OPIM 407

Case I

Selin Ceydeli

Sinan Yılmaz

Göktuğ Oktaylar

03.04.2023

A. INTRODUCTION: Introducing the company, the primary objective of this study and the report

Carpet industry of the world has been dominated by the exports of the Indian market approximately by 40 percent. Carpet production in India helps 2 million people's employment as shown by research in 2018. Champo Carpets is one of the carpet manufacturing companies from Bhadohi, India. The company scattered all around the region approximately 1000 square kilometers where the workers from many villages and towns put effort. In 2020, Champo Carpets had employed 1500 workers. Moreover, the company has strength of producing 200.000 carpets per month in which the products are licensed and certified by many leading companies. Carpet manufacturing process is a difficult process in which it involves many disciplines to work together. As mentioned in the case study, manufacturing process involves design, raw material management, dyeing, weaving, or hand-tufting, finalizing, and quality control.

Champo Carpets has many carpet types to produce in which it takes 3 weeks to 3 months to produce a specific carpet according to its design and difficulty. There are mainly four types of carpet types that Champo Carpets produce. Hand-tufted carpet is the first one in which it needs the least effort and is the cheapest one. The second one is hand-knotted carpet where production requires a lot of skills which makes the product expensive. The third and fourth carpet types are consecutively Kilims and Durries.

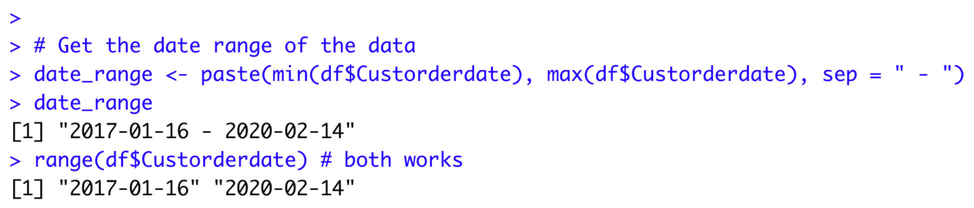
Champo Carpets' sales and marketing tactic involves creating a sample product to send to their clients, potential clients and to show in fairgrounds. However, the sample production is very costly as its doubles or triples the production cost in every step of the manufacturing process (Exhibit 2). Champo Carpets is trying to find an efficient way to create its sample products/designs where they can profit. The company is collecting data of the production since 2017. They would like to use carpet attributes for the creation of customer segments and building a recommendation system for their clients.

In section B of the report, we will briefly introduce the data to the reader. In section C, we will explain the results of the exploratory data analysis we have conducted with the R programming language. The aim this section will be to analyze and investigate the data to understand the main characteristics of the variables. In section D, we will provide collaborative filtering-based recommendations for the samples. In section E, we will implement customer segmentation using clustering. Lastly, in section F, we will present our conclusions summarizing the main findings of the case analysis.

B. BRIEF INTRODUCTION TO DATA

2.1 The Range, Number of Variables, and Observations

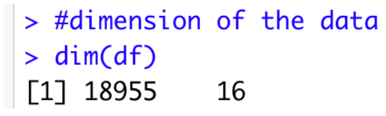
Champo Carpets has established a data analysis system in its manufacturing and selling system to visualize its data in a computer environment. Champo Carpets aims to lower the costs of the sample carpet production and personalize the samples according to the preferences of its old customers and future customers based on the collected data. Figure 2.1.1 displays an R code that shows the collected data containing all the orders from 2017 to 2020:



```
>
> # Get the date range of the data
> date_range <- paste(min(df$Custorderdate), max(df$Custorderdate), sep = " - ")
> date_range
[1] "2017-01-16 - 2020-02-14"
> range(df$Custorderdate) # both works
[1] "2017-01-16" "2020-02-14"
```

Figure 2.1.1. Range of the Data

From 2017-01-06 to 2020-02-14, Champo Carpets had recorded 18955 transactions and every transaction contains 16 different details about the transaction (Figure 2.1.2). Order Type, Unit Name and Color are instances of the 16 details of the transaction. Based on these features, in the following chapters, a detailed analysis will be conducted.



```
> #dimension of the data
> dim(df)
[1] 18955    16
```

Figure 2.1.2. Number of Variables and Observations

2.2 New List of Attributes in the Data

Table 1 below displays a list of the attributes in the data, specifying each feature and describing its type and measurement unit.

Table 1. Revised Version of Exhibit 4 from the Case Study

Feature	Description	Feature Type	Measurement Unit
Order Type	Area-wise or piece-wise	Categorical	Area-wise or piecewise
Order Category	Order or sample	Categorical	Order or sample
Customer Code	45 Customers (B2B customers)	Identifier	H-1, M-1, E-2, …

Country Name	Originating country of customer	Categorical	USA, CANADA, INDIA, …
Customer Order ID	Order identification number	Identifier	12612, 13001, SAMPLE_21-JUN, …
Customer Order Date	Date on which order was placed	Date	From 2017-01-06 to 2020-02-14
Unit Name	Unit of measurement	Categorical	feet, meter, inch, piece, and weight
Quantity Required	Number of units ordered	Quantitative	Min: 1 – Max: 6400
Total Area	Carpet area	Quantitative	Min: 0,04 – Max: 1024 in feet-squared
Amount	Revenue generated from the carpet sales	Quantitative	Min: 0 – Max: 599719.68 in USD
Item Name	Name of the items produced	Categorical	Hand-tufted, Durry, Knotted, ...
Quality Name	Quality of material used	Categorical	TUFTED 60C+VISC 2/16 5PLY, D.B. LEFA VISCOSE, …
Design Name	Code assigned to describe the design	Categorical	FLEUR TRELLIS, SEVILLA FLORAL, PLAIN, HOMER, …
Color Name	Color of the carpet	Categorical	SILVER, GREY, BEIGE, NAVY, …
Shape Name	Shape of the carpet	Categorical	Rectangular (REC), square, round, octogen and oval
Total Area (AreaFt)	Total carpet area ordered	Quantitative	Min: 0.4444 – Max: 645.7222 in feet-squared

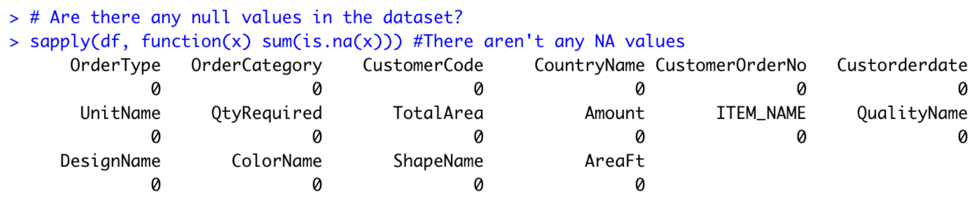
C. EXPLORATORY DATA ANALYSIS

3.1 The Dataset: Raw Data-Order and Sample

Champo Carpets has been among the few companies who have implemented an Enterprise Resource Planning (ERP) system for efficient collection of business data. Over the years, data has been accumulated from ERP operations showing several attributes related to the carpets the company is selling. The careful analysis of the data is crucial for identifying the most important customers and products.

We have decided to conduct our exploratory data analysis on the combined dataset, named "Raw Data-Order and Sample", which is the first sheet in the excel file provided in the case study.

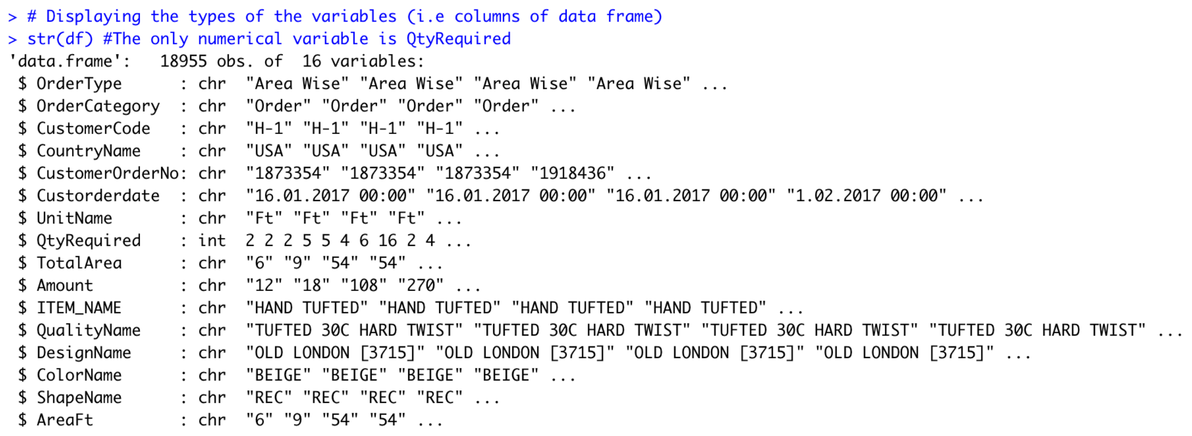
Before making analysis on the dataset, the missing values should be handled. Therefore, we checked if there exist any missing values in our dataset using R's is.na() function and as can be seen in Figure 3.1.1, we observed that there aren't any missing values in either of the columns:



```
> # Are there any null values in the dataset?
> sapply(df, function(x) sum(is.na(x))) #There aren't any NA values
      OrderType   OrderCategory    CustomerCode     CountryName CustomerOrderNo    Custorderdate
              0               0               0               0               0                0
       UnitName     QtyRequired       TotalArea          Amount       ITEM_NAME      QualityName
              0               0               0               0               0                0
     DesignName       ColorName       ShapeName          AreaFt
              0               0               0               0
```

Figure 3.1.1. Missing Value Analysis

Moreover, we checked the datatype of the entire dataset as well as the datatype of each column using R's str() function, whose result is displayed in Figure 3.1.2:

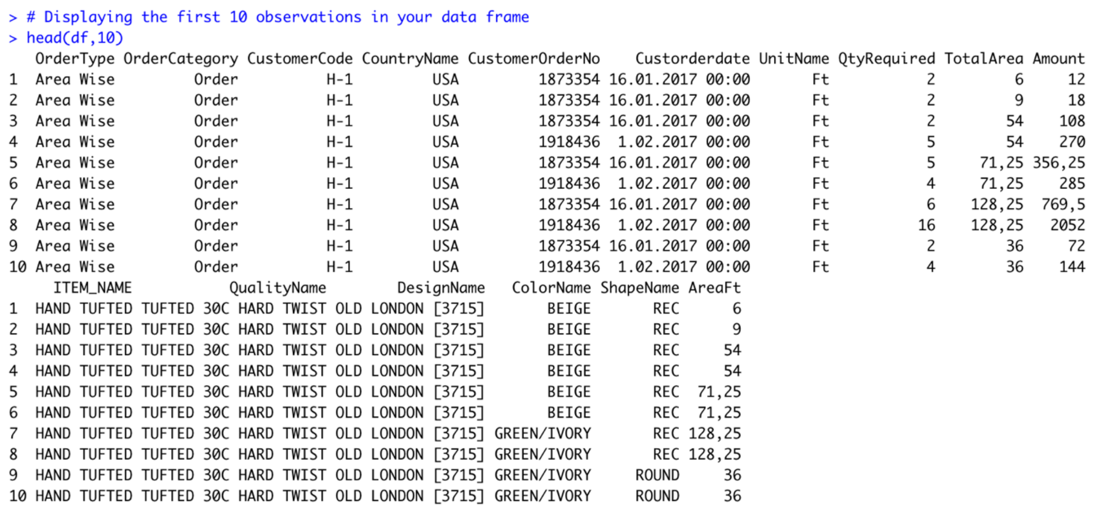


```
> # Displaying the types of the variables (i.e columns of data frame)
> str(df) #The only numerical variable is QtyRequired
'data.frame':	18955 obs. of  16 variables:
 $ OrderType      : chr  "Area Wise" "Area Wise" "Area Wise" "Area Wise" ...
 $ OrderCategory  : chr  "Order" "Order" "Order" "Order" ...
 $ CustomerCode   : chr  "H-1" "H-1" "H-1" "H-1" ...
 $ CountryName    : chr  "USA" "USA" "USA" "USA" ...
 $ CustomerOrderNo: chr  "1873354" "1873354" "1873354" "1918436" ...
 $ Custorderdate  : chr  "16.01.2017 00:00" "16.01.2017 00:00" "16.01.2017 00:00" "1.02.2017 00:00" ...
 $ UnitName       : chr  "Ft" "Ft" "Ft" "Ft" ...
 $ QtyRequired    : int  2 2 2 5 5 4 6 16 2 4 ...
 $ TotalArea      : chr  "6" "9" "54" "54" ...
 $ Amount         : chr  "12" "18" "108" "270" ...
 $ ITEM_NAME      : chr  "HAND TUFTED" "HAND TUFTED" "HAND TUFTED" "HAND TUFTED" ...
 $ QualityName    : chr  "TUFTED 30C HARD TWIST" "TUFTED 30C HARD TWIST" "TUFTED 30C HARD TWIST" "TUFTED 30C HARD TWIST" ...
 $ DesignName     : chr  "OLD LONDON [3715]" "OLD LONDON [3715]" "OLD LONDON [3715]" "OLD LONDON [3715]" ...
 $ ColorName      : chr  "BEIGE" "BEIGE" "BEIGE" "BEIGE" ...
 $ ShapeName      : chr  "REC" "REC" "REC" "REC" ...
 $ AreaFt         : chr  "6" "9" "54" "54" ...
```

Figure 3.1.2. Displaying the Variable Types

It is observed in Figure 3.1.2 that the only numerical variable in the dataset is the QtyRequired variable, which is of type integer. The remaining variables are categorical as they are of type character.

