Analysis of Different Items in Different Health Foods Stores

Keaton Raymond
kraymond@uccs.edu

Abstract

This analysis aims to analyze different metrics of items sold in health foods stores. This will be done using ggplot2, a powerful visualization package in R, to create histograms, bar charts, and scatterplots. Through these visualizations we will discover that a higher visibility means a lower stock, and gain insights into product pricing as well as the distribution of products within the store for product groups or fat content.

Introduction

For this assignment, I was tasked with recreating a total of nine graphs that were given to me using the ggplot2 package in R. Ggplot2 is a powerful package full of functions to make any visualization you could imagine. Of the graphs I must recreate, three were histograms, three were bar charts, and three were scatterplots. I will also have to use packages to arrange multiple graphs into a single image.

Data

The data given to us represents a list of what are presumably food items in different stores. There are metrics for both the items themselves and for the store that the item is in. The item columns include weight, fat content, and type (dairy, breakfast, canned, etc.), while the store columns include the size, type, and establishment year. The data didn't require any cleaning, other than dropping rows that had NA values in the columns that I was looking at for a given graph (apart from the graph in Part C, which included null values).

For combining the data, my first thought was to merge the two tables based on the assumption that the two tables had matching unique item identifiers. However, this resulted in many duplicated rows for a total of over 27,000 rows. When creating the graphs, they had the same shape as the given graphs, but the y axis was roughly doubled in all the graphs, other than the scatter plots in Part D. I then reread the question and noticed that it said to combine the data, not to merge the data. I tried instead putting all the data into one table, keeping all the original rows, and essentially appending the two as opposed to merging them. To do this, both data frames must have all the same columns. With some visual inspection, I noticed that both tables had all the same columns, except the extra column in the train data set, called Item_Outlet_Sales. So, I added that column to the test data set and set it to NA, which meant that I could now append the two data frames and all my graphs were fixed, so long as I accurately dropped the rows with NA values in the columns that I was actively working with.

Graphs

Part A

Part A (Fig. 1) consisted of three separate histograms that are all arranged onto a single image using the grid.arrange() function.

The first graph was the counts of different item weights. To create it, I first looked at how to create a histogram using ggplot2 and initially ran it with the x axis being item_weight. By default, the y axis of the histogram doesn't need to be set, as it counts the instances of each weight that fits into the given bucket. From there, I had an all-gray histogram that didn't have any outlines on the bars. I then looked at how to change the color, as the default gray wasn't quite correct. Finally, I looked at how to add an outline to each border and the first graph was complete. This graph shows the count of items of different weights. You can see that, other than a few outliers such as the lightest or heaviest items, the distribution of the items by weight is mostly within a deviation of roughly 200, with the highest concentration being within the weight range of 6-10.

The second graph was the counts of different item visibilities. To create this graph, I used the same code as I used to create the first graph, except with a different x axis. When looking at this graph, you can see that the more visible an item is, the lower the count it is. Additionally, around visibility 0.2 the count stays extremely low. Assuming that this data includes all items in the store including back stock, this makes a lot of sense; most of the items in the average grocery store are kept in back stock where they are not visible by the customers. The most visible items, such as items by the register, are the most purchased items, meaning that the store would have a lower count of those items.

The third graph represented the count of items of different MRPs (Maximum Retail Price). To create this graph, I initially used the same code that I had used for the previous two graphs except without the outline for the bars. I then looked into how to change the width of the bars on the histogram, which is done through changing the "binwidth" option. To determine the correct bin width, I plugged in increasingly small numbers, starting with around 5 and finally ending at 0.5. When looking at this graph, you can see that there are a lot of spikes within this graph, which I would assume is because of the common trend in the US of the price of an item ending in 95 or 99 cents and many items therefore having a very similar price. For example, it is very common to see two items that are priced at $49.99, but it is very uncommon to see items priced at $50. There are also three gaps in the graph around 50, 150, and a little bit past 200. My guess is that this can also be ascribed to the same reason that there are spikes.

