


### Visualizing Data

Open RStudio and create a new project under your Module 3 folder and call it **Mod3Assignment1**. For this assignment, you will be creating an R Markdown document that will include the creation of plots and basics around exploratory data analysis (EDA). Once completed, all you need to do is submit the word document that is created.

#### Part 1: Create the R Markdown Document

- 1.) In RStudio, select *File -> New File -> Text File*. This will create a blank text file in the same area that scripts were created in previous assignments (upper left panel). Save this file to your project as **Mod3Assign1Answer.rmd** (it is important to save with the .rmd extension as this saves the text file as an R Markdown file).
- 2.) Create a Header 1 with the title: **Module 3 - Assignment 1**
- 3.) Create a Header 2 with the title: **Last Name, First Name** (replace with your name)
- 4.) Create a Header 3 with the title: **Data Visualization**
- 5.) Click on the dropdown arrow next to the Knit icon  at the top of the R Markdown Pane in RStudio and select Knit to Word.
- 6.) Notice that you now have a document in your files for the project named AssignmentAnswer.docx. This is the file you will be uploading later to Canvas.
- 7.) Download the two datasets from Canvas and save them to the folder you created for this assignment. You will then need to import those datasets into the current project folder (this is important because you will be importing the files in the R Markdown document in a few steps).
- 8.) Type out a paragraph after the header explaining that you will be using the datasets containing candy rankings and production that can be found on the Canvas course website.
- 9.) Next, add a chunk of code that will load both the tidyverse package. You will also need to load the datasets in the R Markdown document as well so use the following code<sup>1</sup>:

```
library(tidyverse)
```

```
candy_data <- read_csv("candy_data.csv")
```

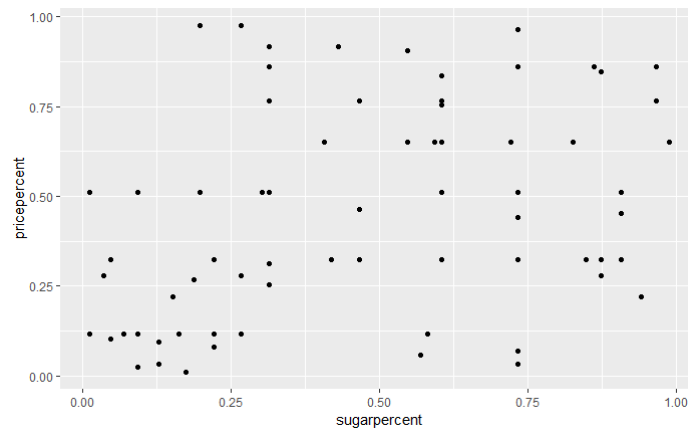
```
candy_production <- read_csv("candy_production.csv")
```

---

<sup>1</sup> Note: This code may change slightly due to corrections for a date field. See video on this assignment for more details.

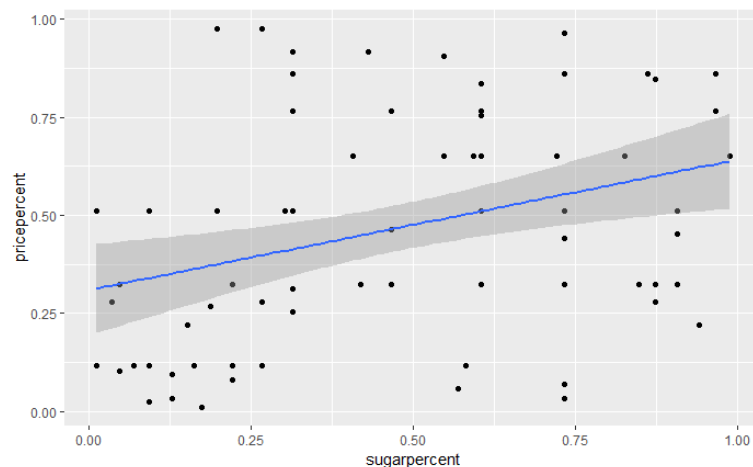
### Part 2 – Scatterplots

- 1.) For this assignment, we are going to explore the most popular layers (plots) within ggplot2: `geom_point()`, `geom_bar()` and `geom_line()`
- 2.) Create a Header 4 with the title: **Visualization with Scatterplots (`geom_point`)**
- 3.) For this part, we will be using a scatterplot to compare the sugar content to price of various types of candy on the market. You will need to create a new chunk of code after the header above where you will use the `candy_data` as your dataset and create a scatterplot with an x-axis of `sugarpercent` and y-axis of `pricepercent`. You should get a chart like the one below:



- 4.) A useful addition to the plot is a visual representation of how closely `sugarpercent` and `pricepercent` are related to each other, and the strength of that relationship. In the same chunk of code, let's add a fitted line to the chart to see this relationship. This can be added by including another line of code after the scatterplot code (don't forget to add the "+" sign after your scatterplot code to continue to the next line where this code is located):

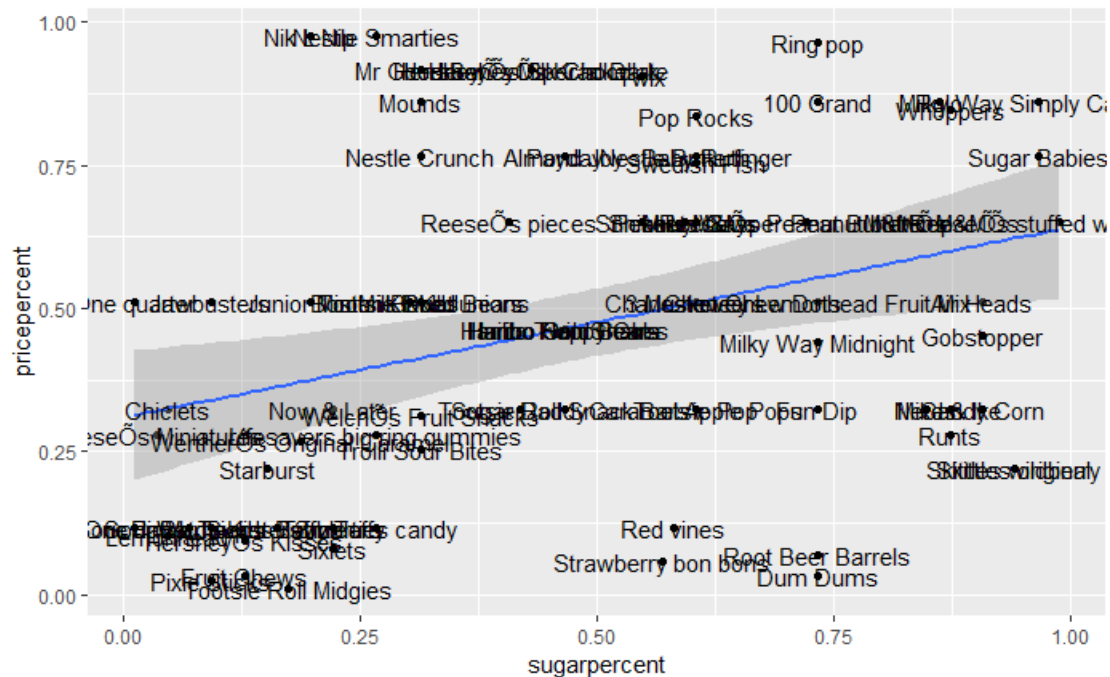
```
geom_smooth(method = "lm") + # adds a fitted line
```



## Module 3: Assignment #1

- 5.) Currently, we don't know what each dot represents. You will need to include an additional argument in your `aes()` command to include a label. This can be added by using a comma after your y-axis declaration and then include **label = competitorname**. You will also need to add an additional line of code to include a layer on your plot to display the text on the plot. To do this, add the following line of code after your fitted line that was added in the previous section (the second line below):

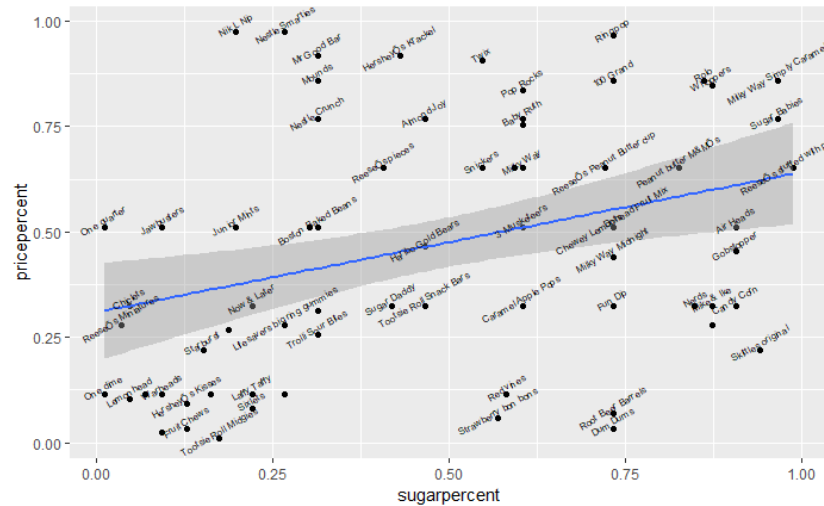
```
geom_smooth(method = "lm") + # adds a fitted line  
geom_text() # and then add labels to the points
```