Lastly, we looked at the first 10 rows of the dataset to understand how data is stored in the dataset using R's head() function, whose result is displayed in Figure 3.1.3:

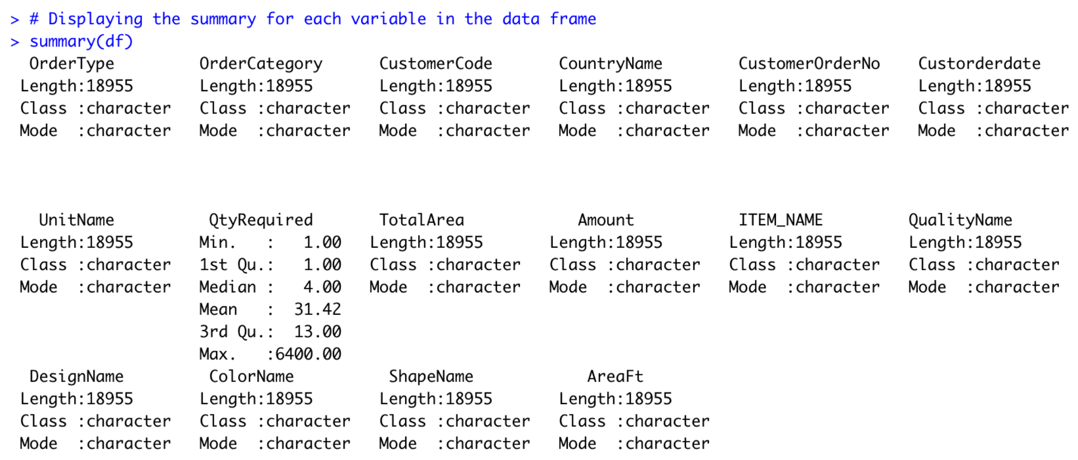


```
> # Displaying the first 10 observations in your data frame
> head(df,10)
   OrderType OrderCategory CustomerCode CountryName CustomerOrderNo    Custorderdate UnitName QtyRequired TotalArea Amount
1  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           2         6     12
2  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           2         9     18
3  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           2        54    108
4  Area Wise         Order          H-1         USA         1918436  1.02.2017 00:00       Ft           5        54    270
5  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           5     71,25 356,25
6  Area Wise         Order          H-1         USA         1918436  1.02.2017 00:00       Ft           4     71,25    285
7  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           6    128,25  769,5
8  Area Wise         Order          H-1         USA         1918436  1.02.2017 00:00       Ft          16    128,25   2052
9  Area Wise         Order          H-1         USA         1873354 16.01.2017 00:00       Ft           2        36     72
10 Area Wise         Order          H-1         USA         1918436  1.02.2017 00:00       Ft           4        36    144
     ITEM_NAME           QualityName        DesignName   ColorName ShapeName AreaFt
1  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC      6
2  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC      9
3  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC     54
4  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC     54
5  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC  71,25
6  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715]       BEIGE       REC  71,25
7  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715] GREEN/IVORY       REC 128,25
8  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715] GREEN/IVORY       REC 128,25
9  HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715] GREEN/IVORY     ROUND     36
10 HAND TUFTED TUFTED 30C HARD TWIST OLD LONDON [3715] GREEN/IVORY     ROUND     36
```

Figure 3.1.3. Displaying the First 10 Observations

3.2 Descriptive Statistics

In Figure 3.2.1, the measures of location for the dataset are described using R's summary() function:

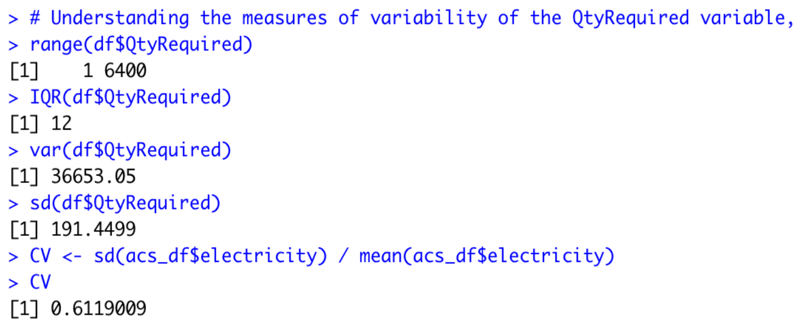


```
> # Displaying the summary for each variable in the data frame
> summary(df)
  OrderType          OrderCategory      CustomerCode       CountryName        CustomerOrderNo    Custorderdate     
 Length:18955       Length:18955       Length:18955       Length:18955       Length:18955       Length:18955      
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character  



   UnitName          QtyRequired         TotalArea            Amount            ITEM_NAME          QualityName       
 Length:18955       Min.   :   1.00   Length:18955       Length:18955       Length:18955       Length:18955      
 Class :character   1st Qu.:   1.00   Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Median :   4.00   Mode  :character   Mode  :character   Mode  :character   Mode  :character  
                    Mean   :  31.42                                                                               
                    3rd Qu.:  13.00                                                                               
                    Max.   :6400.00                                                                               
  DesignName          ColorName           ShapeName            AreaFt         
 Length:18955       Length:18955       Length:18955       Length:18955      
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
```

Figure 3.2.1. Summary Statistics of the Dataset

Since the only numerical variable in the dataset is quantity required, only that column's measures of location values are calculated. For the remaining variables, their length and class information are printed, stating that they are of type character.

Moreover, to describe how spread or varied the quantity required variable is, the measures of dispersion are calculated. These measures are used to complement the measures of central tendency in summarizing the dataset. The results of the measures of dispersion are calculated in R and displayed in Figure 3.2.2:



```
> # Understanding the measures of variability of the QtyRequired variable,
> range(df$QtyRequired)
[1]    1 6400
> IQR(df$QtyRequired)
[1] 12
> var(df$QtyRequired)
[1] 36653.05
> sd(df$QtyRequired)
[1] 191.4499
> CV <- sd(acs_df$electricity) / mean(acs_df$electricity)
> CV
[1] 0.6119009
```

Figure 3.2.2. Measures of Dispersion

To comment of the variability of the quantity required variable, commenting on the coefficient of variation (CV) calculated at the last step in the R code in Figure 3.2.2 is important. Noting that CV is a measure of relative variability that can be used to compare the variation between datasets with different units, a coefficient of variation (CV) of 0.6119009 for the quantity required variable indicates that the standard deviation is quite high relative to the mean.

3.3 Data Visualizations

Bar charts can be used to visualize the distribution of categorical variables, showing their frequency in the dataset. In Figure 3.3.1, the bar chart displaying the distribution of countries as customers of Champo Carpets is demonstrated:

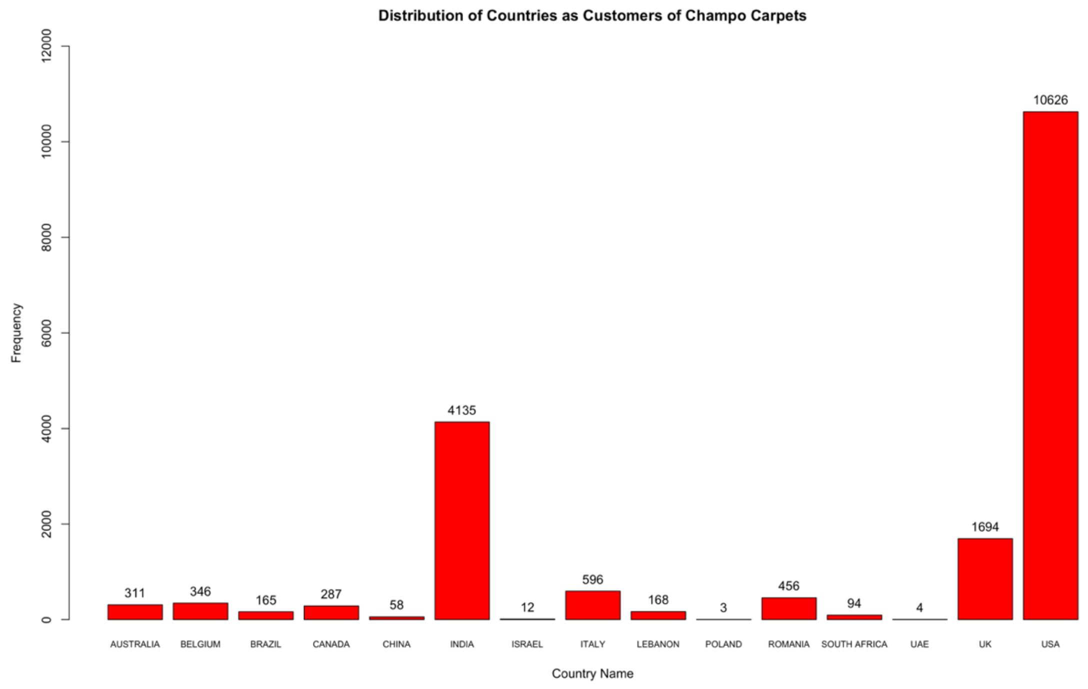


Figure 3.3.1. Bar Chart of Country Names

To understand the distribution of Countries with respect to the sales made and read the bar chart demonstrated in Figure 3.3.1 better, we checked the percent frequency distribution of country names in the dataset and the results are presented in Figure 3.3.2:

```
> #Checking the percent frequency of the Countries
> 100*table(df$CountryName)/length(df$CountryName)

  AUSTRALIA      BELGIUM       BRAZIL       CANADA        CHINA        INDIA       ISRAEL        ITALY      LEBANON 
 1.64072804   1.82537589   0.87048272   1.51411237   0.30598787  21.81482458   0.06330783   3.14428911   0.88630968 
     POLAND      ROMANIA SOUTH AFRICA          UAE           UK          USA 
 0.01582696   2.40569771   0.49591137   0.02110261   8.93695595  56.05908731 
```

Figure 3.3.2. Percent Frequency Distribution of Country Names

As shown in the percent frequency distribution table in Figure 3.3.2, 56.1% of the purchases were made by the USA. Following it, 21.8% of the purchases were made in the domestic market, by India. As the last remarkable purchase, UK had made the 8.9% of the purchases. The rest of the purchases are less than 3%. Therefore, it can be concluded that the distribution of purchases around the countries is highly skewed. Champo Carpets mostly does business with three countries: USA, India, and UK.

To further understand the distribution of purchases made by each country, we have drawn a stacked bar chart, showing the total quantity produced for each country. The bar chart is segmented and colored based on the number of items produced. This stacked bar chart is displayed in Figure 3.3.3:

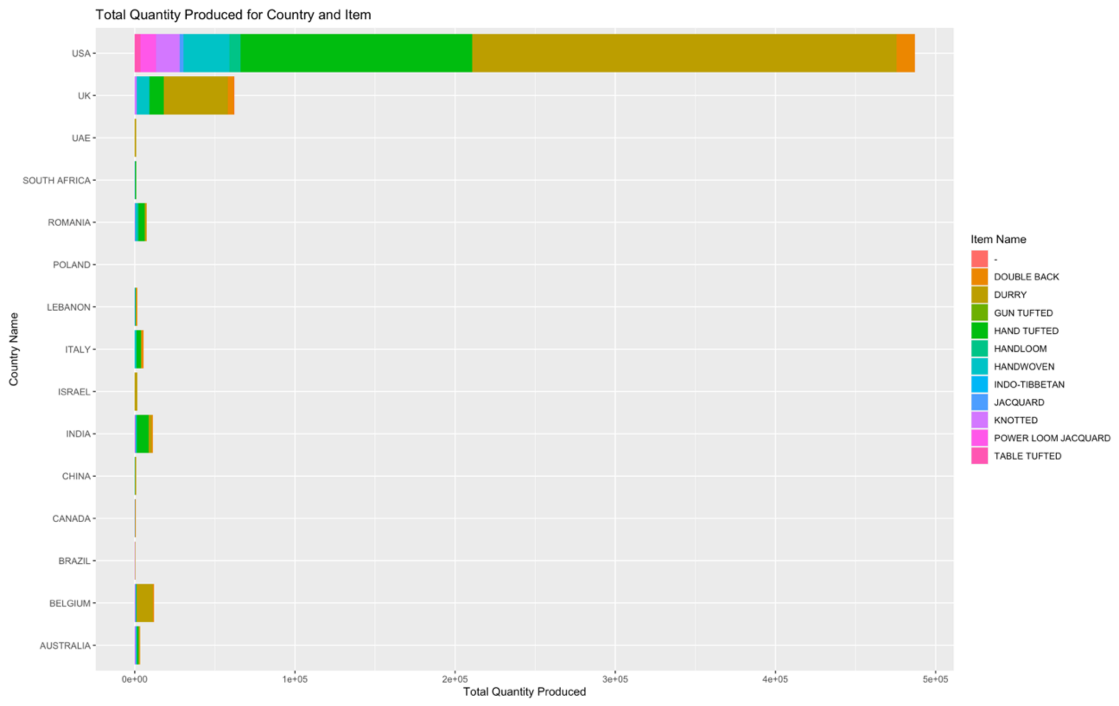


Figure 3.3.3. Stacked Bar Chart Showing the Total Quantity Produced for Each Country

The stacked bar chart in Figure 3.3.3 shows the total quantity of each item produced for each country. Each bar represents a country, and the height of each segment within the bar represents the quantity of a particular item produced for that country. The fill color of each segment represents the name of the item. This visualization helps to identify which items are produced more for which country and thus, which items have high demand in certain countries.