The final thing that I needed to do was to arrange all three graphs into a single image. Thus far for testing each of the individual graphs, I simply created all of them in their own image. Now that I had all the individual graphs created properly, I researched the grid.arrange() function and found that it is actually very simple to use. I first just passed all the graphs into the function, but this arranged them in a way where all three graphs were in one column. So, I found in the documentation that there is an argument to change the number of rows the graphs are in and changed that argument to two. This forced the graphs to be in two rows, which created the proper arrangement.

Part B

Part B (Fig. 2) was only a single chart where I needed to get the counts of each item type and represent them as a bar chart with a legend on the right showing each item type and its respective count.

To create this graph, the first thing that I had to do was find out how to create a data frame with each unique item type and its respective count. To achieve this, I found that the table() function in R can be used to create a frequency table quickly and easily. This returned a table with two columns, the unique item types, and the count for each one. From there, I just used the as.data.frame() function to convert the table back to a data frame so that I could use ggplot2. Now, I could move on to creating the graph; I followed the same methodology described in Part A, except to create a bar chart and now a histogram. I used the item type as the x axis, the count as the y axis, then slowly added the other features such as colors, the graph title, the legend, and changing the angle of the x axis labels to be at a 45-degree angle so that they were legible. Looking at this graph, you can see that the stores have the largest count of fruits and vegetables and snacks, and the least amount of seafood and breakfast foods. To the right of the graph, there is also a legend where you can see in ascending order the exact amounts for each item type. This distribution of food types tells me that this is likely a dataset for health food stores as opposed to general grocery stores.

Part C

Part C (Fig. 3) consists of two separate bar charts that are both arranged onto a single image using the grid.arrange() function.

For both graphs, creating the basic graph followed the same process as Part B for generating the frequency table and generating the basic bar chart. However, when generating the frequency table for the outlet size graph, I toggled the useNA option to "ifany", which includes NA values as those were included in the graph. To add the count labels onto the bars of the graph, I found that I could add the function geom_label() to the plot, which adds labels that I can set to be equal to the count and add a white background for each of the labels. The only other extra step, other than changing the color of the bars, was for the outlet size graph; I needed to reorganize the bars because, in the given graph, the NA bar is on the left, but by default it was on the right. To change this, I used the scale_x_discrete() function which allows you to reorder the bars by name with the limits option. Finally, to arrange the graphs into a single image, I followed the same process as in Part A, except instead of changing the number of rows to two, the number of rows was set to one, which forced the graphs to be next to each other as opposed to on top of each other.

When looking at the outlet size graph which shows the counts for each of the outlet sizes as well as a count for the rows that had an NA outlet size, you can see that there

are the most medium sized outlets and the least high sized outlets. However, there are a lot of rows that have an NA size, meaning that, depending on the outlet size for those NA rows, the results of this graph could change.

The fat content graph shows the counts for each of the different fat contents of the items, either low fat or regular. This graph has five columns, but only has two different options, because "LF", "low fat", and "Low Fat" are all counted as different values but have the same meaning, and "reg" and "Regular" are also counted as different values but have the same meaning. The high concentration of low-fat items shown in this graph further solidifies my conclusion from above that this data is likely on a health food store. If I was tasked with creating this graph on my own, I would combine the columns that have the same meaning to make it more concise and readable.

Part D

Part D (Fig. 4) consists of three separate scatterplots arranged in a single image, except, unlike Parts A or C, which used the grid.arrange() function, uses the plot_grid() function to arrange the three graphs into one image.

Creating all three of the graphs followed the same process and used the same code, other than the value represented on the x axis. Upon initially looking, these graphs looked like a bar chart or a histogram, except with points rather than solid bars. This took me a second to understand that they are actually scatterplots where the data just happens to be arranged in a manner that looks similar to a bar chart or histogram. For these graphs, I followed the same methodology as with the previous graphs, except created a scatterplot rather than a bar chart or a histogram. The only extra options that were necessary for these graphs past the base scatterplot were changing the color and the opacity of the points. Changing the opacity gives you the ability to see another dimension of the scatterplot because you can see the density of the points based on how opaque that part of the graph is. For example, if one part of the graph is very transparent, then you know that there are only a few points there; However, if another part is fully opaque, you know that there are many points there.

