p3`. Replace the four `...` with either `!!input$categorical_variable` or `input$categorical_variable`.

The final code in this task is

```
# TASK 8: Bar chart
output$p3 <- renderPlot({
  ggplot(data = mtcars,
         aes(x = factor(!!input$categorical_variable),
             fill = factor(!!input$categorical_variable))) +
    geom_bar() +
    labs(x = input$categorical_variable,
         title = paste("Trend of", input$categorical_variable))
})
```

## TASK 9 - Create scatter plot in the server

The last graph is the scatter plot. This plot will show the continuous variable that the user selects on the x axis and as the color will show the categorical variable the user selects.

- Set the `x` as `!!input$continuous_variable`
- Set the `color` as `factor(!!input$categorical_variable)`
- In `paste()`, replace `...` with `input$continuous_variable`

The final code in this task is

```
# TASK 9: Scatter plot
output$p4 <- renderPlot({
  ggplot(mtcars, aes(x = !!input$continuous_variable,
                     y = wt,
                     color = factor(!!input$categorical_variable))) +
    geom_point(size = 3) +
    labs(title = paste("Distribution of", input$continuous_variable, "with respect to Weight"))
})
```

You have finished both the UI and server code. Now, you can test that the application works by selecting `Run App` on the top right. It may take a moment for the graphs to load initially.

Once the graphs are loaded, you can play around with the dashboard!

### Solution

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

## Next Steps

Congratulations, you have successfully created a Shiny application using different components and widgets! 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. The next lab will the final project where you will leverage everything you have learned from the lessons and labs.

### Author(s)

Tiffany Zhu

Saishruthi Swaminathan

### Changelog

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

Continue