Regional Analysis of a Supermarket – Technical Report

Ethan Ericson

University of Arkansas

DASC 1223H – Data Science in Today's World

Dr. Schubert

03/01/2024

Technical Contributions

Overview of Techniques

For our study on the product performance of a fictitious supermarket, I contributed several valuable techniques. These techniques were:

1. Chi Squared Analysis
2. Mosaic Plot Analysis
3. Bar Chart Analysis
4. Random Forest Regression Analysis
5. Target Guided Ordinal Encoding

Through my knowledge of these domains, I was able to craft informative visualizations and contribute to the design of our machine learning model.

Technical Steps

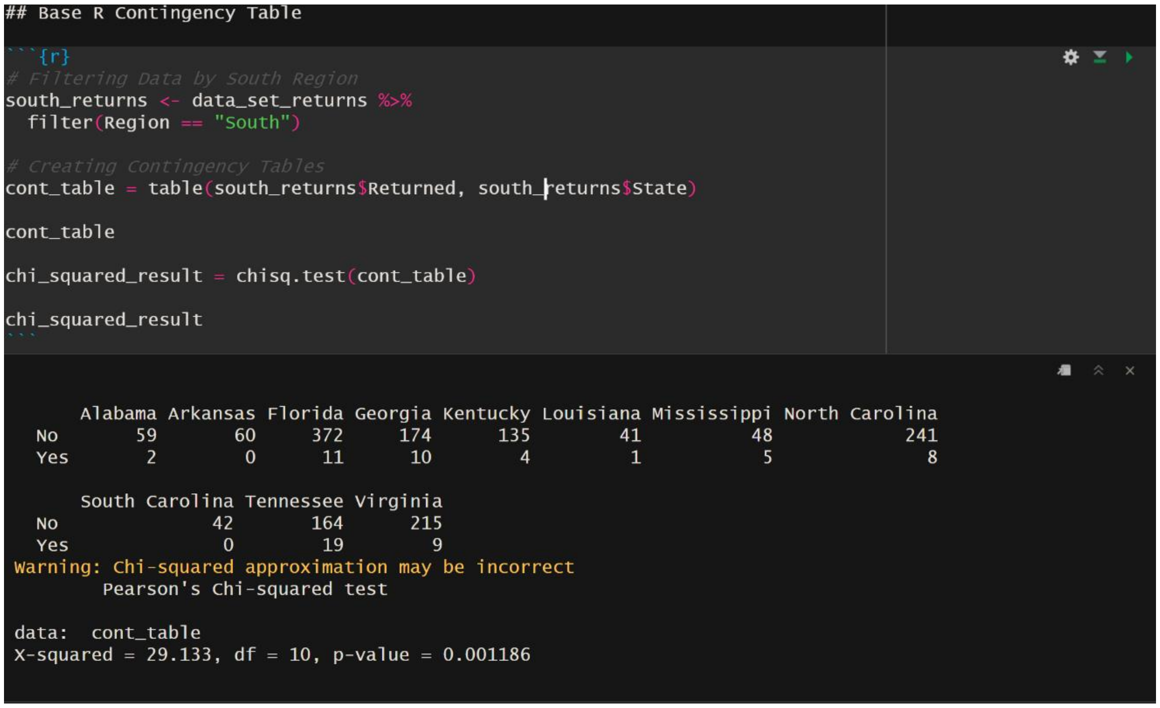
Chi Squared Analysis and Mosaic Plotting

The first technical step I took before beginning the study was to research Chi Squared Analysis in R. I knew from our dataset, (US) Superstore – Sample, that we would be working with many categorical variables, and I knew that Chi Squared tests were made for categorical variable comparison. Through online keyword searches, I was able to find a website that explained the steps of performing Chi Squared Analysis in base R. In R, "a chi-square test of independence is used to analyze the frequency table (i.e. contingency table) formed by two categorical variables. The chi-square test evaluates whether there is a significant association between the categories of the two variables. (STHDA)" So, according to the STHDA (Statistical Tools for High-throughput Data Analysis) website, I could use contingency tables to determine the association between two categorical variables.

The next step in this process was learning how to create a contingency table. The STHDA website provided sample code for this process, which created contingency tables in R using the base R table() function. After cleaning the returns dataset with distinct() and left joining returns to the orders dataset, I filtered by the southern region and created a contingency table to visualize the relationship between returns and states in the south.

Image 1

Base R Table and Chi Squared Test



~~~
## Base R Contingency Table

```{r}
# Filtering Data by South Region
south_returns <- data_set_returns %>%
  filter(Region == "South")

# Creating Contingency Tables
cont_table = table(south_returns$Returned, south_returns$State)

cont_table

chi_squared_result = chisq.test(cont_table)

chi_squared_result
```
~~~

```
       Alabama Arkansas Florida Georgia Kentucky Louisiana Mississippi North Carolina
   No       59       60     372     174      135        41          48            241
   Yes       2        0      11      10        4         1           5              8

       South Carolina Tennessee Virginia
   No              42       164      215
   Yes              0        19        9
Warning: Chi-squared approximation may be incorrect
         Pearson's Chi-squared test

data:  cont_table
X-squared = 29.133, df = 10, p-value = 0.001186
```