An interesting observation is that comparing Figure 3.3.1 and Figure 3.3.3, we can conclude that even though India is the second largest purchaser of the products of Champo Carpets in terms of number of purchases made, the total quantity produced for Indian market is not comparable with the USA nor with the UK, as shown in Figure 3.3.3.

Likewise, a second stacked bar chart is drawn showing the total quantity of each item produced for each customer, as shown in Figure 3.3.4:

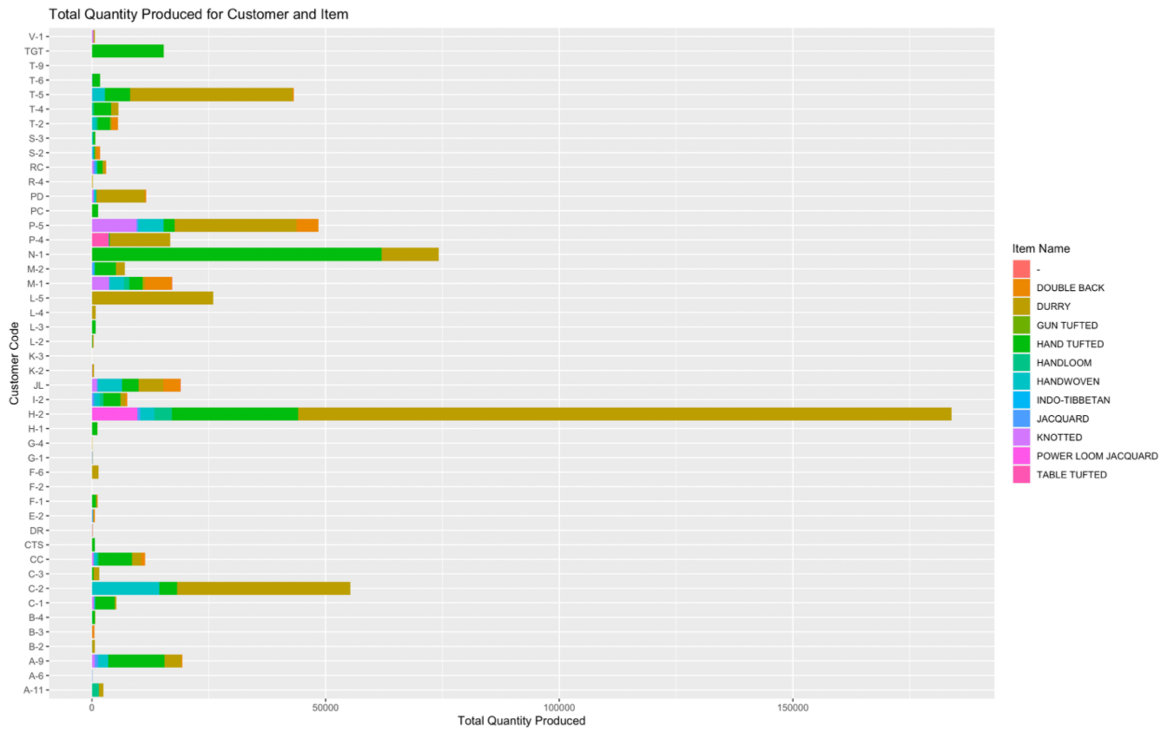


Figure 3.3.4. Stacked Bar Chart Showing the Total Quantity Produced for Each Customer

Figure 3.3.4 shows the total quantity produced for each customer for each item, with each item represented by a different color in the stacked bar. Looking at this bar chart, Champo Carpets can identify its most important customers with respect to the total quantity produced for them.

We continued with our data visualizations by drawing box plots to summarize the distribution of the quantity required variable, which is a numerical variable, for each category of ITEM_NAME variable. Figure 3.3.5 displays the initial version of the box plot before reducing the range of the quantity required variable, i.e., before eliminating some of the outliers:

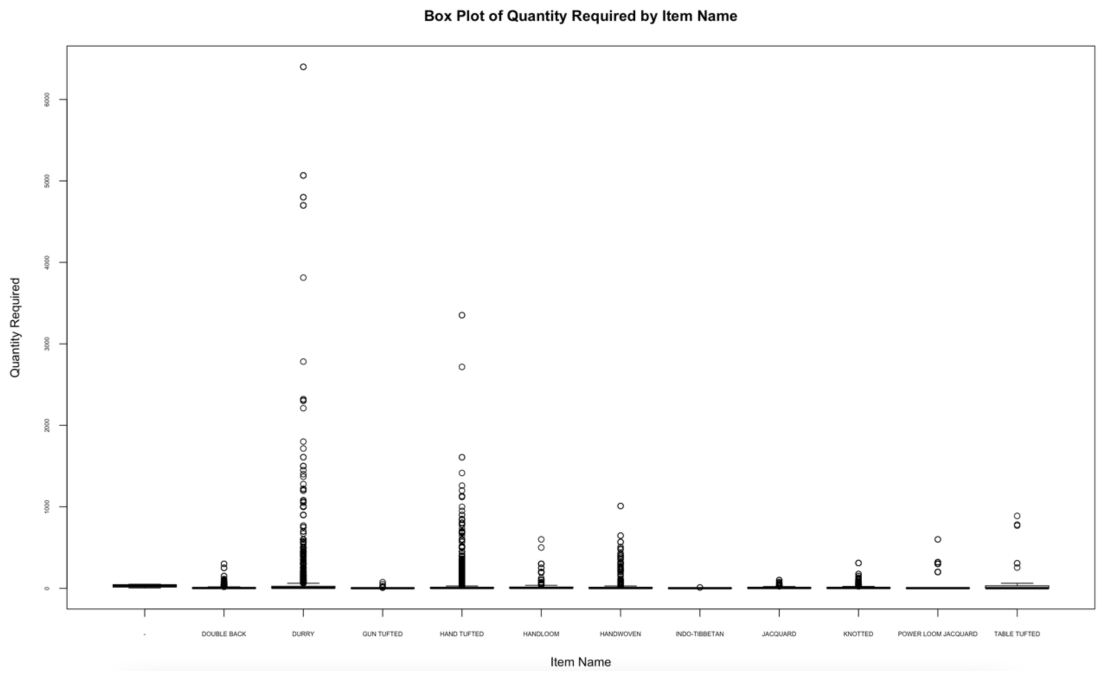


Figure 3.3.5. Box Plot of Quantity Required by Item Name Original Version

In the box plot in Figure 3.3.5, it is seen that there are data points that deviate significantly from the rest of the data, as shown with black dots. Outliers is one type of anomaly that can occur in a dataset and for the quantity required variable, outliers exist for each item type. However, it is possible that such large quantities are produced for a particular item if a huge purchase is made. Therefore, we cannot confidently name these outliers as anomalies without making sure that these data refer to any unexpected or unusual observations. Regardless, the existence of the outliers prevents us from demonstrating the general shape of the distribution using the box plot.

Since we wanted to identify the shape, central tendency, and spread of the quantity required variable for each type of item produced by the Champo Carpets, we decided to eliminate the majority of the outliers and revised the plot as seen in Figure 3.3.6:

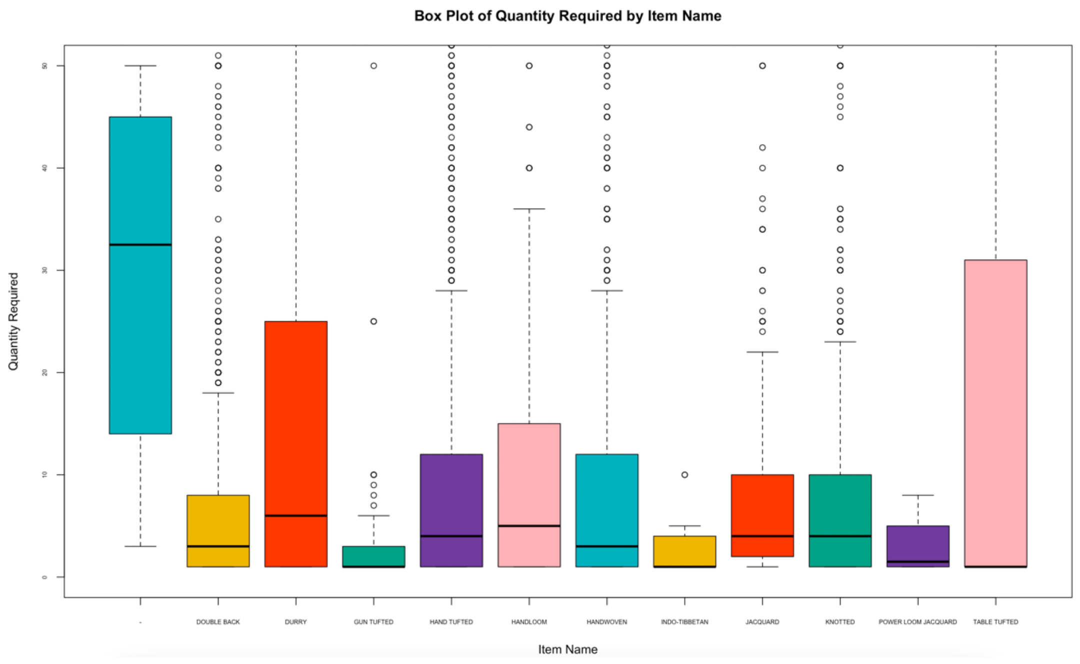


Figure 3.3.6. Box Plot of Quantity Required by Item Name Revised Version

With the revisions made in Figure 3.3.6, the ability of the box plot to demonstrate the range, the median, the interquartile range, the skewness of the quantity required variable with respect to each type of item is improved greatly. Only the ability of the box plot to show the presence of outliers is reduced. Nevertheless, these data points that are significantly different from the rest of the data are shown in the original version, in Figure 3.3.5.

For understanding which type of item is produced the most and which is produced the least, looking at the center of the box plots and thus, comparing the medians of the data would be logical. It is seen in Figure 3.3.6 that the item Durry has the largest median for the quantity required and Champo Carpets is most likely to get orders for this item whereas the items Gun Tufted and Table Tufted have the smallest medians, which signifies that the company is least likely to get orders for these items.

A similar observation is made for the box plots drawn to summarize the distribution of the quantity required variable for each category of the shape name variable, as shown in Figure 3.3.7:

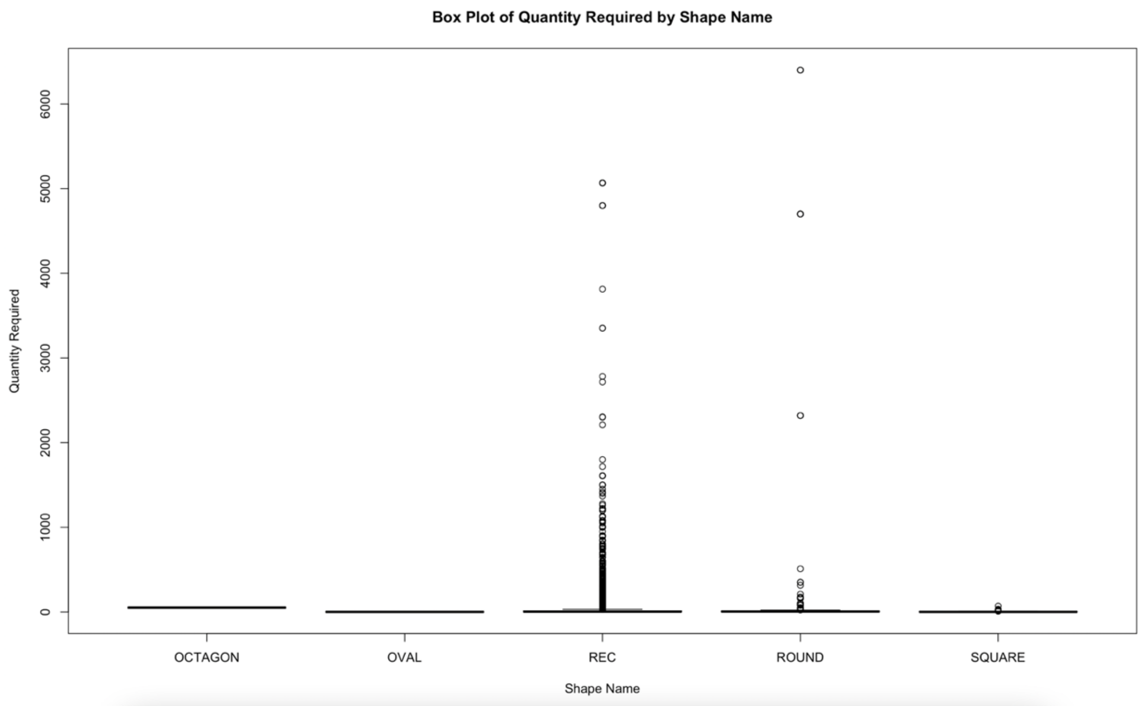


Figure 3.3.7. Box Plot of Quantity Required by Shape Name Original Version

The revised version of the box plot with the eliminated outliers is shown in Figure 3.3.8:

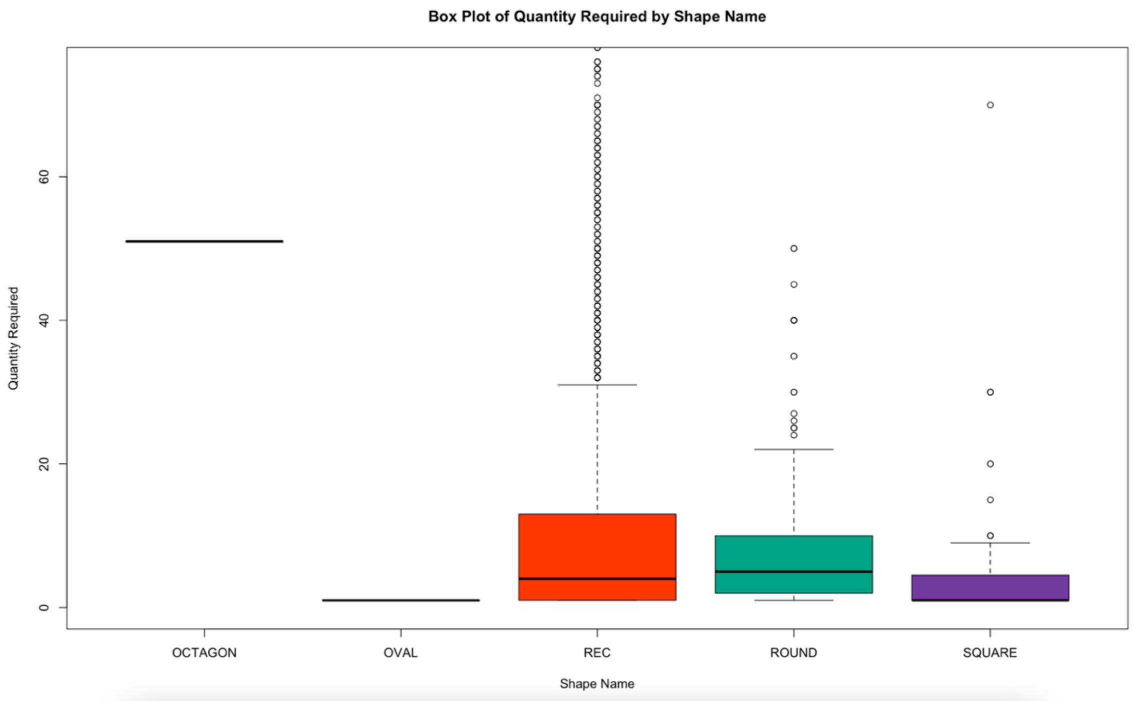


Figure 3.3.8. Box Plot of Quantity Required by Shape Name Revised Version

The box plot in Figure 3.3.8 displays that for octagon and oval shapes, only a handful of purchases are made. That's why the distributions are not observable. Comparing the centers of the box plots, we can conclude that the round carpet purchases are made the most, which is followed by rectangular and square carpets. This box plot is important as it displays the characteristics of the more desired carpets in terms of their shapes.

To further investigate the characteristics of the desired carpets and more popular orders, we decided to draw a pie chart for the order type to show the count of each category of the order type variable. The pie chart is shown in Figure 3.3.9:

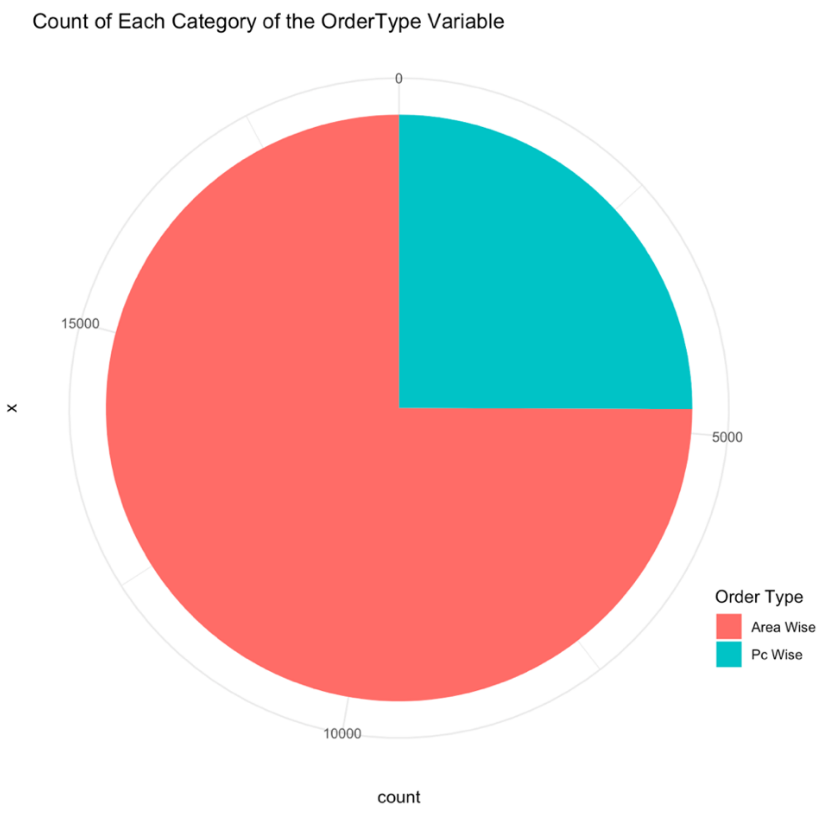


Figure 3.3.9. Pie Chart Showing the Count of Order Types

It is observed from the pie chart in Figure 3.3.9 that pc-wise orders were made three times more than the area-wise orders.

Another important feature of the orders that we needed to investigate was whether the carpets were prepared as orders or as samples. Understanding this distribution was important especially because Champo Carpets was complainant about low conversion rates and the cost of preparing samples. Therefore, we decided to draw a pie chart for the order category to show the count of each category of the order category variable. The pie chart is shown in Figure 3.3.10:

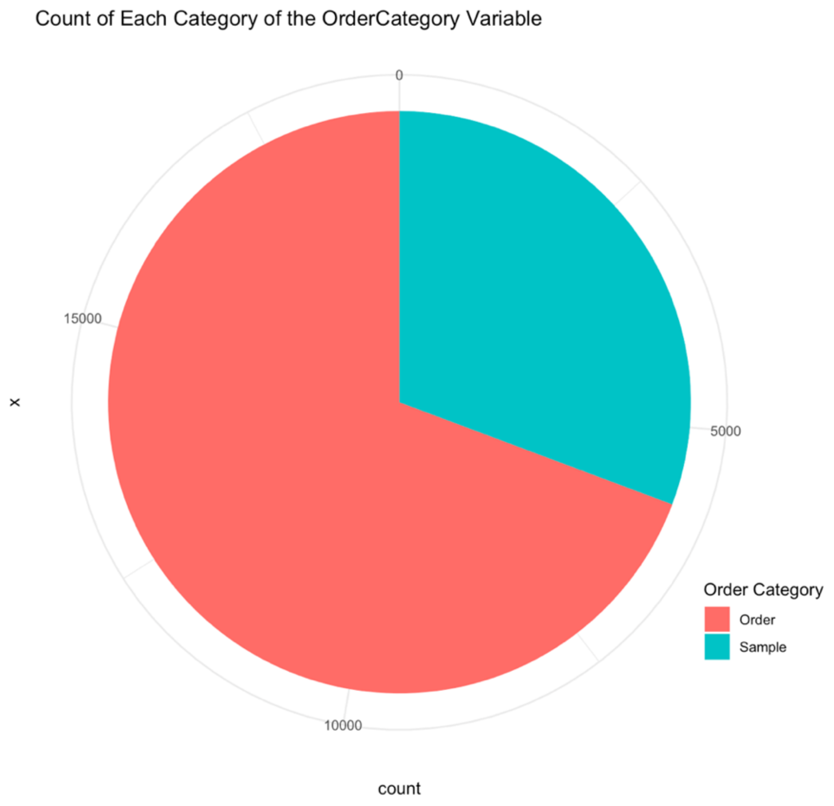


Figure 3.3.10. Pie Chart Showing the Count of Order Categories

It is seen from Figure 3.3.10 that a remarkable percentage of the orders, almost 30% of all orders, are prepared as samples. Considering that these samples do not generate profit to the company as they are not sold, this high percentage signals a cost burden to the Champo Carpets.

D. COLLABORATIVE FILTERING-BASED RECOMMENDATIONS

4.1 Champo's Traditional Recommendation System

Sample strategy is beneficial in many aspects, sending samples to fairs introduces the products of Champo Carpets to a great extent. It also helps to sell products to former customers. However, this strategy could end up being extremely costly as sampling doubles the weaving costs, triples the dyeing costs, and increases the finishing costs by 2.5 times. Hence, Champo Carpets must identify the customers who are likely to place an order after receiving a sample. It can be concluded that the Traditional Recommendation System of Champo Carpets is a financial burden because samples correspond to 30% of the transactions as can be seen in Figure 3.3.10. To analyze the accuracy of the samples sent, we created a bar chart, which is shown in Figure 4.1.1:

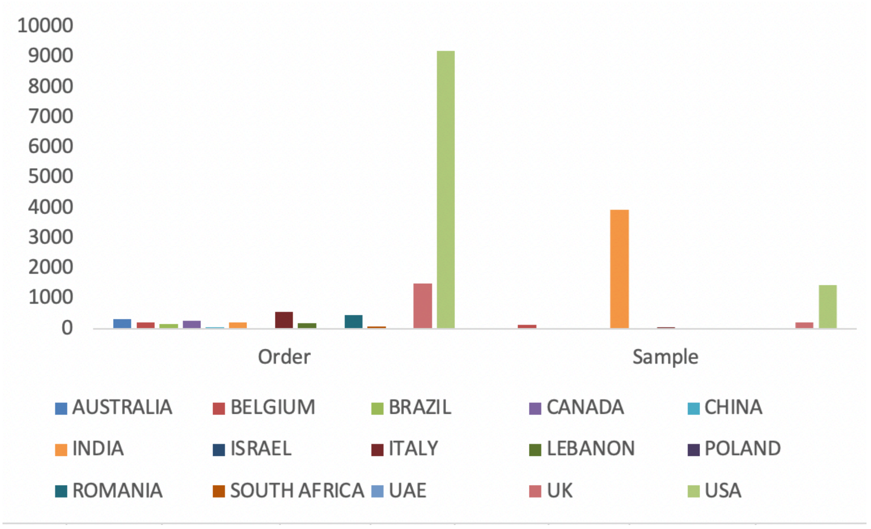


Figure 4.1.1. Bar Chart of Samples and Orders with Respect to Countries

Figure 4.1.1 gives us an outlook on Champo Carpets' Traditional Recommendation System. However, in the graph, there are countries that are negligible with respect to the number of orders they have given. Therefore, to analyze the effectiveness of the traditional recommendation system, we created another graph excluding the negligible countries, as shown in Figure 4.1.2:

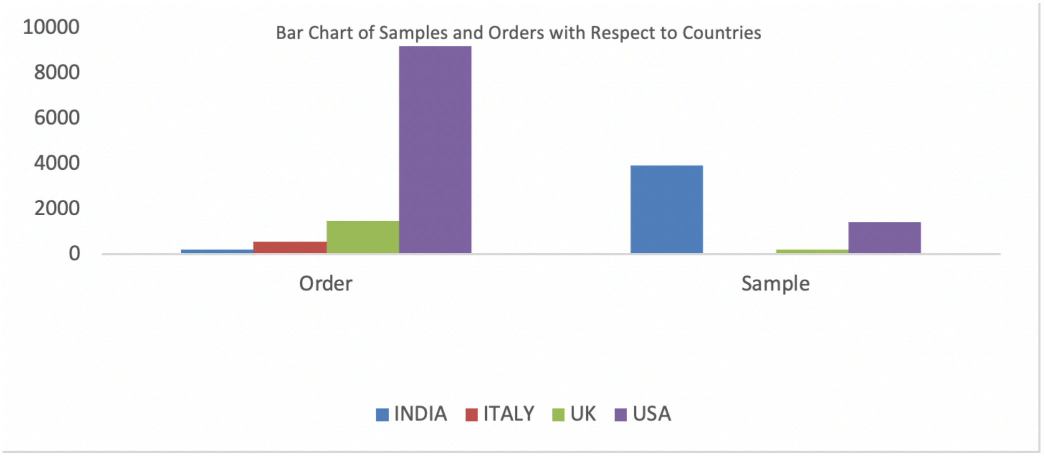


Figure 4.1.2 Bar Chart of Samples and Orders with Respect to Top 5 Countries

Figure 4.1.2 shows us the flaws of Champo Carpets' Traditional Recommendation System, as the countries in the sample and top orders received do not match. India has the 70,1% of the samples but the orders received from India are only 1,7% of the grand total. Thus, we can say that money is wasted when it comes to sending samples to India. Additionally, the number of samples sent to the UK and the US is not efficient as the samples do not cope with the demand from these markets.

All things considered, Champo's current recommendation is seriously flawed, and a collaborative filtering-based recommendation system should be implemented to enhance efficiency in operations and improvement in sales.

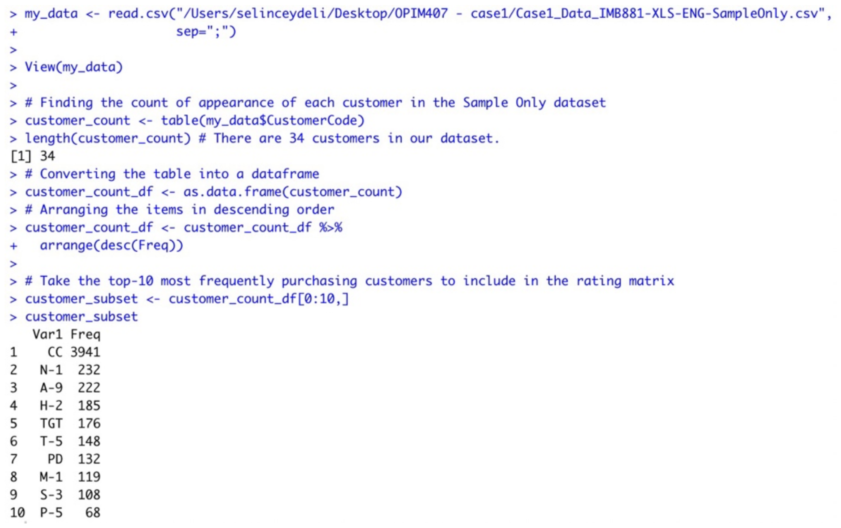
4.2 Benefits of a Collaborative Filtering-based Recommendation System

Collaborative filtering-based recommendation systems are significantly different than traditional ones in many perspectives, as collaborative filtering uses the past data of customers to recommend items based on their behavioral patterns and preferences. Therefore, collaborative filtering is key in making personalized recommendations and increasing the order conversion rate of the samples.

