Module 7: ShinyR

Case Study

Things you will learn in this case-study:

1. Install and load shiny package
2. Components of shinyR
3. Fluid page and server-side logic
4. Shiny App, Shiny Text
5. Reactivity

Back Ground:

Retail Store house a variety of products, often the number of SKU's is in thousands. The merchandise managers are often considered responsible for analyzing the sales of these products. One of such products is the sweets related products like cake, icecream etc, where the merchandise manager believes that the sugar content in the products has an impact on the pricing power of those products and hence the sales. As a result, the merchandise head wants to have a dashboard which can be viewed and analyzed. This is where they require your support.

Overview of the problem:

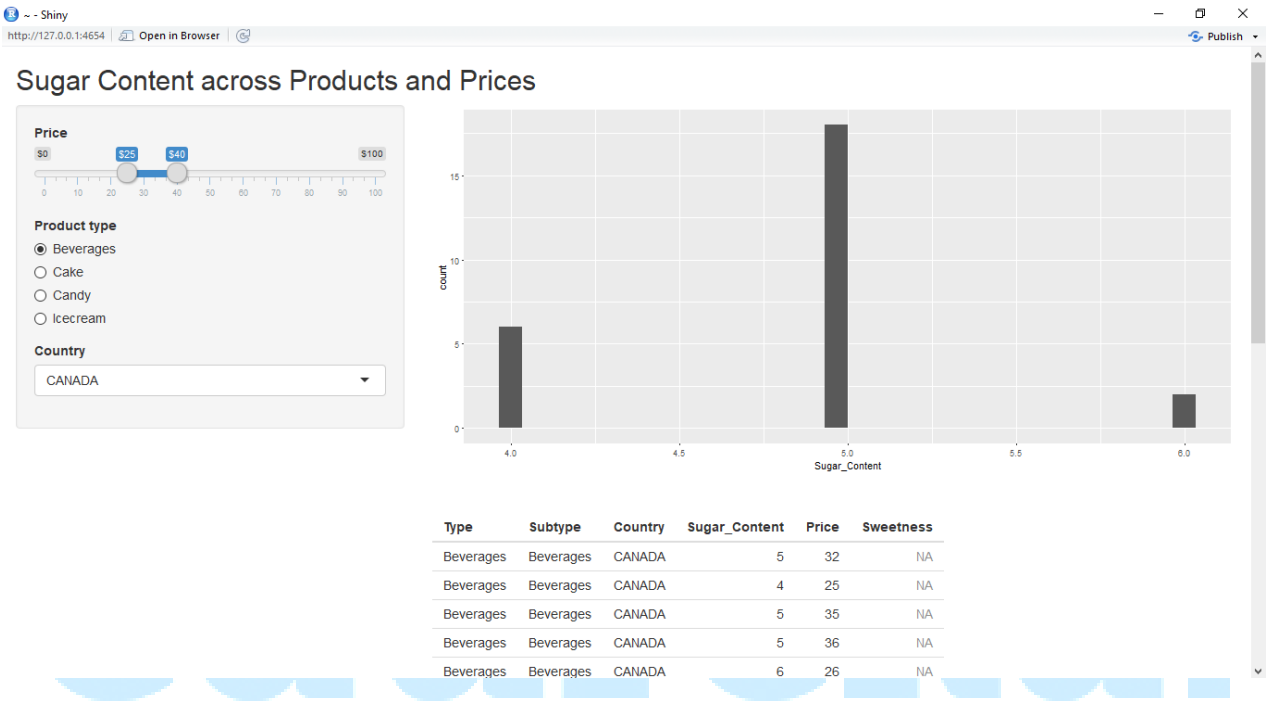
In this project, you will play the role of Data Scientist for this company, and you have been asked to create a web app which can help the management understand the variation in prices with sugar content across product type and across countries. Post this, the Merchandise Head will demonstrate the findings to the CEO. The Data you will be dealing is dummy data and is not real. The data is in data frame format which has been stored in the file named 'data10.csv'

Data set description:

Type: type of sweet - 'Beverages, Cake, Candy, Icecream'
Subtype: Same as above
Country: country from where the observation was recorded.
Sugar_Content: level of sugar in the product
Price: price of that product in dollars
Sweetness: sweetness level score as per the customer survey

Objective:

Based on the knowledge you acquired in Module, you are expected to create an app using R Shiny which will publish the below mentioned output.
You should do the following to create the above shiny output:



1. Load the required libraries and the data.
2. Understand the data structure and provide concise summary on the following -
   - no of observations
   - total number of variables
   - number of continuous variables
   - number of categorical variables
   - number of variables which have missing values
3. Create separate UI and server files and save it in a separate folder where you have kept the data as well.
4. The entire UI will be built by passing comma-separated arguments into the fluidPage()function. So, Let's build the UI interface of the app.
   - Start with the title panel and add the text 'Sugar Content across Products and Prices'
   - Add sidebarLayout, which will include the following
      I. sliderInput (please refer the above screenshot to fill this). This will be the 'priceInput'

II. radioButtons (please refer the above screenshot to fill this). This will be the 'typenput' which will take the Product type.
III. uiOutput (please refer the above screenshot to fill this). This will be the 'countryInput'

- Now let's build the layout of the mainpanel. For this, we need to add placeholders for outputs.
  I. At the top of the main panel we'll have a plot showing some visualization of the results. Since we want a plot, the function we use is plotOutput().
  II. Below the plot, we will have a table that shows all the results. To get a table, we use the tableOutput() function. Please review if the UI plot looks like the one in the screenshot.

5. Now, Implement server logic to create outputs
   - Recall that we have 3 inputs: priceInput, typeInput, and countryInput.
   - Create country output and select 'Canada' as shown in the screenshot above.
   - Now, filter the data based on the values of these three inputs. We'll use dplyr functions to filter the data, so be sure to include dplyr at the top. Then we'll plot the filtered data instead of the original data.
   - We'll create a plot and send it to the coolplot output. You have to create histogram of the variable 'Sugar_Content'.
   - Now, use the renderTable function to plot the data as shown in the screenshot above.

Once you have set up the app, please run the app and confirm if you are getting the same output as the screenshot.

Submission should include the following:

1. Please build the app as per the instructions outlined above and match it with the screenshot shown at the first page.
2. Summary on approach should be documented.
3. R Code File - both ui and server files.