For this part, the main difficulty came from arranging them into one image. To arrange the graphs, it is unlike Parts A or C, where all the graphs in the arrangement are of the same size and you only need to change the row argument to get the proper alignment. For this arrangement, there is one graph on the top row that is the same width as the two other graphs on the bottom row combined. To accomplish this, I needed to create two separate arrangements. One arrangement was the bottom row, with the visibility and MRP graphs side by side. Then, the second arrangement was the final arrangement, which consisted of the weight graph on top and the previous arrangement created on the bottom. This meant that both the top graph and the bottom arrangement were the same width, meaning that the top graph would be as wide as both bottom graphs combined.

In the first graph, each point represents the value for the outlet sales an item has and the weight of said item. This graph shows no real correlation between item weight and sales, as all the points are relatively evenly distributed within a rectangle with a few outliers.

In the second graph, each point represents the value for the outlet sales an item has, as well as the visibility of said item. It is very difficult for me to make sense of this graph, since, according to this graph, the lower the visibility of the item, the more sales it has. This leads me to believe that there may be either a flaw in the way the data is presented, or I may be reading it wrong, and a lower visibility number represents a higher visibility.

In the second graph, each point represents the value for the outlet sales an item has and the MRP of said item. You can see the same three gaps that I had noticed in Part A and I believe that these gaps are there for the same reason as they were in Part A. You can also see that up to 200, the height of the group increases as MRP increases, but the density (i.e., total items sold for that grouping) is very dense; but, in that 200+ group, the density decreases meaning, that the total items in that group are lower, but it has a higher peak for sales.

Results and Analysis

These graphs helped us to discover some insights into the metrics stored for items in health foods stores. We can gain insights about the distribution of item categories, how item visibility affects inventory and sales, and pricing patterns. With some more analysis and visualization, I believe that we would be able to draw further conclusions relating to the sales of specific items and how that relates to their visibility and potentially give these stores tips on how to increase sales. If I could change something about this data to be able to draw further conclusions, I would add a column that gives the exact item type as opposed to a category. For example, we are currently only able to see if a food is canned, but not what it is specifically, which could be anything from pineapple to chili to peppers.

Conclusion

Overall, I discovered how powerful ggplot2 is, and I have only scratched the surface and created very basic visualizations. I understand now why it is the number one visualization package in R. There is a function that allows you to create any graph you want and add anything to the graph that you would like.