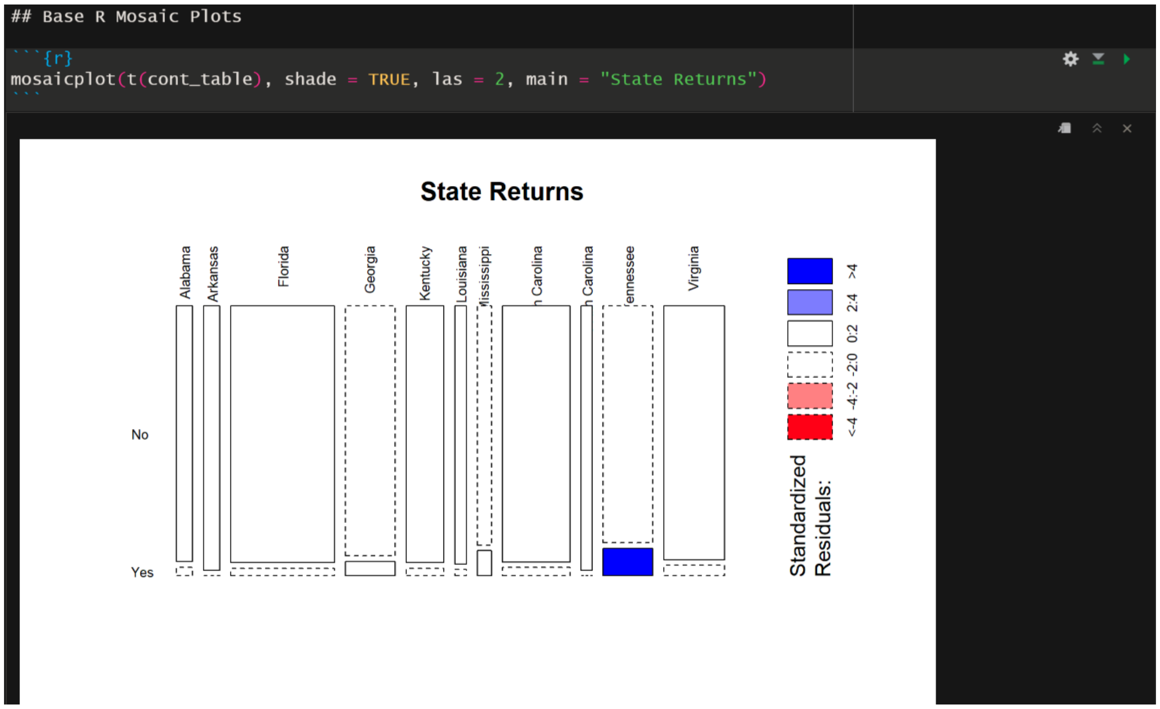
Note. This image depicts a contingency table created in base R with a Chi Squared Test preformed on it.

Following the creation of the contingency table, which you can see in Image 1, I used the chisq.test() method in R to perform a Chi Squared test on the table, resulting in the relevant statistics shown at the bottom of Image 1. I had now learned how to perform a Chi Squared test in R, and now I could focus on visualizing the results.

Still relying on the STHDA website, I used the mosaicplot() base R function to generate a mosaic plot visualizing the residuals of the contingency table, as well as the number of observations in each division of the state variable through the width of the mosaic bars.

Image 2

Base R Mosaic Plot



~~~
## Base R Mosaic Plots

```{r}
mosaicplot(t(cont_table), shade = TRUE, las = 2, main = "State Returns")
```
~~~

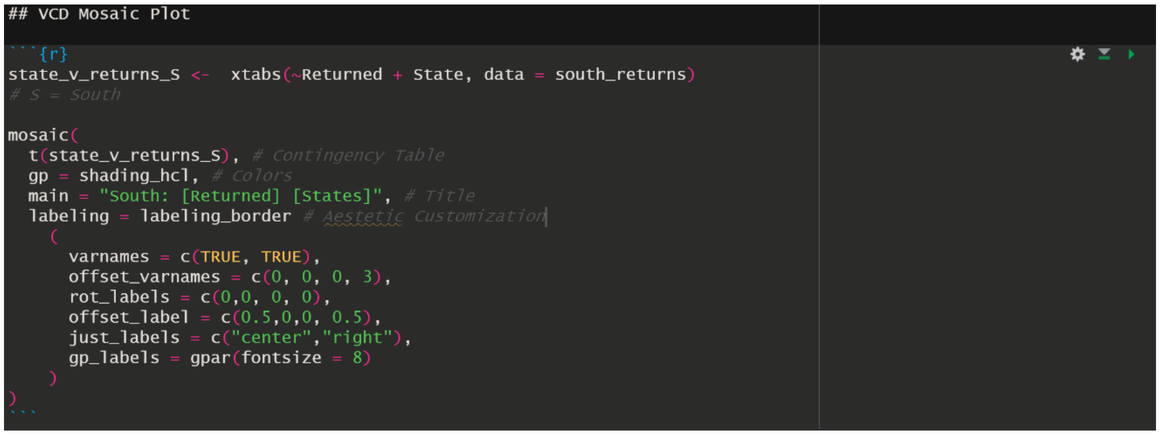
Note. This image depicts a base R mosaic plot.

Having generated the mosaic plot depicted in Image 2, I worked to improve the style of the plot. There were a couple of problems initially. First, the names of the state variables overlapped the mosaic bars. Second, there was no p-value displayed on the visualization.

To solve this issue, I talked with my peer mentor Coy, who advised me to use the function mosaic from the “vcd” package to load my mosaic plots. I took his advice and found the documentation for the mosaic function. Using this documentation as my guide, I transformed my initial base R mosaic plot into a vcd mosaic plot.