Champo Carpets can benefit from the implementation of a collaborative filtering-based recommendation system in many aspects, such as the preparation of personalized samples, improved accuracy, improved customer loyalty, and increased revenue. Collaborative filtering-based recommendation systems allow companies to identify customer patterns and adjust their businesses accordingly. Identifying customers with similar preferences will significantly improve the accuracy of Champo Carpets' samples, which is an area that desperately needs improvement. Additionally, sending samples that are based on customer preferences would improve customer loyalty and improve the brand image of the business, as the customers will feel valued.

4.3 Item-User Information Representation

To create the collaborative filtering-based recommendation system, it is crucial to determine the Item-User Information, hence we identified each customer in the sample data, and we observed that there are 34 customers, our R script can be found in Figure 4.3.1 below.

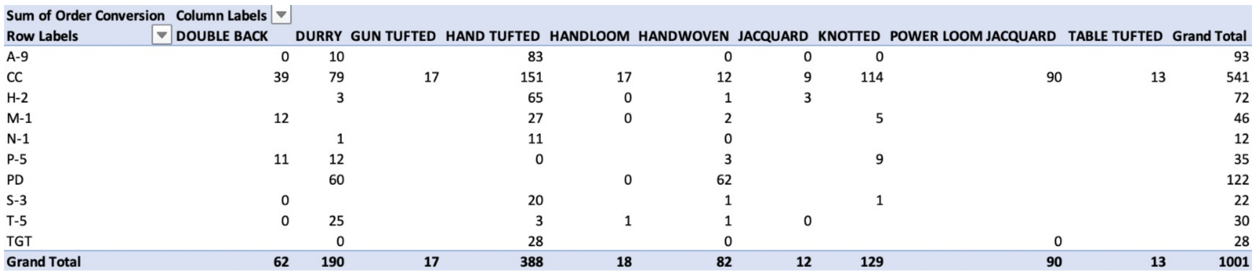


```
> my_data <- read.csv("/Users/selinceydeli/Desktop/OPIM407 - case1/Case1_Data_IMB881-XLS-ENG-SampleOnly.csv",
+                    sep=";")
>
> View(my_data)
>
> # Finding the count of appearance of each customer in the Sample Only dataset
> customer_count <- table(my_data$CustomerCode)
> length(customer_count) # There are 34 customers in our dataset.
[1] 34
> # Converting the table into a dataframe
> customer_count_df <- as.data.frame(customer_count)
> # Arranging the items in descending order
> customer_count_df <- customer_count_df %>%
+   arrange(desc(Freq))
>
> # Take the top-10 most frequently purchasing customers to include in the rating matrix
> customer_subset <- customer_count_df[0:10,]
> customer_subset
   Var1 Freq
1    CC 3941
2   N-1  232
3   A-9  222
4   H-2  185
5   TGT  176
6   T-5  148
7    PD  132
8   M-1  119
9   S-3  108
10  P-5   68
```

Figure 4.3.1. R Script for Identifying Customers

We did not want to include all 34 of the customers into the ratings matrix. Therefore, we decided to sort the customers with respect to the number of purchases they made and take the top-10 most frequently purchasing customers. The reason we limited the matrix with the top-10 most purchasing customers is the cost of sampling. We wanted to include the customers whose purchasing behaviors can be better analyzed due to the high number of purchases they have made so far.

To measure a customer's preference to a sample, it is sensible to consider the samples that are converted into sales, in other words the order conversion value must be 1. Our resultant matrix can be found in Figure 4.3.2 below:



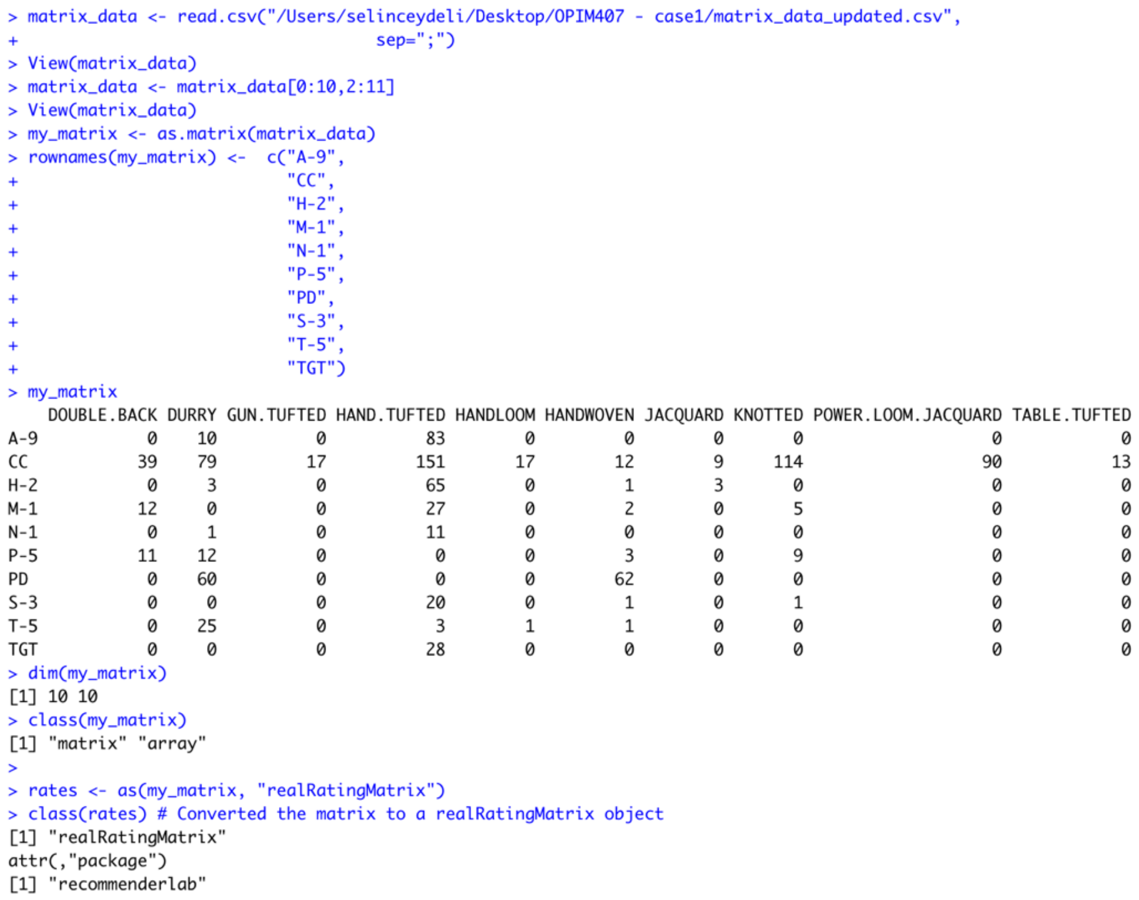
Sum of Order Conversion	Column Labels										
Row Labels	**DOUBLE BACK**	**DURRY**	**GUN TUFTED**	**HAND TUFTED**	**HANDLOOM**	**HANDWOVEN**	**JACQUARD**	**KNOTTED**	**POWER LOOM JACQUARD**	**TABLE TUFTED**	**Grand Total**
A-9	0	10		83		0	0	0			93
CC	39	79	17	151	17	12	9	114	90	13	541
H-2		3		65	0	1	3				72
M-1	12			27	0	2		5			46
N-1		1		11		0					12
P-5	11	12		0		3		9			35
PD		60			0	62					122
S-3	0			20		1		1			22
T-5	0	25		3	1	1	0				30
TGT		0		28		0			0		28
Grand Total	**62**	**190**	**17**	**388**	**18**	**82**	**12**	**129**	**90**	**13**	**1001**

Figure 4.3.2 Customer Preference Matrix

4.4 Implementation of the Collaborative Filtering-based Recommendation System in R

The goal of the R implementation is to create two models for collaborative filtering: user-based and item-based. We explain the process step-by-step with code snippets shown in figures.

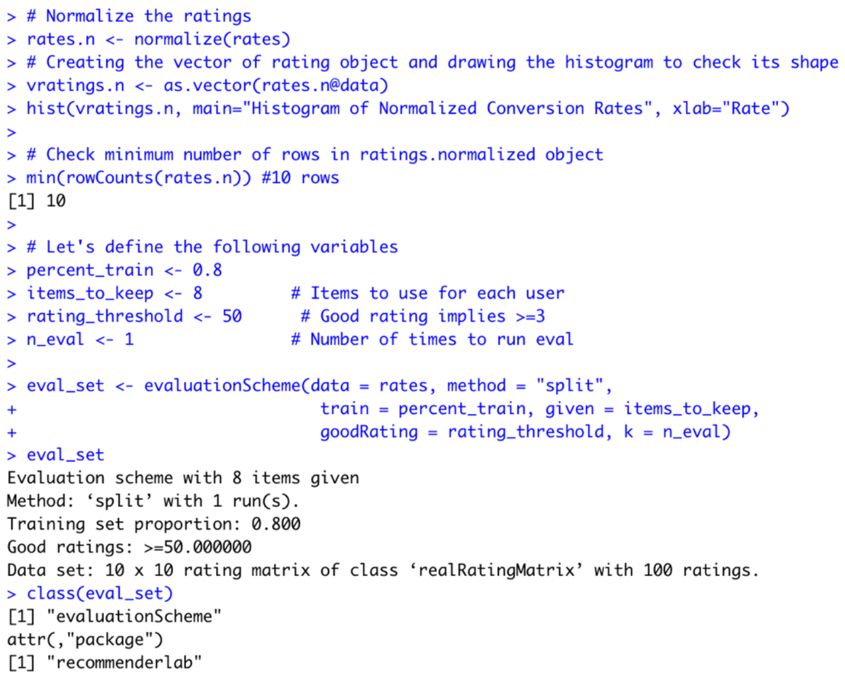
To use an Excel pivot table as an item-user matrix for collaborative filtering in R, we imported it as a data frame, converted it to a matrix object, modified row names to customer codes, and converted it to a real rating matrix object. This process is shown in Figure 4.4.1:



```
> matrix_data <- read.csv("/Users/selinceydeli/Desktop/OPIM407 - case1/matrix_data_updated.csv",
+                        sep=";")
> View(matrix_data)
> matrix_data <- matrix_data[0:10,2:11]
> View(matrix_data)
> my_matrix <- as.matrix(matrix_data)
> rownames(my_matrix) <-  c("A-9",
+                          "CC",
+                          "H-2",
+                          "M-1",
+                          "N-1",
+                          "P-5",
+                          "PD",
+                          "S-3",
+                          "T-5",
+                          "TGT")
> my_matrix
    DOUBLE.BACK DURRY GUN.TUFTED HAND.TUFTED HANDLOOM HANDWOVEN JACQUARD KNOTTED POWER.LOOM.JACQUARD TABLE.TUFTED
A-9           0    10          0          83        0         0        0       0                   0            0
CC           39    79         17         151       17        12        9     114                  90           13
H-2           0     3          0          65        0         1        3       0                   0            0
M-1          12     0          0          27        0         2        0       5                   0            0
N-1           0     1          0          11        0         0        0       0                   0            0
P-5          11    12          0           0        0         3        0       9                   0            0
PD            0    60          0           0        0        62        0       0                   0            0
S-3           0     0          0          20        0         1        0       1                   0            0
T-5           0    25          0           3        1         1        0       0                   0            0
TGT           0     0          0          28        0         0        0       0                   0            0
> dim(my_matrix)
[1] 10 10
> class(my_matrix)
[1] "matrix" "array"
>
> rates <- as(my_matrix, "realRatingMatrix")
> class(rates) # Converted the matrix to a realRatingMatrix object
[1] "realRatingMatrix"
attr(,"package")
[1] "recommenderlab"
```