Appendix

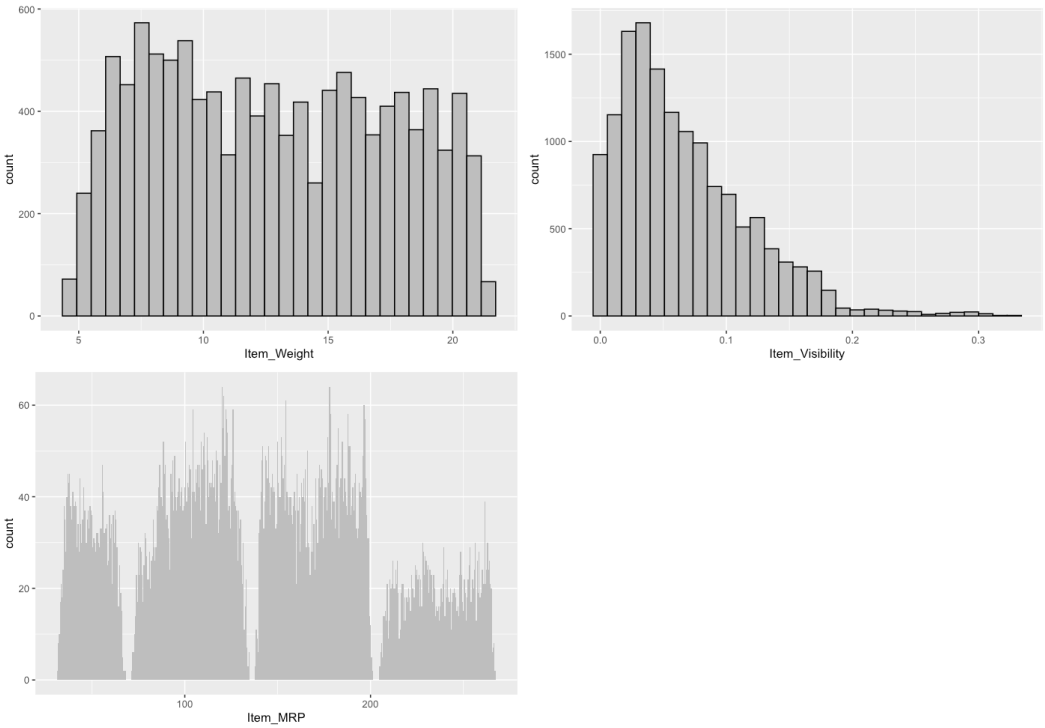

Figure 1: Part A Histograms

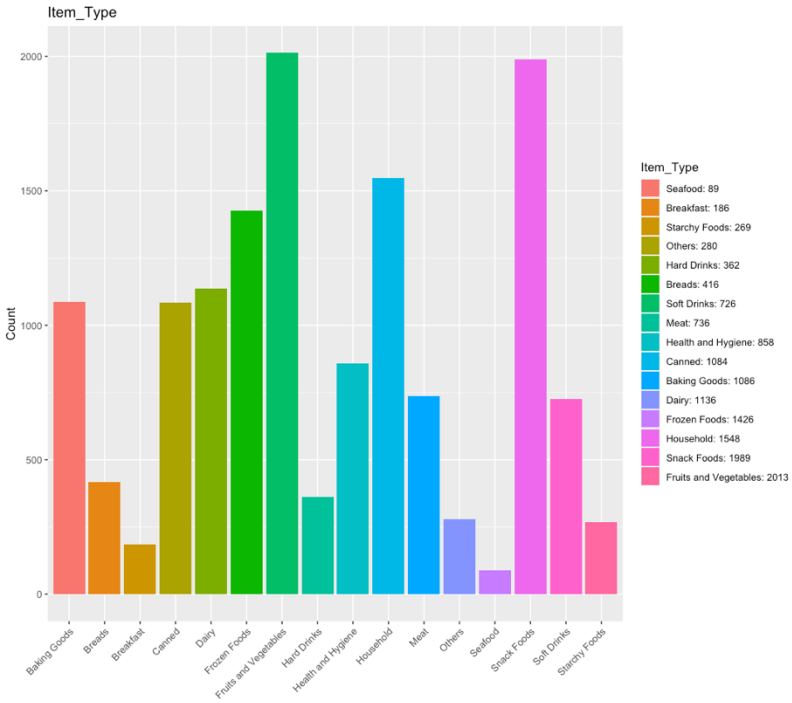

Figure 2: Part B Bar Chart with Legend