Image 3

VCD Mosaic Plot Code

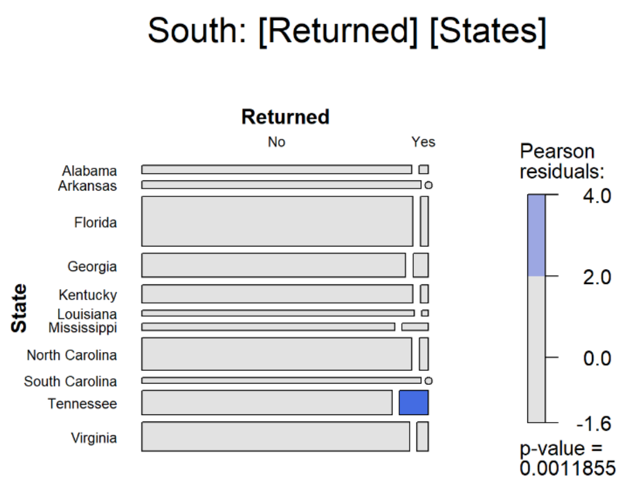


~~~
## VCD Mosaic Plot
```{r}
state_v_returns_S <-  xtabs(~Returned + State, data = south_returns)
# S = South

mosaic(
  t(state_v_returns_S), # Contingency Table
  gp = shading_hcl, # Colors
  main = "South: [Returned] [States]", # Title
  labeling = labeling_border # Aestetic Customization
    (
      varnames = c(TRUE, TRUE),
      offset_varnames = c(0, 0, 0, 3),
      rot_labels = c(0,0, 0, 0),
      offset_label = c(0.5,0,0, 0.5),
      just_labels = c("center","right"),
      gp_labels = gpar(fontsize = 8)
    )
)
```
~~~

Note. This image depicts the code used to display the VCD Mosaic Plot.

Image 4

VCD Mosaic Plot



Note. This image depicts the VCD Mosaic Plot.

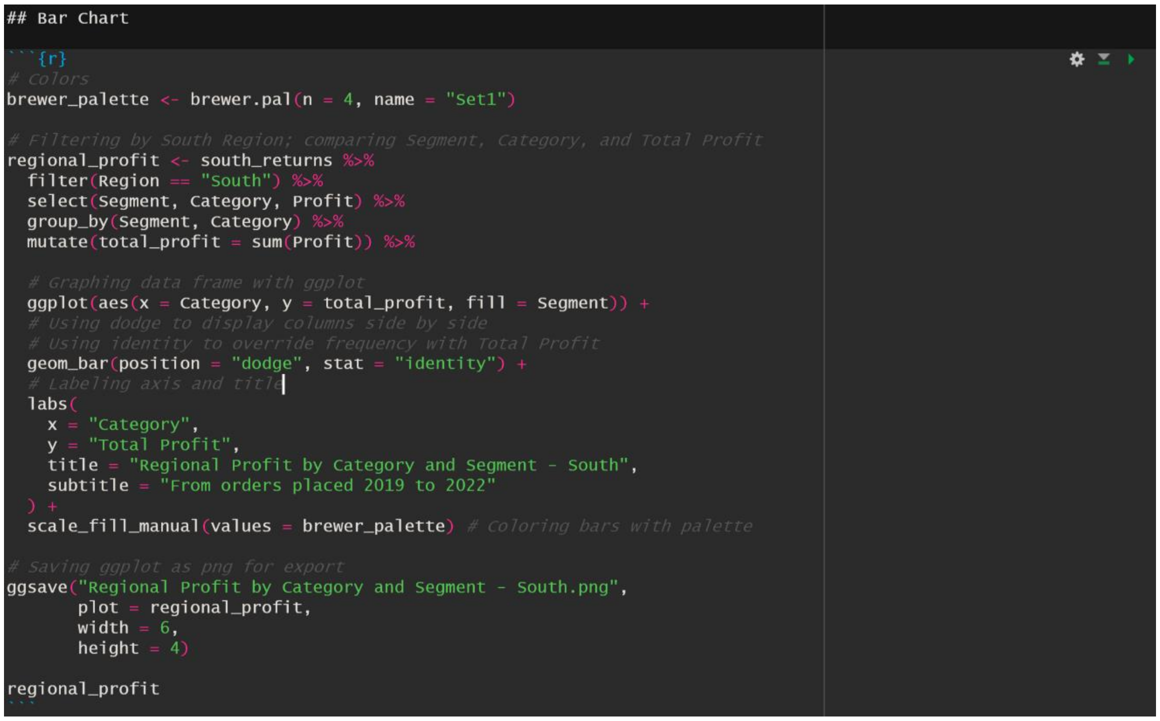
As Image 3 and Image 4 illustrate, the vcd mosaic function required a different kind of table than the one offered by base R. In order to create a vcd mosaic plot, I had to create a contingency table with xtabs(), which simply added a label to the axis of the table so that the mosaic() function could identify the components of the table. The mosaic function was a much more customizable experience than the mosaicplot() function from base R. In mosaic, you could edit the shading of the graph using the gp argument, offset variable names and labels, rotate variable names and labels, change the justification of variable names and labels, and change the font size of labels. P-values were also automatically calculated and displayed with this function. Because of these features the mosaic() function fulfilled all my needs, prompting me to move onto the next technical step of our analysis: bar charts.