Figure 4.4.1. Generating a Real Rating Matrix in R

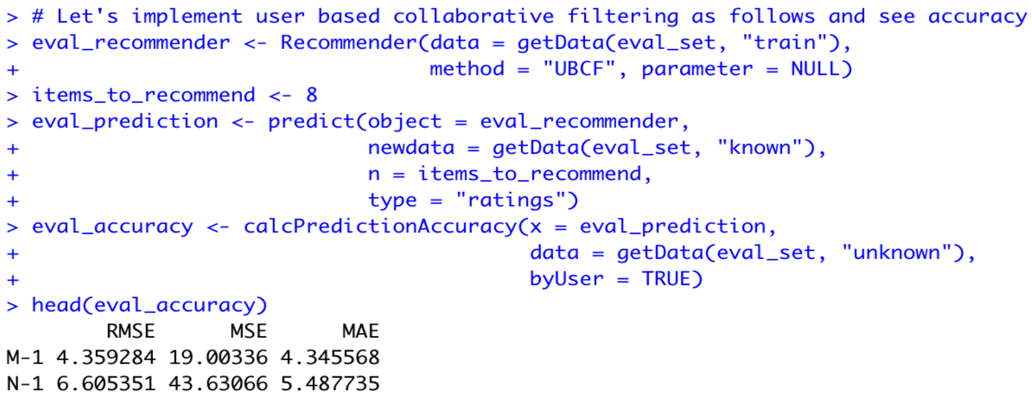
Next, the values in the real rating matrix were normalized and stored in the rates.n object. Then, the minimum number of rows in the rates.n object was found to decide on the items to use for each user. To formulate the evaluation schema, four variables were defined: percent_train, items_to_keep, rating_threshold, and n_eval. In the implementation, holdout data (20% of the dataset) was kept to provide recommendations, and percent_train was set to 0.8. Figure 4.4.2 shows the implementation of these steps:



```
> # Normalize the ratings
> rates.n <- normalize(rates)
> # Creating the vector of rating object and drawing the histogram to check its shape
> vratings.n <- as.vector(rates.n@data)
> hist(vratings.n, main="Histogram of Normalized Conversion Rates", xlab="Rate")
>
> # Check minimum number of rows in ratings.normalized object
> min(rowCounts(rates.n)) #10 rows
[1] 10
>
> # Let's define the following variables
> percent_train <- 0.8
> items_to_keep <- 8         # Items to use for each user
> rating_threshold <- 50      # Good rating implies >=3
> n_eval <- 1                # Number of times to run eval
>
> eval_set <- evaluationScheme(data = rates, method = "split",
+                               train = percent_train, given = items_to_keep,
+                               goodRating = rating_threshold, k = n_eval)
> eval_set
Evaluation scheme with 8 items given
Method: ‘split’ with 1 run(s).
Training set proportion: 0.800
Good ratings: >=50.000000
Data set: 10 x 10 rating matrix of class ‘realRatingMatrix’ with 100 ratings.
> class(eval_set)
[1] "evaluationScheme"
attr(,"package")
[1] "recommenderlab"
```

Figure 4.4.2. Defining the Evaluation Schema

Lastly, we implemented user-based and item-based collaborative filtering models and evaluated their accuracies. The user-based model generated two customer names (M-1 and N-1), as shown in Figure 4.4.3:

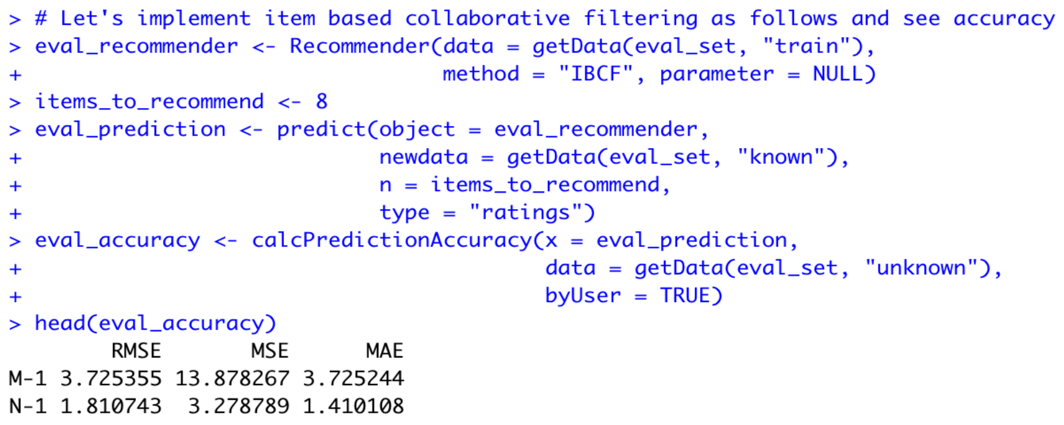


```
> # Let's implement user based collaborative filtering as follows and see accuracy
> eval_recommender <- Recommender(data = getData(eval_set, "train"),
+                                 method = "UBCF", parameter = NULL)
> items_to_recommend <- 8
> eval_prediction <- predict(object = eval_recommender,
+                            newdata = getData(eval_set, "known"),
+                            n = items_to_recommend,
+                            type = "ratings")
> eval_accuracy <- calcPredictionAccuracy(x = eval_prediction,
+                                         data = getData(eval_set, "unknown"),
+                                         byUser = TRUE)
> head(eval_accuracy)
        RMSE      MSE      MAE
M-1 4.359284 19.00336 4.345568
N-1 6.605351 43.63066 5.487735
```

Figure 4.4.3. User-Based Collaborative Filtering Implementation

The user-based collaborative filtering result, showing that customers M-1 and N-1 have similar preferences for recommended products based on their previous ratings, is important for Champo Carpets when sending samples to its customers. This information can be used to make personalized recommendations and suggest items as samples that the customers are more likely to be interested in based on their similarities in preferences. By identifying customers with similar tastes, Champo Carpets can confidently send the same sample that they have sent to one customer to the other customer, knowing there is a high likelihood that they will also like it.

Similarly, the item-based model generated the same two customer names (M-1 and N-1), as shown in Figure 4.4.4:



```
> # Let's implement item based collaborative filtering as follows and see accuracy
> eval_recommender <- Recommender(data = getData(eval_set, "train"),
+                                 method = "IBCF", parameter = NULL)
> items_to_recommend <- 8
> eval_prediction <- predict(object = eval_recommender,
+                            newdata = getData(eval_set, "known"),
+                            n = items_to_recommend,
+                            type = "ratings")
> eval_accuracy <- calcPredictionAccuracy(x = eval_prediction,
+                                         data = getData(eval_set, "unknown"),
+                                         byUser = TRUE)
> head(eval_accuracy)
        RMSE       MSE      MAE
M-1 3.725355 13.878267 3.725244
N-1 1.810743  3.278789 1.410108
```

Figure 4.4.4. Item-Based Collaborative Filtering Implementation

The result of the item-based model indicates the customers M-1 and N-1 have rated similar items positively. Champo Carpets can send items as samples that are like the items that these customers have previously rated positively, with the expectation that they will also enjoy these new samples.

This information yielded by two model can be useful for Champo Carpets in deciding which samples to send to a customer, as it allows the company to target customers who are more likely to be interested in the products it is offering as samples.

4.5 Comparing the Two Models

Both models are most certainly more effective when compared to the Champo Carpets' traditional recommendation system as these models are data-driven and goal oriented. When comparing the two models, we must consider the main problem addressed in the case, which is the price problem when producing sample products. So, it is feasible to use item-based model rather than user-based because it narrows down the variety of samples to send. Determined sample types can be produced in the required quantity so that the dramatic increases in producing costs caused by different variety of sample models is prevented. Item-based approach aligns with the Champo's operations perfectly. Also, the error values in figures 4.4.3 and 4.4.4 show that the accuracy of item-based model is better than user-based model, which is crucial because customers do place an order if the samples are suitable for their needs and preferences. Due to these two reasons, item-based approach is preferable over user-based approach.

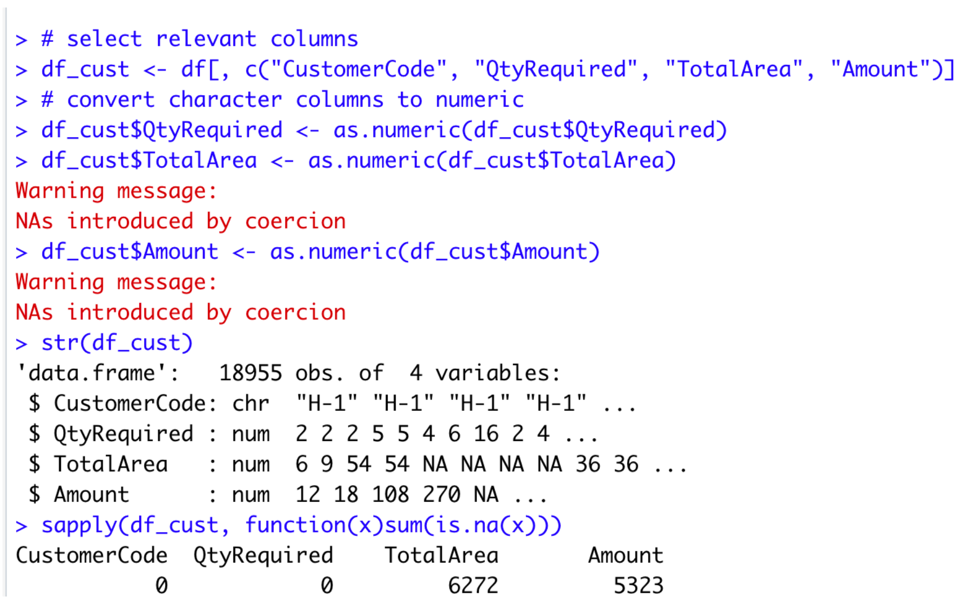
E. CUSTOMER SEGMENTATION USING CLUSTERING

5.1 About Segmentation and the Data

In the dataset there are 18955 observations. For the customer segmentation analysis, the main dataset will be used and there are 18955 observations according to R's STR function. Clustering could benefit to identify hidden groups inside the dataset. For instances, it can group similar transactions and help Champo Carpets to reveal hidden features. In this analysis, k-means clustering will be used since with the specific k value, k many centers are created and according to centers all the data points distributed with Euclidian distance calculation.

5.2 Strategy and Implementation

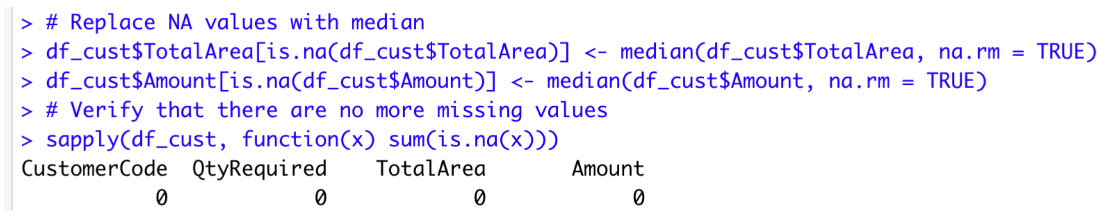
Champo Carpets, at the end of the customer segmentation using clustering will obtain grouped transactions in which it could be useful for resources allocation. For grouping all the transaction, the main dataset must be used for clustering since it contains both sample and order data. To perform customer segmentation with clustering, numerical data is needed for calculating Euclidian distance. The numerical features, such as "QtyRequired", "Amount", and "TotalArea" are stored as char data type so it must be converted to numerical. To deal with this, char features which must be numerical had been transformed to numerical and stored in a new data frame variable for analysis. However, after the transformation to numerical value, many NA variables had been emerged as shown in Figure 5.2.1:



```
> # select relevant columns
> df_cust <- df[, c("CustomerCode", "QtyRequired", "TotalArea", "Amount")]
> # convert character columns to numeric
> df_cust$QtyRequired <- as.numeric(df_cust$QtyRequired)
> df_cust$TotalArea <- as.numeric(df_cust$TotalArea)
Warning message:
NAs introduced by coercion
> df_cust$Amount <- as.numeric(df_cust$Amount)
Warning message:
NAs introduced by coercion
> str(df_cust)
'data.frame':   18955 obs. of  4 variables:
 $ CustomerCode: chr  "H-1" "H-1" "H-1" "H-1" ...
 $ QtyRequired : num  2 2 2 5 5 4 6 16 2 4 ...
 $ TotalArea   : num  6 9 54 54 NA NA NA NA 36 36 ...
 $ Amount      : num  12 18 108 270 NA ...
> sapply(df_cust, function(x)sum(is.na(x)))
CustomerCode  QtyRequired    TotalArea       Amount 
           0            0         6272         5323 
```

Figure 5.2.1. Numerical Conversion of the Quantitative Variables

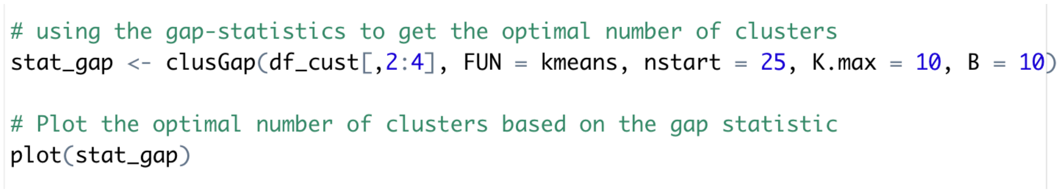
Emerged NA values median imputation in which missing values are filled with the median of all observations as shown in Figure 5.2.2:



```
> # Replace NA values with median
> df_cust$TotalArea[is.na(df_cust$TotalArea)] <- median(df_cust$TotalArea, na.rm = TRUE)
> df_cust$Amount[is.na(df_cust$Amount)] <- median(df_cust$Amount, na.rm = TRUE)
> # Verify that there are no more missing values
> sapply(df_cust, function(x) sum(is.na(x)))
CustomerCode  QtyRequired    TotalArea       Amount 
           0            0            0            0 
```

Figure 5.2.2. Median Imputation

After the handling of the NA values and before clustering the data, our main problem is choosing the right k value. To determine the optimal number of the k value, gap statistics is applied. To measure it, we use R's clusGap function which calculates the gap values of every k value until the k.max. In our case, we calculated gap values of the 10 k values as seen in figure 5.2.3:



```
# using the gap-statistics to get the optimal number of clusters
stat_gap <- clusGap(df_cust[,2:4], FUN = kmeans, nstart = 25, K.max = 10, B = 10)

# Plot the optimal number of clusters based on the gap statistic
plot(stat_gap)
```

Figure 5.2.3. Getting the Optimal Number of Clusters Using Gap Statistics

The results of the gap statistics are shown in Figure 5.2.4 below where the highest gap value that had been emerged is the first k-value, and the lowest gap value that had been emerged is the third k-value. Normally the k value having the highest value would be chosen for the analysis. The algorithm suggests k value to be 1 which is creating only 1 group. Since we filled our NA values with median of the features and since there are many NA values created after transforming the data types to numerical, there is a possibility that NA value handling mislead the gap statistic. To further understand, if there is a mislead in gap statistics, we decided to choose our k value to be the second largest gap value from the graphic, which is 2.