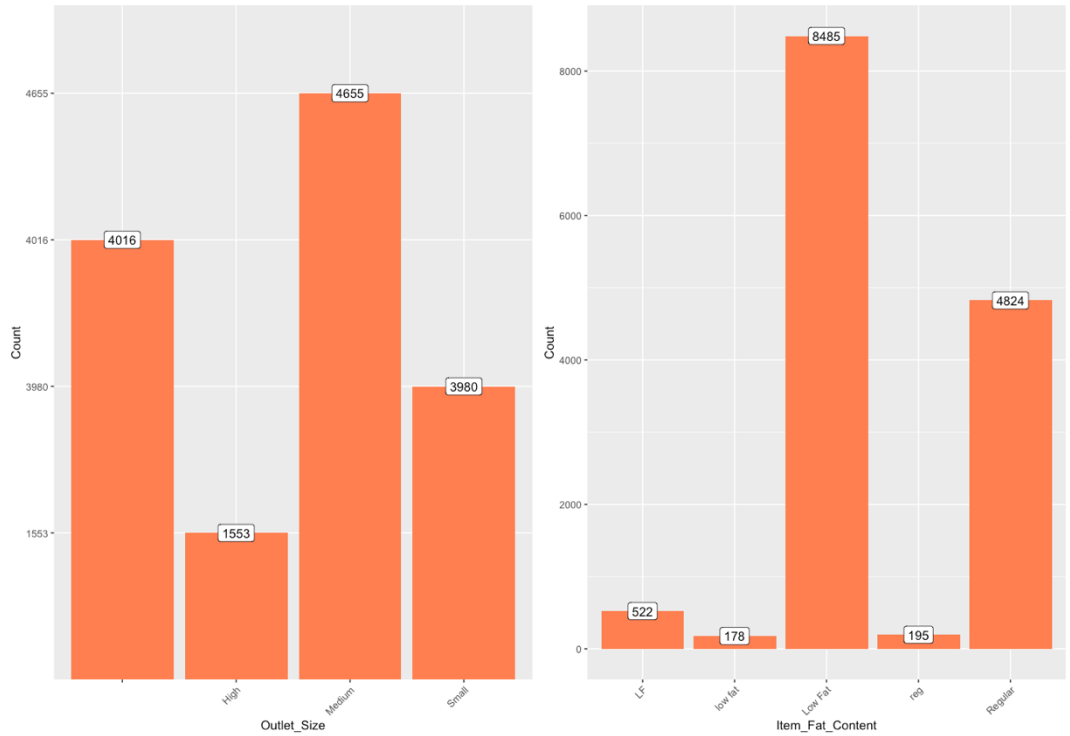

Figure 4: Part C Bar Charts with Labels

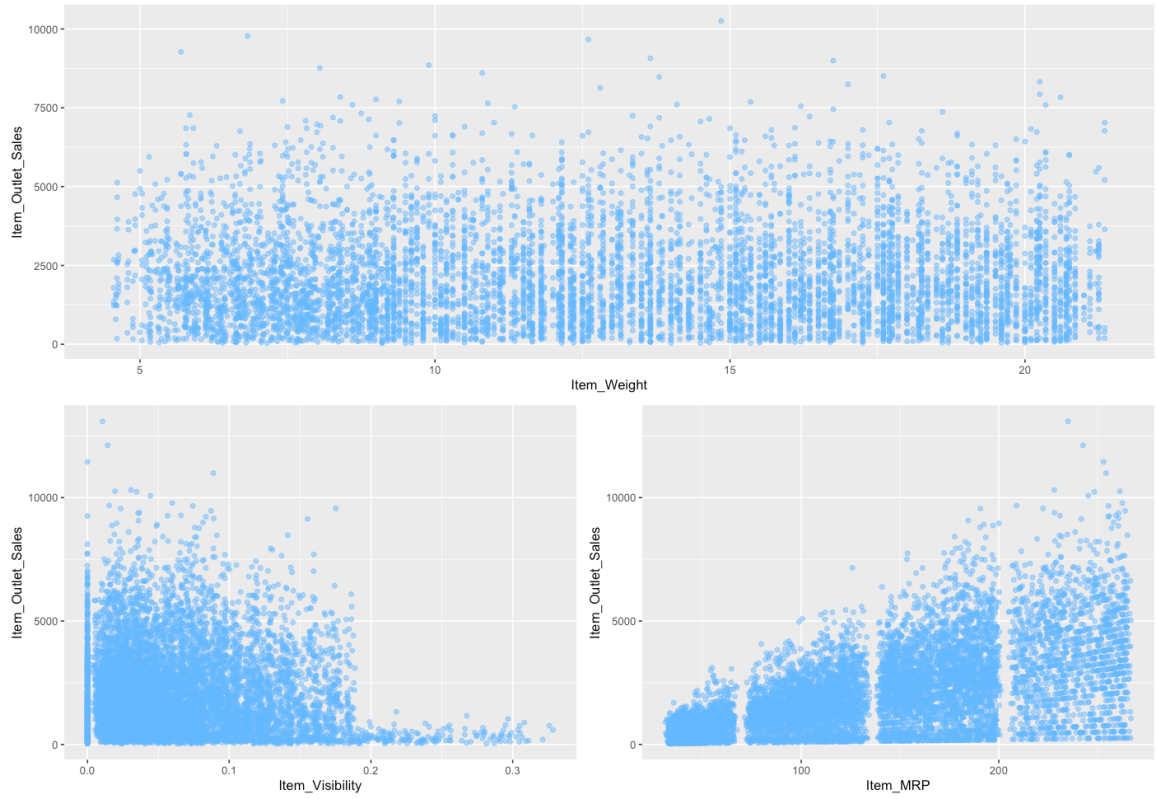

Figure 3: Part D Scatterplots