Bar Charts

During a team meeting in which team members shared their visualizations and associated techniques, Nate presented a series of bar chart analysis comparing Total Profit to Category and Segment. The whole team thought the bar charts were well done, and therefore should be replicated within each of our regional analyses. Using Nate's visualizations as a reference, I created bar charts displaying the relationship between Total Profit, Category, and Segment and added a consistent color scheme to all my ggplot visualizations using the RColorBrewer package.

Image 5

Bar Chart Code

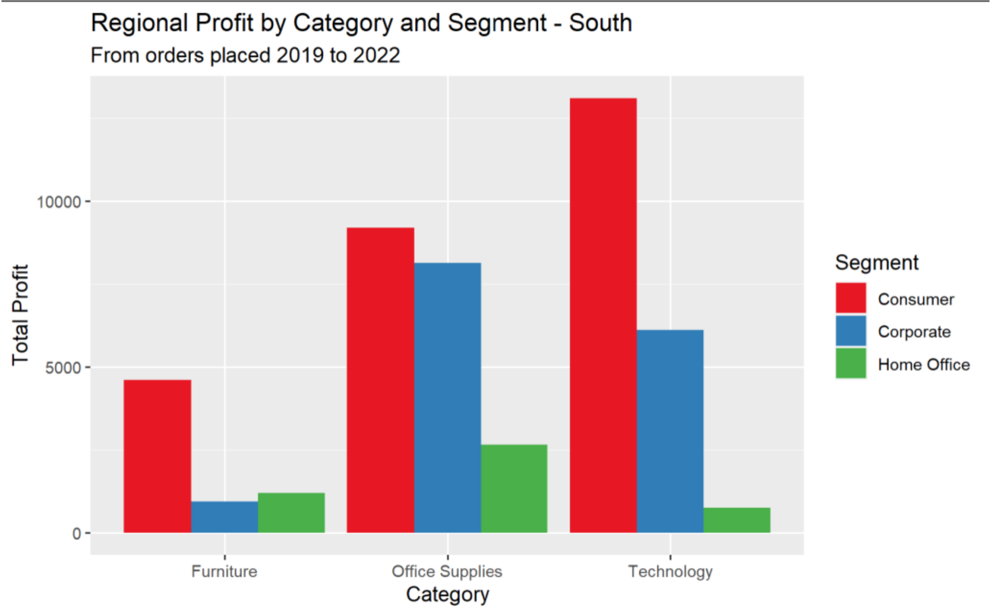


~~~
## Bar Chart
```{r}
# Colors
brewer_palette <- brewer.pal(n = 4, name = "Set1")

# Filtering by South Region; comparing Segment, Category, and Total Profit
regional_profit <- south_returns %>%
  filter(Region == "South") %>%
  select(Segment, Category, Profit) %>%
  group_by(Segment, Category) %>%
  mutate(total_profit = sum(Profit)) %>%

  # Graphing data frame with ggplot
  ggplot(aes(x = Category, y = total_profit, fill = Segment)) +
  # Using dodge to display columns side by side
  # Using identity to override frequency with Total Profit
  geom_bar(position = "dodge", stat = "identity") +
  # Labeling axis and title
  labs(
    x = "Category",
    y = "Total Profit",
    title = "Regional Profit by Category and Segment - South",
    subtitle = "From orders placed 2019 to 2022"
  ) +
  scale_fill_manual(values = brewer_palette) # Coloring bars with palette

# Saving ggplot as png for export
ggsave("Regional Profit by Category and Segment - South.png",
       plot = regional_profit,
       width = 6,
       height = 4)

regional_profit
```
~~~

Note. This image displays the code for the bar chart.

Image 6

Bar Chart



Note. This image displays a bar chart comparing southern Total Profit, Category, and Segment.

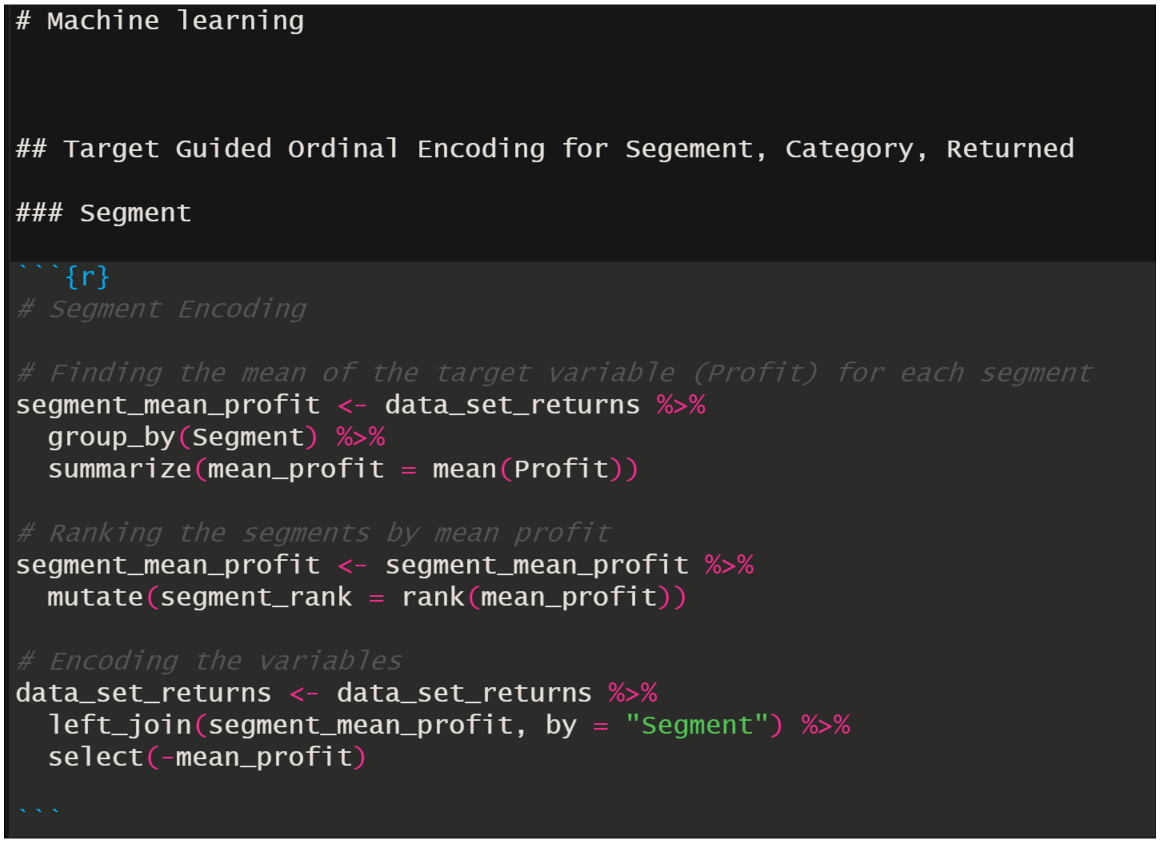
Using my knowledge of the dyplr package, as well as ggplot, I was able to filter the south_returns data set into a tibble of Segment, Category, and Total Profit for each of those groups. I was then able to plot this tibble using ggplot with a geom_bar layer. By using position = "dodge" and stat = "identity" as arguments within geom_bar, I successfully replaced Frequency along the y axis with Total Profit and separated the bars of the bar chart. With an informative visualization created, I moved onto developing a machine learning model.

Machine Learning Modeling with Random Forest

To predict total profit based on predictor variables of Category, Segment, and Returned Status, we developed a machine learning model using the Random Forest method. During another meeting with my peer mentor, I was advised to use a random forest regression method to train my model. Taking this advice, I found an online tutorial that explained what a random forest regression model was and provided some example code for its implementation in R. I learned that a random forest regression uses the mode of a "forest" of decision trees trained on different parts of the training to predict its target (Bhalla). With this knowledge, I began implementing the machine learning model, using a process called target guided ordinal encoding to rank the predictor variables by their mean total profit to associate a weight with each of the categorical variables.

Image 7

Target Guided Ordinal Encoding



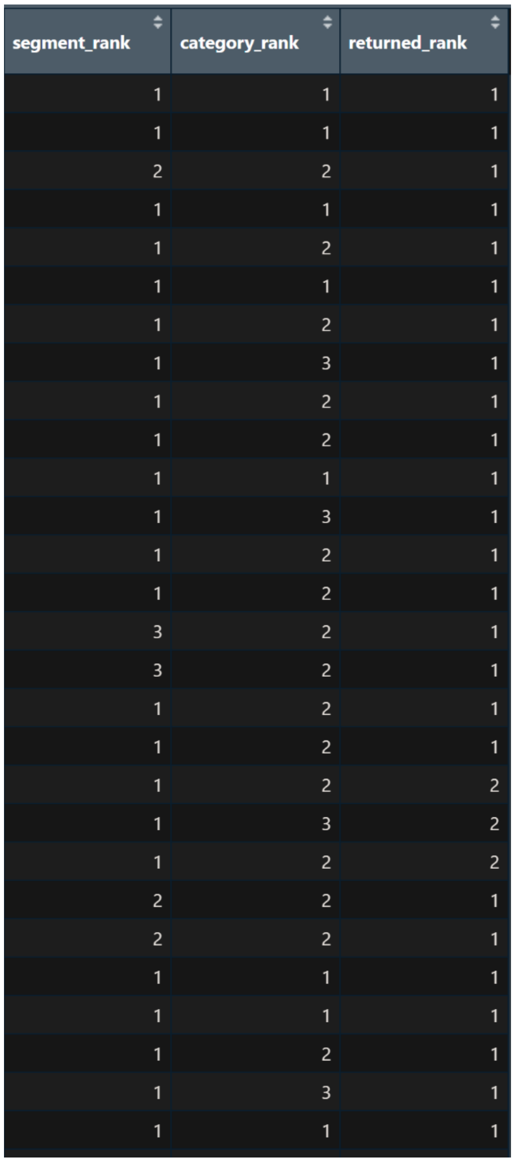
~~~
# Machine learning


## Target Guided Ordinal Encoding for Segement, Category, Returned

### Segment

```{r}
# Segment Encoding

# Finding the mean of the target variable (Profit) for each segment
segment_mean_profit <- data_set_returns %>%
  group_by(Segment) %>%
  summarize(mean_profit = mean(Profit))

# Ranking the segments by mean profit
segment_mean_profit <- segment_mean_profit %>%
  mutate(segment_rank = rank(mean_profit))

# Encoding the variables
data_set_returns <- data_set_returns %>%
  left_join(segment_mean_profit, by = "Segment") %>%
  select(-mean_profit)

```
~~~

Note. This image depicts the encoding process, which was repeated for Category and Returned.

Image 7 represents the encoding process for the predictor variable of Segment. The mean Total Profit of each segment is first calculated, then the segments are ranked based on the value of their mean Total Profits, finally being joined onto the overall data frame. This process was repeated for Category and Returned as well. The encoding of the categorical resulting in a modified data frame, partially shown in Image 8, that contained the ranks of each Segment, Category, and Returned status by their associated mean Total Profits.

Image 8

Target Guided Ordinal Encoding Ranks



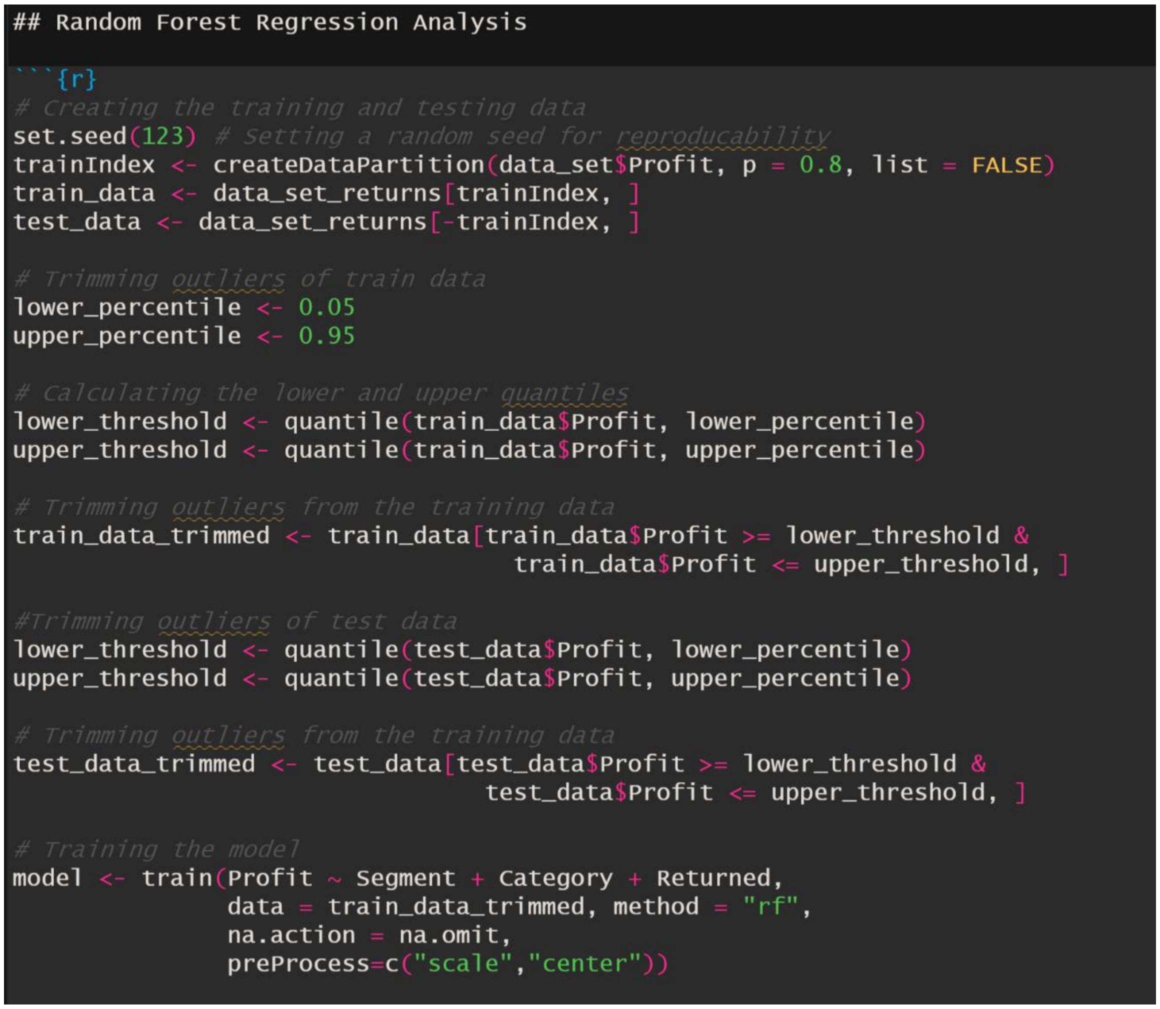
segment_rank	category_rank	returned_rank
1	1	1
1	1	1
2	2	1
1	1	1
1	2	1
1	1	1
1	2	1
1	3	1
1	2	1
1	2	1
1	1	1
1	3	1
1	2	1
1	2	1
3	2	1
3	2	1
1	2	1
1	2	1
1	2	2
1	3	2
1	2	2
2	2	1
2	2	1
1	1	1
1	1	1
1	2	1
1	3	1
1	1	1

Note. This image represents the encoded ranks of the categorical variables.

With the categorical variables encoded, all that remained was to create a training and testing data set for the model and build the machine learning model. I imported the caret and randomForest packages in order to perform my machine learning analysis, then using the createDataPartion() function to create a training set of 80% of the dataset. I then trimmed outliers from the dataset in an effort to improve accuracy, before feeding the training set into the model.

Image 9

Random Forest Regression Model Training



~~~
## Random Forest Regression Analysis

```{r}
# Creating the training and testing data
set.seed(123) # Setting a random seed for reproducability
trainIndex <- createDataPartition(data_set$Profit, p = 0.8, list = FALSE)
train_data <- data_set_returns[trainIndex, ]
test_data <- data_set_returns[-trainIndex, ]

# Trimming outliers of train data
lower_percentile <- 0.05
upper_percentile <- 0.95

# Calculating the lower and upper quantiles
lower_threshold <- quantile(train_data$Profit, lower_percentile)
upper_threshold <- quantile(train_data$Profit, upper_percentile)

# Trimming outliers from the training data
train_data_trimmed <- train_data[train_data$Profit >= lower_threshold &
                                  train_data$Profit <= upper_threshold, ]

#Trimming outliers of test data
lower_threshold <- quantile(test_data$Profit, lower_percentile)
upper_threshold <- quantile(test_data$Profit, upper_percentile)

# Trimming outliers from the training data
test_data_trimmed <- test_data[test_data$Profit >= lower_threshold &
                                test_data$Profit <= upper_threshold, ]

# Training the model
model <- train(Profit ~ Segment + Category + Returned,
               data = train_data_trimmed, method = "rf",
               na.action = na.omit,
               preProcess=c("scale","center"))
~~~

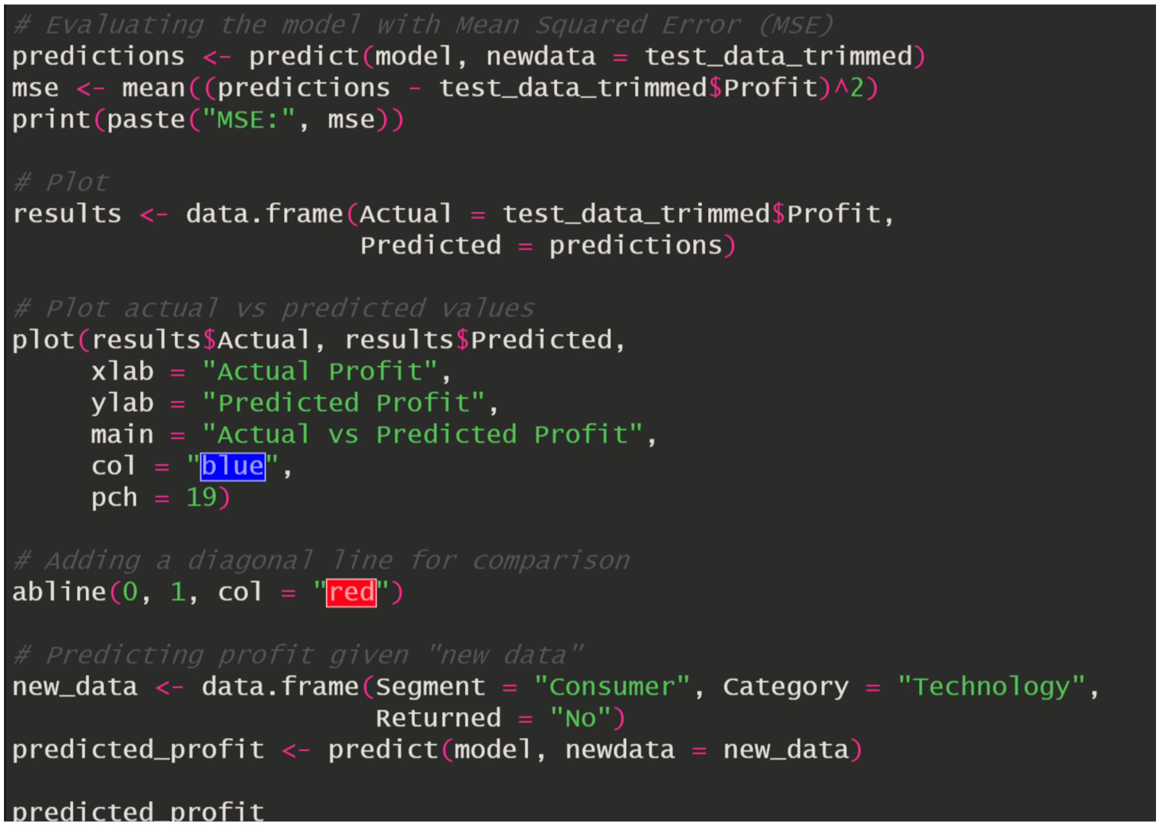
Note. This image depicts the preprocessing techniques used before training the model.

As demonstrated in Image 9, the lower and upper 5% of data are trimmed off to reduce outliers, and trained data is pre-processed using the preprocess argument within the train() function. The model is trained using random forest, shown as “rf”, and is assigned a target variable of Profit based on Segment, Category, and Returned.

Following the training of the model, I tested the model’s predictions against the test data, graphing the accuracy of the model on a scatterplot and recording its MSE.

Image 10

Random Forest Regression Testing



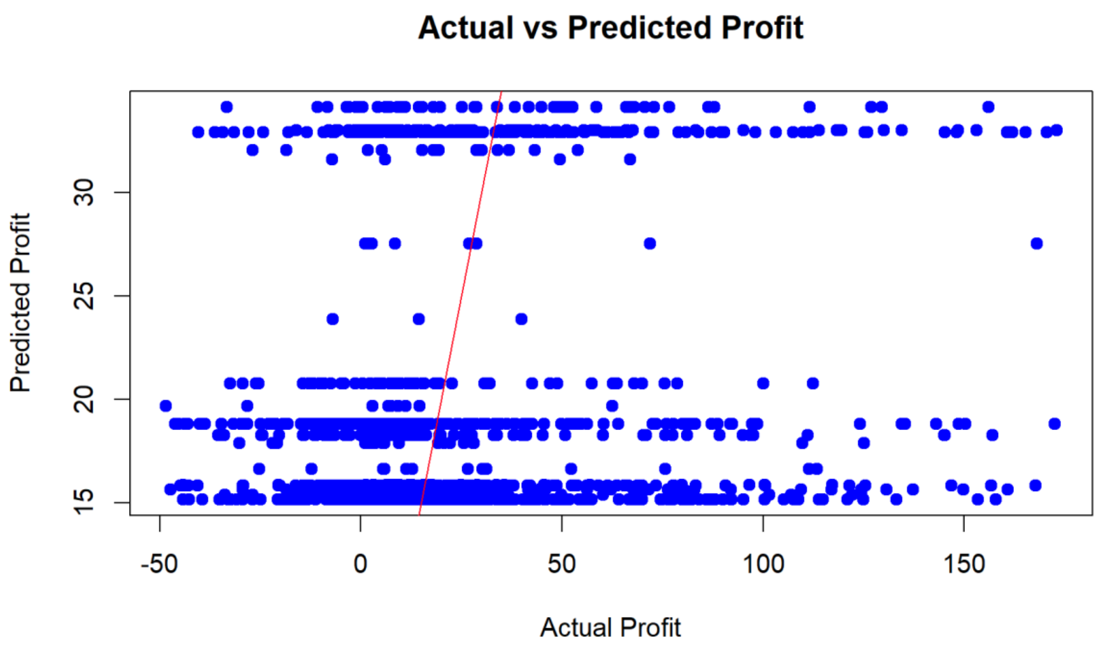
```
# Evaluating the model with Mean Squared Error (MSE)
predictions <- predict(model, newdata = test_data_trimmed)
mse <- mean((predictions - test_data_trimmed$Profit)^2)
print(paste("MSE:", mse))

# Plot
results <- data.frame(Actual = test_data_trimmed$Profit,
                      Predicted = predictions)

# Plot actual vs predicted values
plot(results$Actual, results$Predicted,
     xlab = "Actual Profit",
     ylab = "Predicted Profit",
     main = "Actual vs Predicted Profit",
     col = "blue",
     pch = 19)

# Adding a diagonal line for comparison
abline(0, 1, col = "red")

# Predicting profit given "new data"
new_data <- data.frame(Segment = "Consumer", Category = "Technology",
                       Returned = "No")
predicted_profit <- predict(model, newdata = new_data)

predicted_profit
```

Note. This image portrays the model testing process.

By using the predict function, I was able to make a series of predictions using the test data as a reference. I printed out the mean squared error of the model to evaluate its accuracy, before assigning the predictions of the model and the actual values of the test data to a data frame. By graphing this data frame, I was able to visualize how accurate the model was at predicting profit.

Image 11

Random Forest Regression Accuracy



Note. This image depicts a scatter plot of the model's predicted values against the actual profits of the test set. The red line symbolizes correct guesses.

Image 11 was accompanied by an MSE: of 1080.95, which is very high. The points distance from the red line symbolizing correct guesses and the large MSE indicates that the machine learning model is highly inaccurate. Our team believes that this is due to the presence of few predictor variables, and the high number of categorical variables we used in our analysis that didn't work well with the machine learning model we trained.

The completion of the machine learning model marked the end of my technical skills journey on this project, prompting me to consider the results of my analyses and relate them to our study's overall hypotheses.

Results

My techniques directly contributed to the testing of both our two hypotheses, which focused on the relationship between Total Profit, Category, and Segment, as well as the relationship of returns to locational information. The hypotheses were:

1. There exists a correlation between product performance (measured by profit), the consumer segment, and product category within the superstore.
2. Product returns are correlated with the Region, State, Segment, Category and can be used to predict product performance in a specific location by pinpointing areas with high or low returns.

Without the mosaics that I learned and taught the group how to construct, we would not have been to use the residuals of returns to pinpoint areas of focus for the supermarket. Additionally, the associated Chi Squared testing within these mosaics displayed the correlation between returns and the locational data that we were comparing returns to, enabling us to draw connections between some of the variables we hypothesized returns would be connected to.

The bar charts that I helped implement were also extremely useful in determining the relationship between product performance, consumer segment, and product category as they enabled us to get an in depth look at the interactions between these variables across regions and states.

Finally, the machine learning model that I spearheaded the development of did not contribute significantly to the testing of our hypotheses, but it was an educational experience that I enjoyed learning from.

Bibliography

Bhalla, D. (n.d.). *A complete guide to Random Forest in R*. ListenData. https://www.listendata.com/2014/11/random-forest-with-r.html

Friendly, M. (2023, August 21). *Permuting variable levels*. R-Packages. https://cran.r-project.org/web/packages/vcdExtra/vignettes/mosaics.html

Helsloot, R. (2020, November 8). *Change the size of labels in mosaic function, R*. Stack Overflow. https://stackoverflow.com/questions/61579713/change-the-size-of-labels-in-mosaic-function-r

Mudadla, S. (2023, April 27). *Target guided ordinal Encoding with Example*. Medium. https://medium.com/@sujathamudadla1213/target-guided-ordinal-encoding-with-example-450323fea78e

STHDA. (n.d.). *Chi-Square Test of Independence in R - Easy Guides - Wiki - STHDA*. Www.sthda.com. http://www.sthda.com/english/wiki/chi-square-test-of-independence-in-r