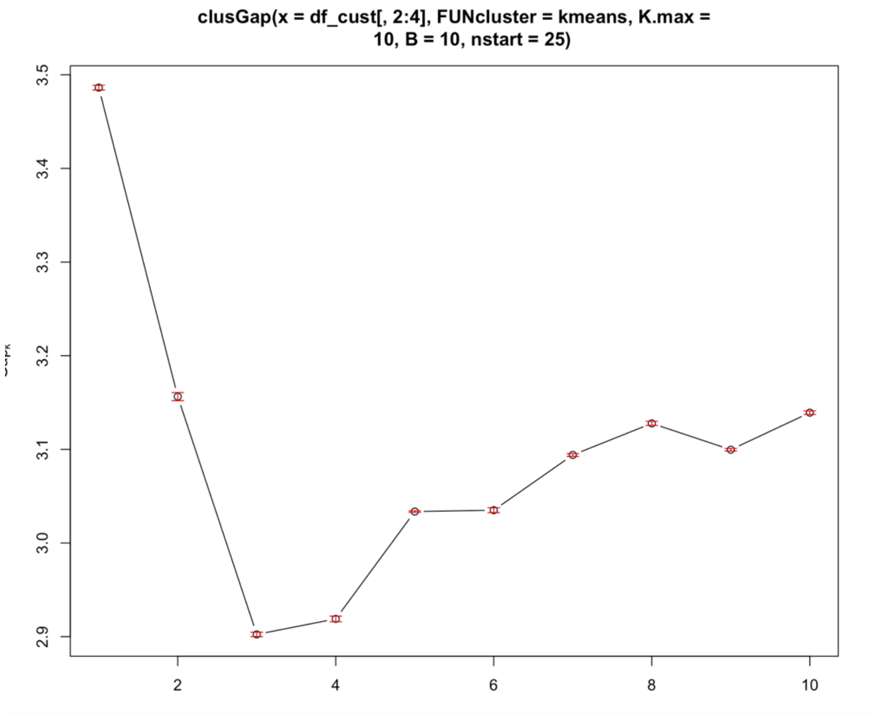
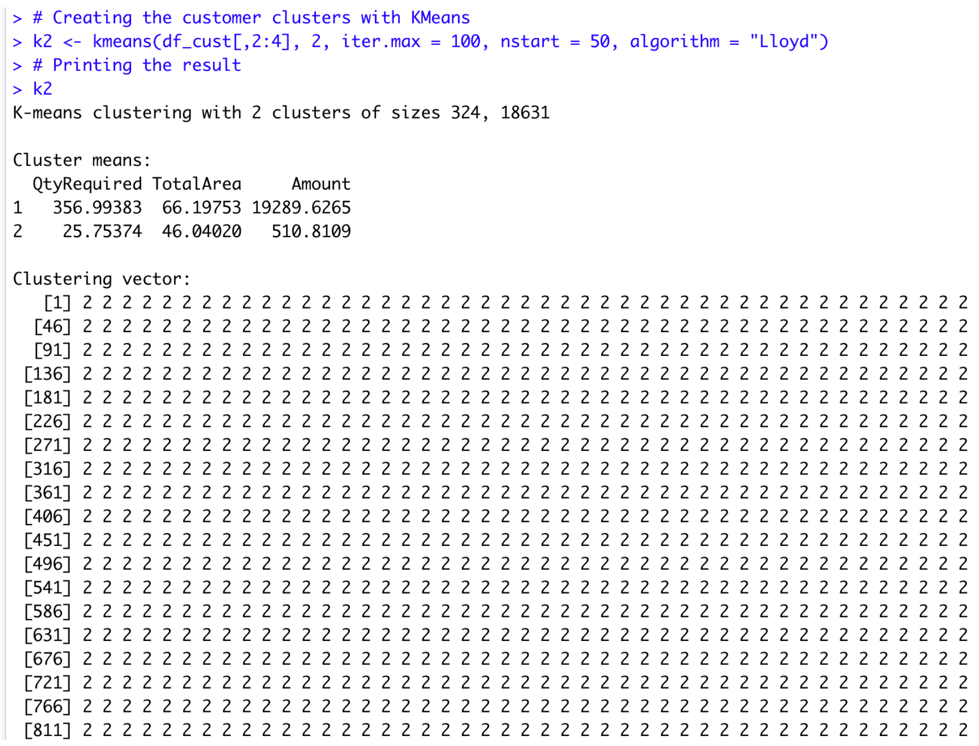


Figure 5.2.4. Result of Gap Statistics

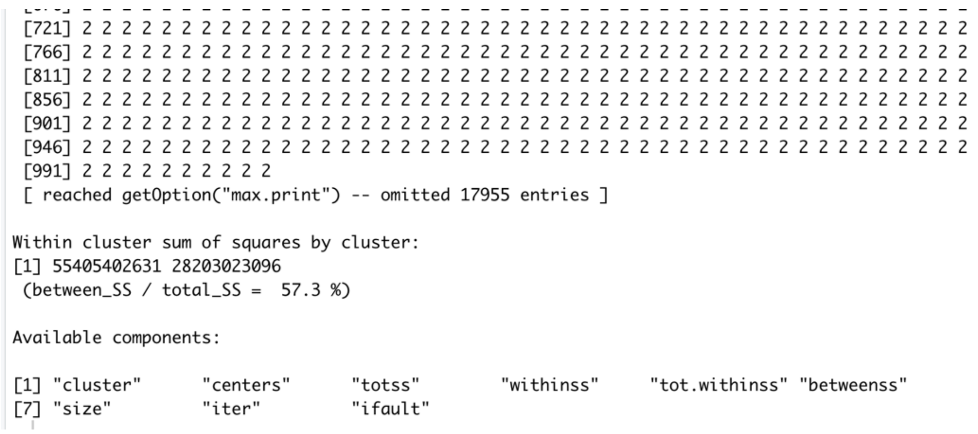
After determining the optimal k-value, the k-means function in R was applied to the data, and the resulting clusters were analyzed. The clusters were then visualized using a plot that showed the amount and quantity required features for each customer segment. The results are presented in figures 5.2.5 and 5.2.6.

In cluster 1, customers who had requested higher amounts of order were grouped together, while in cluster 2, customers who ordered lower amounts of order for each feature were grouped together. The sum of squares by cluster suggests that the algorithm performed reasonably well, achieving a score of 57.3 percent. Overall, these findings suggest that the k-means clustering approach was effective in identifying distinct customer segments based on their purchasing behaviors.



```
> # Creating the customer clusters with KMeans
> k2 <- kmeans(df_cust[,2:4], 2, iter.max = 100, nstart = 50, algorithm = "Lloyd")
> # Printing the result
> k2
K-means clustering with 2 clusters of sizes 324, 18631

Cluster means:
  QtyRequired TotalArea     Amount
1   356.99383  66.19753 19289.6265
2    25.75374  46.04020   510.8109

Clustering vector:
   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  [46] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  [91] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [136] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [181] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [226] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [271] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [316] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [361] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [406] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [451] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [496] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [541] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [586] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [631] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [676] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [721] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [766] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [811] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Figure 5.2.5. Result of K-means Clustering



```
 [721] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [766] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [811] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [856] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [901] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [946] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [991] 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 17955 entries ]

Within cluster sum of squares by cluster:
[1] 55405402631 28203023096
 (between_SS / total_SS =  57.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

Figure 5.2.6. Result of K-means Clustering

The two clusters, which are grouped based on their order volumes using Amount and QtyRequired variables, can be observed in the plot in Figure 5.2.7. By grouping customers into two clusters based on their order volumes using Amount and QtyRequired variables, Champo Carpets can efficiently allocate resources by arranging production according to cluster centroid values. This analysis is particularly beneficial for customers in cluster 1 with high order volumes. By anticipating purchase intentions from these customers and planning ahead, Champo can optimize their stock management and production processes. The insights gained from this clustering analysis can ultimately lead to improved customer satisfaction and profitability for Champo Carpets.

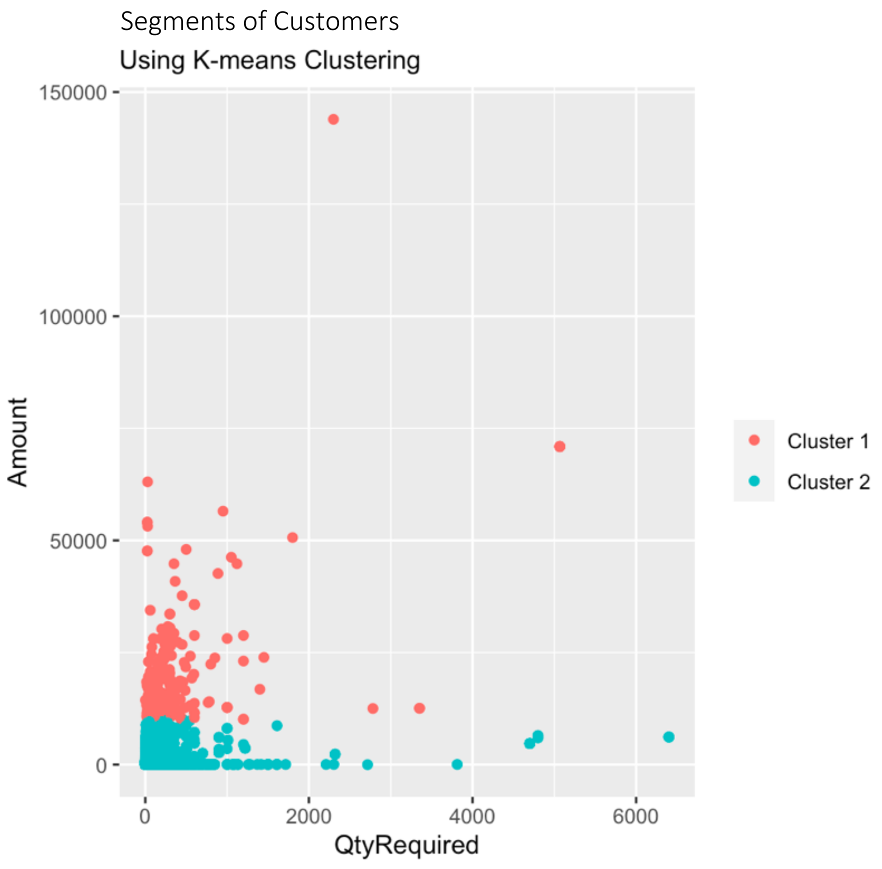


Figure 5.2.7 Segments of Customers

F. CONCLUSION

Champo Carpets lacked personalized recommendations due to traditional sample preparation, resulting in low conversion rates. After conducting a detailed exploratory data analysis, we proposed using carpet attributes to create customer segments and build a recommendation system for clients. Collaborative filtering was conducted, with the item-based model outperforming the user-based model in predicting customer preferences. Customers M-1 and N-1 had been identified to have similar preferences in both models. Additionally, customer segmentation using clustering was conducted to identify the hidden customer groups. Two groups had emerged with clustering the numerical values of the main dataset. Similar orders and sample orders from customers had been classified.

The case highlights that traditional decision-making systems are insufficient for generating maximum revenue and maintaining customer loyalty. Data analysis is necessary to identify suitable attributes and inform decision-making using appropriate analytics models like collaborative filtering and customer segmentation. Applying appropriate data models is crucial for making company-wide decisions as they help to uncover patterns and insights in large datasets. These models allow companies to better understand their customers, their preferences, and their behaviors, which can then inform important decisions.

For future work on the same case study, further analysis on customer behavior and preferences can be conducted using advanced analytics techniques such as machine learning and artificial intelligence. Moreover, it could be asked from Champo Carpets to collect more data on customer demographics to better understand the customer behavior and preferences, and the current collaborative filtering and customer segmentation models can be trained with this new information. As a last remark for future work, the recommendation system defined in this case study can be implemented as a decision-making system, replacing Champo Carpets' traditional decision-making system, and the effectiveness of this new system can be evaluated through customer feedback and comparison of the conversion rates before and after the implementation.

G. TABLE DESIGN PRINCIPLES AND GRAPHICAL EXCELLENCE

In our case report, there are 12 charts and 1 table, each item is put in this report to illustrate a specific point; to deliver these points in the right way, we complied with the requirements of graphical excellence, we tried to give the most information with the least ink usage. Each graph has an introduction and a title. We made scatter charts and box plots colorful because the dispersion of the data is important for these data visualization items. Our table in the pages 2 and 3, meets the requirements of table design principles as it has proper column headers and there is no unnecessary text.

H. APPENDIX

8.1 R-Script of Exploratory Data Analysis

```r
# Loading the required packages
library(ggplot2)
library(reshape2)
library(dplyr)
# Importing Data from CSV file To a Data Frame Object
df = read.csv("/Users/selinceydeli/Desktop/OPIM407 - case1/Case1_Data_IMB881-XLS-ENG.csv", sep=";")
# Displaying data in df data frame
View(df)
# Displaying the types of the variables (i.e columns of data frame)
str(df) #The only numerical variable is QtyRequired
# Displaying the first five and first 10 observations in your data frame
head(df)
head(df,10)
# Displaying the last five and last 10 observations in your data frame
tail(df)
tail(df,10)
# Are there any null values in the dataset?
sapply(df, function(x) sum(is.na(x))) #There aren't any NA values
# Descriptive Statistics
# Displaying the summary for each variable in the data frame
summary(df)
# The only numerical variable is QtyRequired-> I will apply summary statistics to this variable
summary(df$QtyRequired)
# Understanding the measures of variability of the QtyRequired variable,
range(df$QtyRequired)
IQR(df$QtyRequired)
var(df$QtyRequired)
sd(df$QtyRequired)
CV <- sd(acs_df$electricity) / mean(acs_df$electricity)
# The CV is the ratio of the standard deviation to the mean, expressed as a percentage. It is a measure of relative variability that can
# be used to compare the variation between datasets with different units.
CV <- sd(acs_df$electricity) / mean(acs_df$electricity)
# Result: A coefficient of variation (CV) of 0.6119009 for the QtyRequired variable indicates that the standard deviation is quite high relative to
the mean.
# Data Visualization
# 1. Shape Name Attribute
# Frequency distribution (i.e. a tabular summary) of data showing number of observations (i.e. frequency) in non-overlapping categories (bins)
table(df$ShapeName)
# Relative frequency distribution (i.e. a tabular summary) of data showing the relative frequency in non overlapping categories (bins)
table(df$ShapeName)/length(df$ShapeName)
# Percent frequency distribution (i.e. a tabular summary) of data showing the percent frequency in non overlapping categories (bins)
100*table(df$ShapeName)/length(df$ShapeName)
# Creating a bar plot
ylim <- c(0,20000)
```

```
barplot(table(df$ShapeName), ylim = ylim)
# Adding y-axis values on top of the bars
y_values <- table(df$ShapeName)
text(x = barplot(table(df$ShapeName), ylim = ylim, col="blue",
            xlab = "Carpet Shape", ylab = "Frequency",
            main="Distribution of Carpets Produced Based on Shape") - 0.05,
            y = y_values, label = y_values, pos = 3)
# 2. Country Name Attribute
table(df$CountryName)
table(df$CountryName)/length(df$CountryName)
100*table(df$CountryName)/length(df$CountryName)
ylim <- c(0,12000)
barplot(table(df$CountryName), ylim = ylim, col="red")
mytable <- table(df$CountryName)
mybarplot <- barplot(mytable, ylim = ylim, col = "red",
                     xlab = "Country Name", ylab = "Frequency",
                     main = "Distribution of Countries as Customers of Champo Carpets",
                     cex.names = 0.7)
text(mybarplot, mytable, labels = mytable, pos = 3, cex.names = 0.8)
#Checking the percent frequency of the Countries
100*table(df$CountryName)/length(df$CountryName)
# 3. Stacked Bar Charts
# Converting the QtyRequired column from character type to numeric type
df$QtyRequired <- as.numeric(df$QtyRequired)
# Aggregate the data by customer code and item name
freq <- aggregate(df$QtyRequired, by = list(CustomerCode = df$CustomerCode,
                                            ItemName = df$ITEM_NAME), FUN = sum)
# Sort the data by quantity required in descending order
freq <- freq[order(-freq$x),]
# Plot the aggregated bar chart
ggplot(freq, aes(x = CustomerCode, y = x, fill = ItemName)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Total Quantity Produced for Customer and Item",
       x = "Customer Code",
       y = "Total Quantity Produced",
       fill = "Item Name")
# Converting the QtyRequired column from character type to numeric type
df$QtyRequired <- as.numeric(df$QtyRequired)
# Aggregating the data by customer code and item name
freq <- aggregate(df$QtyRequired, by = list(CountryName = df$CountryName,
                                            ItemName = df$ITEM_NAME), FUN = sum)
# Sort the data by quantity required in descending order
freq <- freq[order(-freq$x),]
# Plot the aggregated bar chart
ggplot(freq, aes(x = CountryName, y = x, fill = ItemName)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Total Quantity Produced for Country and Item",
       x = "Country Name",
       y = "Total Quantity Produced",
       fill = "Item Name")
# Creating a Box Plot for summarizing the distribution of QtyRequired for each category of ITEM_NAME,
my_colors <- c("#00AFBB", "#E7B800", "#FC4E07", "#00A087", "#6A3D9A", "#FFB5B8")
boxplot(QtyRequired ~ ITEM_NAME, data = df,
        main = "Box Plot of Quantity Required by Item Name",
        xlab = "Item Name", ylab = "Quantity Required", cex.axis=0.5)
boxplot(QtyRequired ~ ITEM_NAME, data = df,
        main = "Box Plot of Quantity Required by Item Name",
        xlab = "Item Name", ylab = "Quantity Required", ylim = c(0,1000),
        cex.axis=0.5)
boxplot(QtyRequired ~ ITEM_NAME, data = df,
        main = "Box Plot of Quantity Required by Item Name",
        xlab = "Item Name", ylab = "Quantity Required", ylim = c(0,100),
        col=my_colors, cex.axis=0.5)
boxplot(QtyRequired ~ ITEM_NAME, data = df,
        main = "Box Plot of Quantity Required by Item Name",
        xlab = "Item Name", ylab = "Quantity Required", ylim = c(0,50),
        col=my_colors, cex.axis=0.5)
# Outliers analysis using box plots and IQR calculation
# Method 2: Observations beyond the whiskers of a boxplot
DurrySubset <- df$QtyRequired[df$ITEM_NAME=="DURRY"]
IQR <- quantile(DurrySubset, 0.75) -
  quantile(DurrySubset, 0.25)
# Method 2a" Mild outliers
which(DurrySubset < quantile(DurrySubset, 0.25) - 1.5*IQR |
        DurrySubset > quantile(DurrySubset, 0.75) + 1.5*IQR)
length(which(DurrySubset < quantile(DurrySubset, 0.25) - 1.5*IQR |
               DurrySubset > quantile(DurrySubset, 0.75) + 1.5*IQR))
# There are 618 mild outliers
# Method 2b: Extreme outliers
which(DurrySubset < quantile(DurrySubset, 0.25) - 3*IQR |
        DurrySubset > quantile(DurrySubset, 0.75) + 3*IQR)
length(which(DurrySubset < quantile(DurrySubset, 0.25) - 3*IQR |
               DurrySubset > quantile(DurrySubset, 0.75) + 3*IQR))
# Creating a Box Plot for summarizing the distribution of QtyRequired for each category of ShapeName
boxplot(QtyRequired ~ ShapeName, data = df,
        main = "Box Plot of Quantity Required by Shape Name",
        xlab = "Shape Name", ylab = "Quantity Required",
        col=my_colors)
boxplot(QtyRequired ~ ShapeName, data = df,
        main = "Box Plot of Quantity Required by Shape Name",
        xlab = "Shape Name", ylab = "Quantity Required", ylim = c(0,75),
        col=my_colors)
```

```
# Creating a pie chart to show the count of each category of the OrderType variable
# Create a data frame with the counts for each category
df_counts <- df %>%
  group_by(OrderType) %>%
  summarize(count = n())
# Create the pie chart
ggplot(df_counts, aes(x = "", y = count, fill = OrderType)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  ggtitle("Count of Each Category of the OrderType Variable") +
  theme_minimal() +
  theme(legend.position = c(1.0, 0.2)) +
  labs(fill = "Order Type")
# Creating a pie chart to show the count of each category of the OrderCategory variable
# Create a data frame with the counts for each category
df_counts <- df %>%
  group_by(OrderCategory) %>%
  summarize(count = n())
# Create the pie chart
ggplot(df_counts, aes(x = "", y = count, fill = OrderCategory)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar(theta = "y") +
  ggtitle("Count of Each Category of the OrderCategory Variable") +
  theme_minimal() +
  theme(legend.position = c(1.0, 0.2)) +
  labs(fill = "Order Category")
```

8.2 R-Script of Collaborative Filtering

```
library(dplyr)
library(ggplot2)
library(knitr)
library(recommenderlab)
my_data <- read.csv("/Users/selinceydeli/Desktop/OPIM407 - case1/Case1_Data_IMB881-XLS-ENG-SampleOnly.csv",
          sep=";")
View(my_data)
# Finding the count of appearance of each customer in the Sample Only dataset
customer_count <- table(my_data$CustomerCode)
length(customer_count) # There are 34 customers in our dataset.
# Converting the table into a dataframe
customer_count_df <- as.data.frame(customer_count)
# Arranging the items in descending order
customer_count_df <- customer_count_df %>%
  arrange(desc(Freq))
# Take the top-10 most frequently purchasing customers to include in the rating matrix
customer_subset <- customer_count_df[0:10,]
# Finding the count of appearance of each item in the Sample Only dataset
```

```
product_count <- table(my_data$ITEM_NAME)
# Converting the table into a dataframe
product_count_df <- as.data.frame(product_count)
# Arranging the items in descending order
product_count_df <- product_count_df %>%
  arrange(desc(Freq))
matrix_data <- read.csv("/Users/selinceydeli/Desktop/OPIM407 - case1/matrix_data_updated.csv",
                        sep=";")
View(matrix_data)
matrix_data <- matrix_data[0:10,2:11]
my_matrix <- as.matrix(matrix_data)
rownames(my_matrix) <-  c("A-9",
                     "CC",
                     "H-2",
                     "M-1",
                     "N-1",
                     "P-5",
                     "PD",
                     "S-3",
                     "T-5",
                     "TGT")
my_matrix
dim(my_matrix)
class(my_matrix)
rates <- as(my_matrix, "realRatingMatrix")
class(rates) # Converted the matrix to a realRatingMatrix object
# Normalize the ratings
rates.n <- normalize(rates)
# Creating the vector of rating object and drawing the histogram to check its shape
vratings.n <- as.vector(rates.n@data)
hist(vratings.n, main="Histogram of Normalized Conversion Rates", xlab="Rate")
# Check minimum number of rows in ratings.normalized object
min(rowCounts(rates.n)) #10 rows
# Let's define the following variables
percent_train <- 0.8
items_to_keep <- 8         # Items to use for each user
rating_threshold <- 50     # Good rating implies >=3
n_eval <- 1                # Number of times to run eval
eval_set <- evaluationScheme(data = rates, method = "split",
                       train = percent_train, given = items_to_keep,
                       goodRating = rating_threshold, k = n_eval)
class(eval_set)

# Let's implement user based collaborative filtering as follows and see accuracy
eval_recommender <- Recommender(data = getData(eval_set, "train"),
                        method = "UBCF", parameter = NULL)
items_to_recommend <- 8
```

```
eval_prediction <- predict(object = eval_recommender,
                    newdata = getData(eval_set, "known"),
                    n = items_to_recommend,
                    type = "ratings")
eval_accuracy <- calcPredictionAccuracy(x = eval_prediction,
                         data = getData(eval_set, "unknown"),
                         byUser = TRUE)
head(eval_accuracy)
# Let's implement item based collaborative filtering as follows and see accuracy
eval_recommender <- Recommender(data = getData(eval_set, "train"),
                      method = "IBCF", parameter = NULL)
items_to_recommend <- 8
eval_prediction <- predict(object = eval_recommender,
                    newdata = getData(eval_set, "known"),
                    n = items_to_recommend,
                    type = "ratings")
eval_accuracy <- calcPredictionAccuracy(x = eval_prediction,
                         data = getData(eval_set, "unknown"),
                         byUser = TRUE)
head(eval_accuracy)
```

8.3 R-Script of Customer Segmentation using Clustering

```
# Loading the required packages
library(ggplot2)
library(reshape2)
library(dplyr)
library(tibble)
library(cluster)
# Importing Data from CSV file To a Data Frame Object
df = read.csv("/Users/sinanyilmaz/Desktop/OKUL/402/OPIM 407/Case_1/Case1_Data_IMB881-XLS-ENG.csv",
       sep=";")
names(df)
str(df)
# store every transaction order or sample and necessary information
summary(df)
# select relevant columns
df_cust <- df[, c("CustomerCode", "QtyRequired", "TotalArea", "Amount")]
# convert character columns to numeric
df_cust$QtyRequired <- as.numeric(df_cust$QtyRequired)
df_cust$TotalArea <- as.numeric(df_cust$TotalArea)
df_cust$Amount <- as.numeric(df_cust$Amount)
str(df_cust)
# check for missing values
sapply(df_cust, function(x) sum(is.na(x)))
# median imputation
df_cust$TotalArea[is.na(df_cust$TotalArea)] <- median(df_cust$TotalArea, na.rm = TRUE)
```

```
df_cust$Amount[is.na(df_cust$Amount)] <- median(df_cust$Amount, na.rm = TRUE)
# Verify that there are no more missing values
sapply(df_cust, function(x) sum(is.na(x)))
# using the gap-statistics to get the optimal number of clusters
stat_gap <- clusGap(df_cust[,2:4], FUN = kmeans, nstart = 25, K.max = 10, B = 10)
# Plot the optimal number of clusters based on the gap statistic
plot(stat_gap)
# Creating the customer clusters with KMeans
k2 <- kmeans(df_cust[,2:4], 2, iter.max = 100, nstart = 50, algorithm = "Lloyd")
# Create a plot of the customers segments
ggplot(df_cust, aes(x = QtyRequired, y = Amount)) +
  geom_point(stat = "identity", aes(color = as.factor(k2$cluster))) +
  scale_color_discrete(name = " ",
                       breaks = c("1", "2"),
                       labels = c("Cluster 1", "Cluster 2")) +
  ggtitle("Segments of Customers", subtitle = "Using K-means Clustering")
```