- 6.) While this helps with what each dot represents, it is still challenging to see what each one is. You can change how the text appears by using the `geom_text()` layer with adding the code below:

```
geom_text(check_overlap = T, # automatically reduce overlap (deletes some labels)  
          vjust = "bottom", # adjust the vertical orientation  
          nudge_y = 0.01, # move the text up a bit so it doesn't touch the points  
          angle = 30, # tilt the text 30 degrees  
          size = 2 # make the text smaller (to reduce overlap more)  
          ) # and then add labels to the points
```

## Module 3: Assignment #1



- 7.) It would also be nice to change the labels as well. This is easily done by adding an additional line of code to our function:

```
labs(title = "Sugar by Price Scatterplot", # plot title
     x = "Sugar content (percentile)", # x axis label
     y = "Price (percentile)" # y axis label
)
```



- 8.) Based upon your results, create a paragraph after the chunk of code that explains which candy has the most sugar and lowest price. Also, which is the most expensive candy with the highest sugar content?

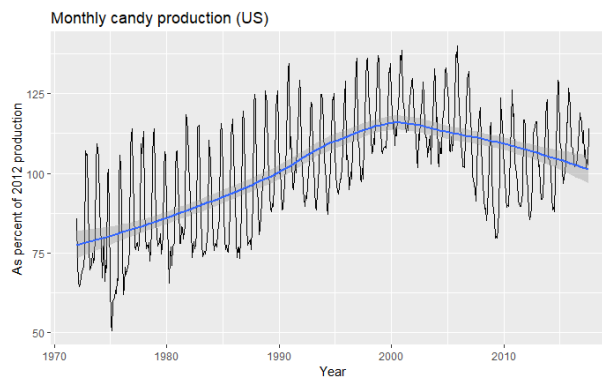
### Part 3 – Line Charts

- 9.) We are now going to use the candy production dataset to see how monthly candy production has changed over the years compared to a specific year (2012).

- 10.) First, create a header 4 with the title **Line Chart of Candy Production**

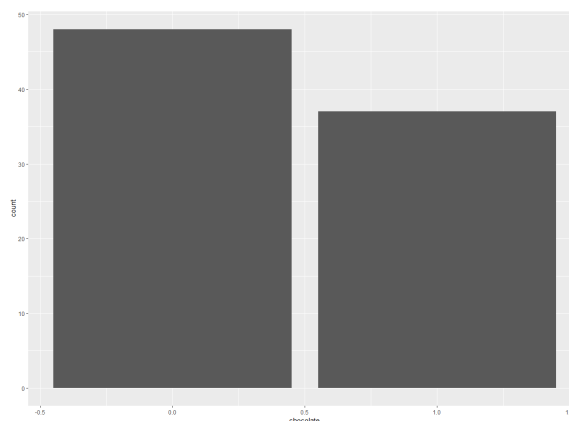
## Module 3: Assignment #1

- 11.) Add a short paragraph describing that we are using the candy production dataset that will display a specific date and how production during that month is compared to 2012.
- 12.) Add a new chunk of code and create a line chart using the `candy_production` dataset with an x-axis of `observation_date` and y-axis of the variable `IPG3113N` (this is a percentage that compares the monthly production for that date to 2012). Also add a fitted line (just leave the parentheses blank and don't add a method). Finally add a title and axis labels with the following:
- Title => Monthly Candy Production
  - X-axis => Year
  - Y-axis => As percent of 2012 production



### Part 4 – Bar Charts

- 13.) Maybe we would like to know more about the `candy_data` such as what percentage of candy has different ingredients. This can be done using a bar chart.
- 14.) First, create a header 4 with the title **Bar Chart of Ingredients**
- 15.) Create a new chunk of code and create a bar chart that uses the `candy_data` dataset and will plot the variable `chocolate`. Below is the output.



## Module 3: Assignment #1

16.) Notice that there are only two values in our dataset (0 and 1). If you go back to the dataset, you will notice that 0 represents No Chocolate and 1 represents Chocolate. This is because the dataset is using what we call categorical variables (either 0 and 1). We can also represent these as Logical or True/False. We will learn more about how to change these variables in subsequent sections but for now, add the following code after your bar chart in the same code chunk:

```
# select out the columns that have the features of the candy (chocolate, caramel, etc.)
candyFeatures <- candy_data %>% select(2:10)

# make sure that these are booleans (Logical)
candyFeatures[] <- lapply(candyFeatures, as.logical)
```

17.) The above code will make a subset of data called candyFeatures and then changes the 0,1 to False and True. The reason we create the subset is to ensure that we do not alter our original dataset.

18.) Now, add the code to create another bar chart but use this new dataset (candyFeatures) instead of the original. You will get a chart similar to the one below:

