Table of Contents

1. Introduction

The origins of "Black Friday" stem not from a day filled with shopping, discounts, and a turn of the holiday season, but rather with a financial crisis. The first recorded use of the term "Black Friday" was recorded on September 24th, 1869 when two Wall Street businessmen, Jay Gould and Jim Fisk, decided to artifically inflate the price of gold and attempted to sell it for profit. The result of their nefarious actions, on that specific Friday in September 1869, the price of Gold dropped and the United States plunged into a state of financial devestation.[1]

Picture 1. https://justcreative.com/wp-content/uploads/2018/10/black-friday-deals.jpg

1.1 First Recorded Use

Various stories exist regarding the first recorded use of the term as it relates to holiday shopping, but its connotation continued to keep a negetive stigma associated with it until the late 20th century. "Black Friday" and its relation to consumerism first derived from 1950s Philadelphia. Philadelphia suburbinites descended on the city after the Thanksgiving holiday, to watch the traditional Army college football game and take advantage of sales and promotions, Philidelphia Police Officers who were assigned to work that weekend coined the term due to their long grueling shifts and the mass amounts of people/shoppers. Philidelphia businesses also started to use the term to describe the long lines and shopping mayhem at their stores.

[1] https://en.wikipedia.org/wiki/Black_Friday_(shopping)

1.2. Use Within Business

One of the possible explanation for the term as it relates to consumers and retailers is that "Black Friday" represents the first day of the year in which businesses were turning profits and accounting was done on a hand-written ledger. As described Oxford Dictionary, "The use of colors in accounting refers back to the bookkeeping practice of recording the credit side of an account in a ledger in *black* ink and the debit side in *red* ink." (Oxford Dictionaries) Hence the name, "Black Friday" being associated with businesses debits overtaking their credits. Although this idea might make sense, the claim hasn't been completely verified.[2]

1.3 Description of dataset

The dataset here is a sample of the transactions made in a retail store. The store wants to know better the customer purchase behaviour against different products. Specifically, here the problem is a regression problem where we are trying to predict the dependent variable (the amount of purchase) with the help of the information contained in the other variables.

Classification problem can also be settled in this dataset since several variables are categorical, and some other approaches could be "Predicting the age of the consumer" or even "Predict the category of goods bought". This dataset is also particularly convenient for clustering and maybe find different clusters of consumers within it.[3]

1. Main aims of the project are:

- analysis all of dataset in relation to variables
- showing the possibility of packages in R and the Python

2. My research hypotheses are:

1. The male customers have a higher average spending then the female.
2. Which gender and in which age have achaived more purchase.
3. The customers are young people.

[2] Oxford Dictionaries- Black Friday
[3] https://www.kaggle.com/mehdidag/black-friday

2. Required R – Packages

There are several R packages that useful for analyzing this dataset.

- dplyr - tool frome processing dataset. *2.1*
- ggplot2 - creating graphics. *2.2*
- plotly - to help make pie chart. *2.3*

2.1.The dplyr package makes these steps fast and easy:

- By constraining your options, it helps you think about your data manipulation challenges.
- It provides simple “verbs”, functions that correspond to the most common data manipulation tasks, to help you translate your thoughts into code.
- It uses efficient backends, so you spend less time waiting for the computer.[4]

2.2 The ggplot2 is a data visualization package for the statistical programming language R. Created by Hadley Wickham in 2005, ggplot2 is an implementation of Leland Wilkinson's *Grammar of Graphics* — a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. ggplot2 can serve as a replacement for the base graphics in R and contains a number of defaults for web and print display of common scales. Since 2005, ggplot2 has grown in use to become one of the most popular R packages.[5]

2.3 Plotly's R graphing library makes interactive, publication-quality graphs online. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, and 3D (WebGL based) charts.[6]

[4] https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
[5] https://en.wikipedia.org/wiki/Ggplot2
[6] https://plot.ly/r/

2.4 Required Python 3 – Packages

numpy as np - linear algebra
pandas as pd - data processing, CSV file I/O (e.g. pd.read_csv)
matplotlib.pyplot as plt

There are several Python packages that useful for analyzing dataset's:

2.4.1. NumPy is the fundamental package for scientific computing with Python. It contains among other things:[7]

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

2.4.2. Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis and manipulation tool available in any language.[8]

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously - typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure.

2.4.3. Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.[9]

[7] https://www.numpy.org/
[8] https://pandas.pydata.org/
[9] https://matplotlib.org/

2.3 Which is better for data analysis: R or Python

R is considered to be the best programming language for any statistician as it possesses an extensive catalog of statistical and graphical methods. Python on the other hand can do pretty much the same work as R but it is preferred by the data scientists or data analysts because of its simplicity and high performance. R is a powerful scripting language and highly flexible with a vibrant community and resource bank whereas Python is a widely used, object oriented language which is easy to learn and debug.

R has a much bigger library of statistical packages

If we doing specialized statistical work, R packages cover more techniques. We can find R packages for a wide variety of statistical tasks using the CRAN task view. R packages cover everything from Psychometrics to Genetics to Finance. Although Python, through SciPy and packages like statsmodels, covers the most common techniques, R is far ahead.

Python is better for building analytics tools

R and Python are equally good if you want to find outliers in a dataset, but if we want to create a web service to enable other people to upload datasets and find outliers, Python is better. Python is a general purpose programming language, which means that people have built modules to create websites, interact with a variety of databases, and manage users.

R builds in data analysis functionality by default, whereas Python relies on packages

Python is a general purpose language, most data analysis functionality is available through packages like NumPy and pandas. However, R was built with statistics and data analysis in mind, so many tools that have been added to Python through packages are built into base R.

Python is better for deep learning

Through packages like Lasagne, caffe, keras, and tensorflow, creating deep neural networks is straightforward in Python. Although some of these, like tensorflow, are being ported to R, support is still far better in Python.

Python relies on a few main packages, whereas R has hundreds

In Python, sklearn is the “primary” machine learning package, and pandas is the “primary” data analysis package. This makes it easy to know how to accomplish a task, but also means that a lot of specialized techniques aren’t possible.

R is better for data visualization

Packages like ggplot2 make plotting easier and more customizable in R than in Python. Python is catching up, particularly in the area of interactive plots with packages like Bokeh.

3.1 Exploratory Data Analysis (EDA)

To load the dataset that we will be using for this Exploratory Data Analysis (EDA).

```
#importing the dataset
data = pd.read_csv('../input/BlackFriday.csv')
This dataset has 12 variables
```

`##   User_ID Product_ID Gender   Age Occupation City_Category`
`## 1 1000001  P00069042      F  0-17         10             A`
`## 2 1000001  P00248942      F  0-17         10             A`
`## 3 1000001  P00087842      F  0-17         10             A`
`## 4 1000001  P00085442      F  0-17         10             A`
`## 5 1000002  P00285442      M   55+         16             C`
`## 6 1000003  P00193542      M 26-35         15             A`
`##   Stay_In_Current_City_Years Marital_Status Product_Category_1`
`## 1                          2              0                  3`
`## 2                          2              0                  1`
`## 3                          2              0                 12`
`## 4                          2              0                 12`
`## 5                         4+              0                  8`
`## 6                          3              0                  1`
`##   Product_Category_2 Product_Category_3 Purchase`
`## 1                 NA                 NA     8370`
`## 2                  6                 14    15200`
`## 3                 NA                 NA     1422`
`## 4                 14                 NA     1057`
`## 5                 NA                 NA     7969`
`## 6                  2                 NA    15227`

Table 1. Code from R program with data analises.
Source: Own elaboration.

To read and analize data by using Pandas library:

We will also import all libraries which will be used.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns  # visualization tool
import os
print(os.listdir("../input"))

# read BlackFriday.csv file data from input directory and create dataframe named data
data = pd.read_csv("../input/BlackFriday.csv")
```

After reading csv file , we will check data by using „info()" method of dataframe , we can see column names and they types .

When we execute command we will see there are 537577 entries in file. For Product_Category_1 column , 164278 of 537577 are non-null and that means rest are null. For Product_Category_2 column , 370591 of 537577 are non-null and that means rest are null and in other columns are all full and there is no empty data.

```
# check information about columns data
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
User_ID                       537577 non-null int64
Product_ID                    537577 non-null object
Gender                        537577 non-null object
Age                           537577 non-null object
Occupation                    537577 non-null int64
City_Category                 537577 non-null object
Stay_In_Current_City_Years    537577 non-null object
Marital_Status                537577 non-null int64
Product_Category_1            537577 non-null int64
Product_Category_2            370591 non-null float64
Product_Category_3            164278 non-null float64
Purchase                      537577 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

Table 2. Information about columns data.
Source: Own elaboration.

When we look on first 10 data of dataframe to have knowledge data itself.

```
#data.head(10)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969
5	1000003	P00193542	M	26-35	15	A	3	0	1	2.0	NaN	15227
6	1000004	P00184942	M	46-50	7	B	2	1	1	8.0	17.0	19215
7	1000004	P00346142	M	46-50	7	B	2	1	1	15.0	NaN	15854
8	1000004	P0097242	M	46-50	7	B	2	1	1	16.0	NaN	15686
9	1000005	P00274942	M	26-35	20	A	1	1	8	NaN	NaN	7871

Table 3. First 10 data of dataframe.
Source: Own elaboration.

We can see there are **NaN** m_values for Product_Category_2 and Product_Category_3.

We want to fill them with zero.

```
data.Product_Category_2.fillna(0, inplace=True)
data.Product_Category_3.fillna(0, inplace=True)
```

By using describe method of `dataFrame` , we can learn some statistical information such as mean(average), max, min etc about data.

```
# display columns
data.columns
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation',
'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status',
'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

Now we can check statictical information about numeric data columns

```
data.describe()
```

	User_ID	**Occupation**	**Marital_Status**	**Product_Category_1**	**Product_Category_2**	**Product_Category_3**	**Purchase**
count	5.375770e+05	537577.00000	537577.000000	537577.000000	537577.000000	537577.000000	537577.000000
Mean	1.002992e+06	8.08271	0.408797	5.295546	6.784907	3.871773	9333.859853
std	1.714393e+03	6.52412	0.491612	3.750701	6.211618	6.265963	4981.022133
min	1.000001e+06	0.00000	0.000000	1.000000	0.000000	0.000000	185.000000
25%	1.001495e+06	2.00000	0.000000	1.000000	0.000000	0.000000	5866.000000
50%	1.003031e+06	7.00000	0.000000	5.000000	5.000000	0.000000	8062.000000
75%	1.004417e+06	14.00000	1.000000	8.000000	14.000000	8.000000	12073.000000
max	1.006040e+06	20.00000	1.000000	18.000000	18.000000	18.000000	23961.000000

Table 4 . First 10 data of dataframe to have knowledge data itself.
Source: Own elaboration.

```
data.corr()
```

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
User_ID	1.000000	-0.023024	0.018732	0.003687	0.003663	0.003938	0.005389
Occupation	-0.023024	1.000000	0.024691	-0.008114	0.006792	0.011941	0.021104
Marital_Status	0.018732	0.024691	1.000000	0.020546	0.001146	-0.004363	0.000129
Product_Category_1	0.003687	-0.008114	0.020546	1.000000	-0.040730	-0.389048	-0.314125
Product_Category_2	0.003663	0.006792	0.001146	-0.040730	1.000000	0.090284	0.038395
Product_Category_3	0.003938	0.011941	-0.004363	-0.389048	0.090284	1.000000	0.284120
Purchase	0.005389	0.021104	0.000129	-0.314125	0.038395	0.284120	1.000000

Table 5. First 7 data of dataframe to have knowledge data itself.
Source: Own elaboration

4. Exploratory Data Analysis (EDA) in R

The tidyverse package is what we will use for visualizing and exploring our dataset.

It is knows that for easy to read syntax and massive amounts of useful functions. The scales package will be used mainly to customize plot axis. Lastly the arules package will be utilized in the final part of this kernel, Association Rule Learning and Apriori. Info regarding all packages used during this EDA is provided in the Works Cited section in this kernel.

Lets start with overview of the entire dataset.

```
User_ID             Product_ID     Gender        Age
Min.   :1000001   P00265242:  1858   F:132197   0-17 : 14707
1st Qu.:1001495   P00110742:  1591   M:405380   18-25: 97634
Median :1003031   P00025442:  1586              26-35:214690
Mean   :1002992   P00112142:  1539              36-45:107499
3rd Qu.:1004417   P00057642:  1430              46-50: 44526
Max.   :1006040   P00184942:  1424              51-55: 37618
(Other)  :528149              55+  : 20903
Occupation     City_Category Stay_In_Current_City_Years Marital_Status
Min.   : 0.000   A:144638       0 : 72725                  Min.   :0.0000
1st Qu.: 2.000   B:226493       1 :189192                  1st Qu.:0.0000
Median : 7.000   C:166446       2 : 99459                  Median :0.0000
Mean   : 8.083                  3 : 93312                  Mean   :0.4088
3rd Qu.:14.000                  4+: 82889                  3rd Qu.:1.0000
Max.   :20.000                                             Max.   :1.0000
```

Product_Category_1	Product_Category_2	Product_Category_3	Purchase
Min. : 1.000	Min. : 2.00	Min. : 3.0	Min. : 185
1st Qu.: 1.000	1st Qu.: 5.00	1st Qu.: 9.0	1st Qu.: 5866
Median : 5.000	Median : 9.00	Median :14.0	Median : 8062
Mean : 5.296	Mean : 9.84	Mean :12.7	Mean : 9334
3rd Qu.: 8.000	3rd Qu.:15.00	3rd Qu.:16.0	3rd Qu.:12073
Max. :18.000	Max. :18.00	Max. :18.0	Max. :23961
NA's :166986	NA's :373299		

Table 6 . Overview of the entire dataset.
Source: Own elaboration.

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
1000001	P00069042	F	0-17	10	A	2	0	3	NA	NA	8370
1000001	P00248942	F	0-17	10	A	2	0	1	6	14	15200
1000001	P00087842	F	0-17	10	A	2	0	12	NA	NA	1422
1000001	P00085442	F	0-17	10	A	2	0	12	14	NA	1057
1000002	P00285442	M	55+	16	C	4+	0	8	NA	NA	7969
1000003	P00193542	M	26-35	15	A	3	0	1	2	NA	15227

Table 7 . Overview of the entire dataset.
Source: Own elaboration.

We have 12 different columns, each representing a corresponding variable below.

- User_ID: Unique identifier of shopper.
- Product_ID: Unique identifier of product. (No key given)
- Gender: Sex of shopper.
- Age: Age of shopper split into bins.
- Occupation: Occupation of shopper. (No key given)
- City_Category: Residence location of shopper. (No key given)

- Stay_In_Current_City_Years: Number of years stay in current city.
- Marital_Status: Marital status of shopper.
- Product_Category_1: Product category of purchase.
- Product_Category_2: Product may belong to other category.
- Product_Category_3: Product may belong to other category.
- Purchase: Purchase amount in dollars.

If we look at the first few rows of our dataset, we can see that each row represents a different transaction or item purchased by a specific customer. When we group all transactions by a specific User_ID to get a sum of all purchases made by a single customer.

One critique we can make regarding this dataset is that there isn't a key given regarding the different Product_IDs and the item they represent. (Ie. We can't attribute P00265242 to an item easily recognizable) In reality, we would want to have another dataset which provides the name of an Item and its Product_ID and then join it to our existing dataset. This won't necessarily affect our EDA, but would be more useful during our implementation of the Apriori algorithm and could make some parts of the EDA clearer to interpret.

4.1 Gender

To begin our exploration lets examine the gender of shoppers at this store.

Since each row represents an individual transaction, we must first group the data by User_ID to remove duplicates.

```
dataset_gender = dataset %>%
                    select(User_ID, Gender) %>%
                    group_by(User_ID) %>%
                    distinct()

head(dataset_gender)

summary(dataset_gender$Gender)
```

User_ID	Gender
1000001	F
1000002	M
1000003	M
1000004	M
1000005	M
1000006	F

Table 8. Duplictes.
Source: Own elaboration.

F	1666
M	4225

Table 9. F and M in data Duplictes.
Source: Own elaboration.

We have the dataframe necessary to see each User_IDs corresponding gender and their total counts for reference, lets plot the distribution of gender across our dataset.

```
options(scipen=10000)    # To remove scientific numbering

genderDist  = ggplot(data = dataset_gender) +
                geom_bar(mapping = aes(x = Gender, y = ..count.., fill = Gender)) +
                labs(title = 'Gender of Customers') +
                scale_fill_brewer(palette = 'PuBuGn')
print(genderDist)
```

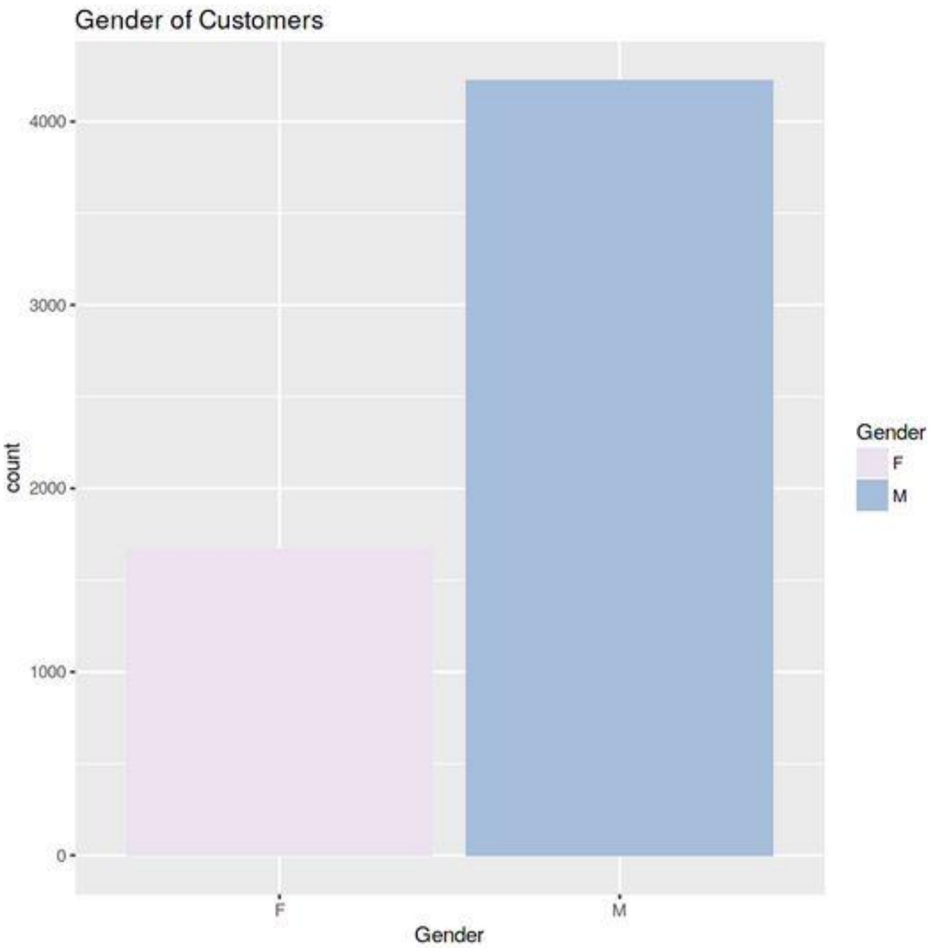


Figure 2. Gender plot.
Source: Own elaboration.

We can see there are quite a few more males than females shopping at our store on Black Friday. This gender split metric is helpful to retailers because some might want to modify their store layout product selection, and other variables differently depending on the gender proportion of their shoppers. (Figure 2)

When we study published in the Clothing and Textiles Research Journal writes,

- "Involvement, variety seeking, and physical environment of stores were selected as antecedents of shopping experience satisfaction....The structural model for female subjects confirmed the existence of the mediating role of hedonic shopping value in shopping satisfaction, whereas the model for male respondents did not." Chang, E., Burns, L. D., & Francis, S. K. (2004) (Abstract)[10]

Although this does not give direct insight into recommended actions for retail stores, it does display a difference in the value derived from shopping and its relationship to gender, which should be taken into account by retailers.
To investigate further, lets compute the average spending amount as it relates to Gender.
For easy interpretation and traceback we will create separate tables and then join them together.

```
total_purchase_user = dataset %>%
                        select(User_ID, Gender, Purchase) %>%
                        group_by(User_ID) %>%
                        arrange(User_ID) %>%
                        summarise(Total_Purchase = sum(Purchase))

user_gender = dataset %>%
                select(User_ID, Gender) %>%
                group_by(User_ID) %>%
                arrange(User_ID) %>%
                distinct()

head(user_gender)
head(total_purchase_user)
```

User_ID	Gender
1000001	F
1000002	M
1000003	M
1000004	M
1000005	M
1000006	F

Table 11. Total Purchase
Source: Own elaboration

[10] Chang, E., Burns, L. D., & Francis, S. K. (2004) (Abstract)

User_ID	Total_Purchase
1000001	333481
1000002	810353
1000003	341635
1000004	205987
1000005	821001
1000006	379450

Table 11a. Total Purchase
Source: Own elaboration.

```
user_purchase_gender = full_join(total_purchase_user, user_gender, by
= "User_ID")
head(user_purchase_gender)
```

User_ID	Total_Purchase	Gender
1000001	333481	F
1000002	810353	M
1000003	341635	M
1000004	205987	M
1000005	821001	M
1000006	379450	F

Table 12. Join Table 2 and Table 3 together.
Source: Own elaboration.

```
average_spending_gender = user_purchase_gender %>%
                           group_by(Gender) %>%
                           summarize(Purchase = sum(as.numeric(Total_Purchase)),
                                     Count = n(),
                                     Average = Purchase/Count)
head(average_spending_gender)
```

Gender	Purchase	Count	Average
F	1164624021	1666	699054.0
M	3853044357	4225	911963.2

Table 13. Join Purchase and Average Together
Source: Own elaboration.

```
genderAverage  = ggplot(data = average_spending_gender) +
                   geom_bar(mapping = aes(x = Gender, y = Average, fill = Gender),
stat = 'identity') +
```

```
                    labs(title = 'Average Spending by Gender') +
                    scale_fill_brewer(palette = 'PuBuGn')
print(genderAverage)
```

Looks like our top 5 best sellers are (by product ID)

- P00265242 = 1858
- P00110742 = 1591
- P00025442 = 1586
- P00112142 = 1539
- P00057642 = 1430

When we have Identified our top 5 best selling products, lets examine the best selling product, P00265242.

```
best_seller = dataset[dataset$Product_ID == 'P00265242', ]


head(best_seller)
```

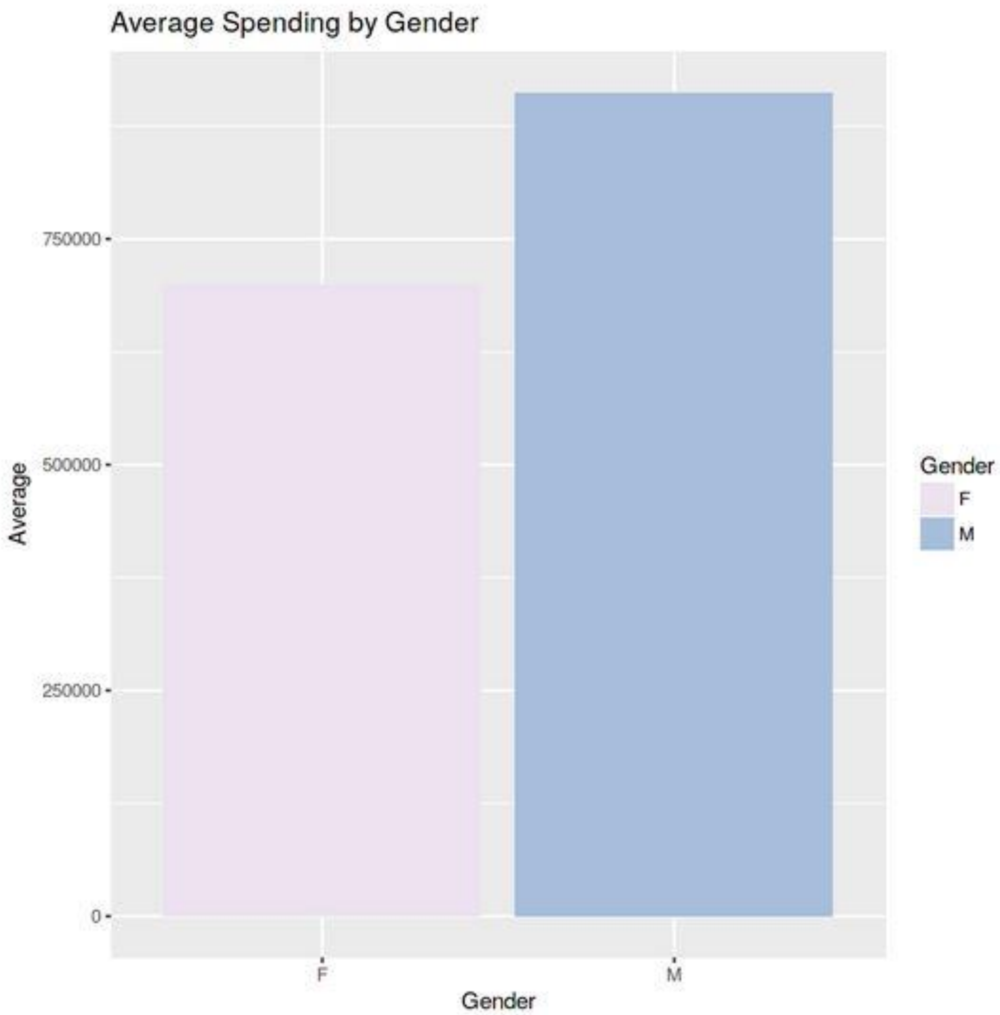


Figure 3. Average spending by Gender plot.
Source: Own elaboration.

We present's an interesting observation. Even though female shoppers make less purchases than males at this specific store, they seem to be purchasing almost as much on average as the male shoppers. This being said, scale needs to be taken into account because females on average are still spending about 250,000 less than males. [Figure 3]

4.2 Top Sellers

Now lets switch gears and examine our top selling products. In this situation, we won't group by product ID since we want to see duplicates, just in case people are buying 2 or more quantities of the same product.

```
top_sellers = dataset %>%
                count(Product_ID, sort = TRUE)

top_5 = head(top_sellers, 5)

top_5
```

Looks like our top 5 best sellers are (by product ID)

- P00265242 = 1858
- P00110742 = 1591
- P00025442 = 1586
- P00112142 = 1539
- P00057642 = 1430

Now that we have Identified our top 5 best selling products, lets examine the best selling product, P00265242.

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
400	1000066	P00265242	M	26-35	18	C	2	0	5	8	NA	8652
1192	1000196	P00265242	F	36-45	9	C	4+	0	5	8	NA	8767
1373	1000222	P00265242	M	26-35	1	A	1	0	5	8	NA	6944
1846	1000301	P00265242	M	18-25	4	B	4+	0	5	8	NA	8628
2210	1000345	P00265242	M	26-35	12	A	2	1	5	8	NA	8593
2405	1000383	P00265242	F	26-35	7	A	4+	1	5	8	NA	6998

Table 13. 5 best selling products.
Source: Own elaboration.

We can see that this product fits into Product_Category_1 = 5 and Product_Category_2 = 8.
As mentioned in the introduction, it would be useful to have a key to reference the item name in order to determine what it is. [Table 13]
Another interesting finding is that even though people are purchasing the same product they are paying different prices.
This could be due to various Black Friday promotions, discounts, or coupon codes. Otherwise, investigation would need to be done regarding the reason for different purchase prices of the same product between customers.

Lets continue to analyze our best seller to see if any relationship to Gender exits.

```
genderDist_bs  = ggplot(data = best_seller) +
                  geom_bar(mapping = aes(x = Gender, y = ..count.., fill = Gender))
+
                  labs(title = 'Gender of Customers (Best Seller)') +
                  scale_fill_brewer(palette = 'PuBuGn')
print(genderDist_bs)
```

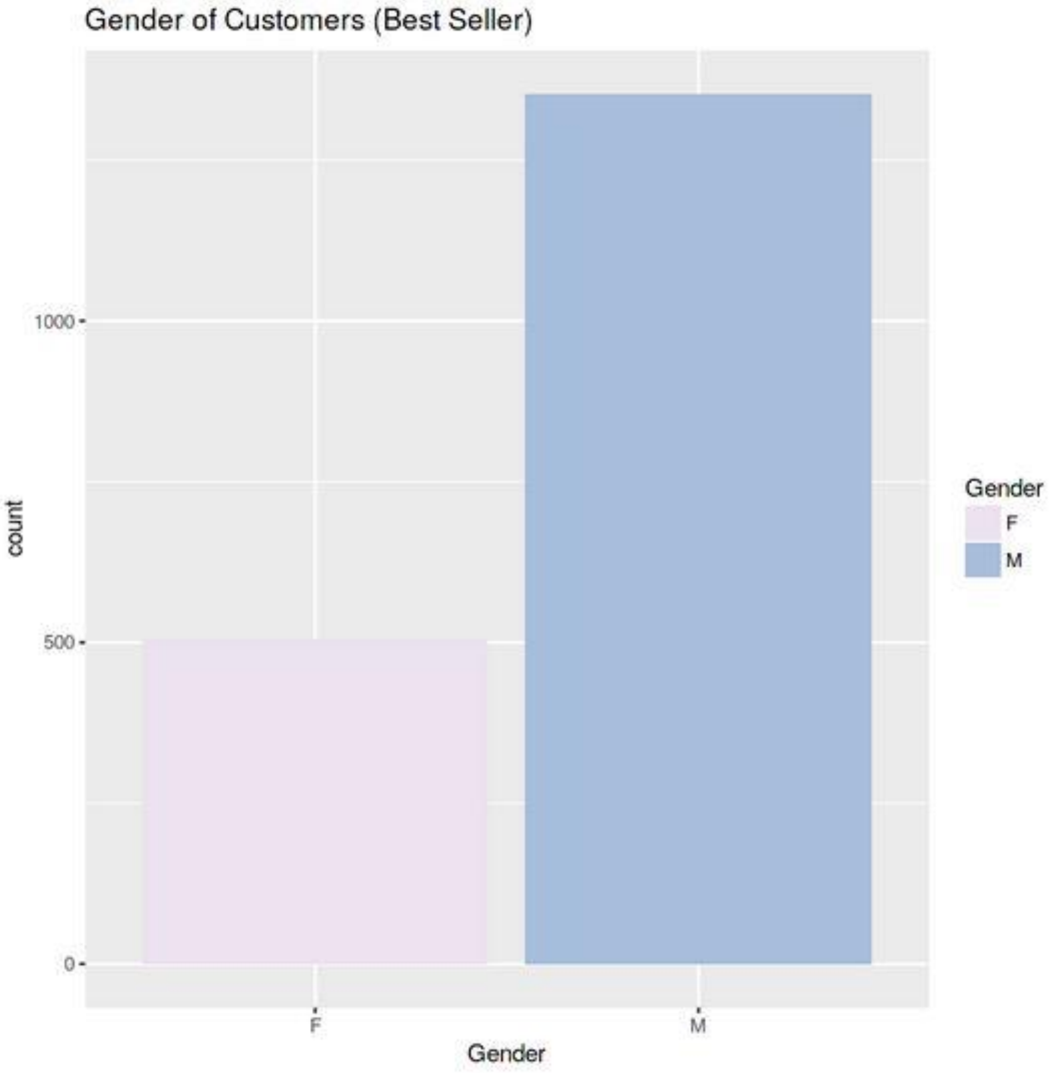


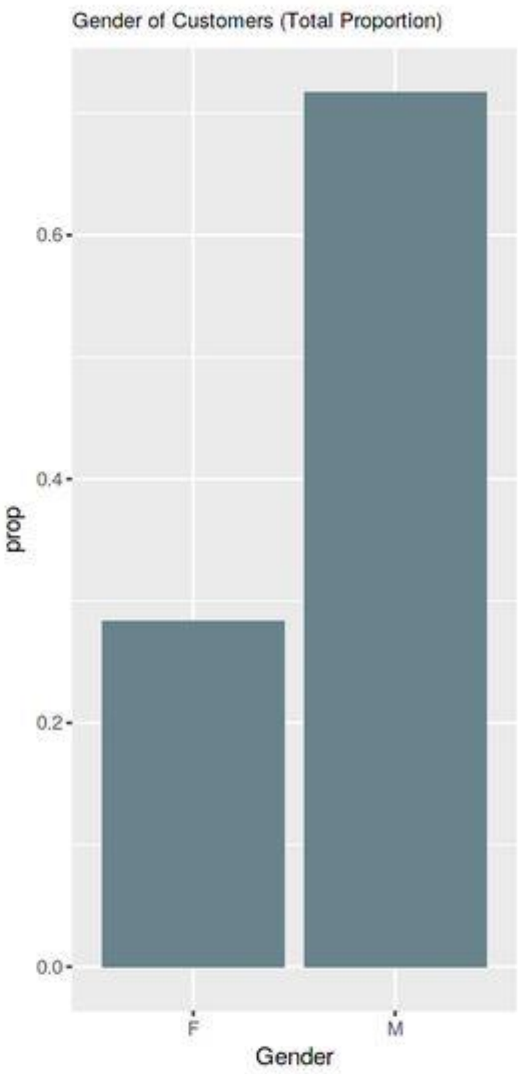
Figure 4. Count and Gender plot
Source: Own elaboration.

We see a similar distribution between genders to our overall dataset gender split - lets confirm.

```
genderDist_bs_prop = ggplot(data = best_seller) +
                          geom_bar(fill = 'lightblue', mapping = aes(x = Gender, y =
..prop.., group = 1, fill = Gender)) +
                          labs(title   =   'Gender   of   Customers   (Best   Seller   -
Proportion)') +
                          theme(plot.title = element_text(size=9.5))

genderDist_prop = ggplot(data = dataset_gender) +
                      geom_bar(fill  =  "lightblue4",  mapping  =  aes(x  =  Gender,  y  =
..prop.., group = 1)) +
                      labs(title = 'Gender of Customers (Total Proportion)') +
                      theme(plot.title = element_text(size=9.5))

grid.arrange(genderDist_prop, genderDist_bs_prop, ncol=2)
```



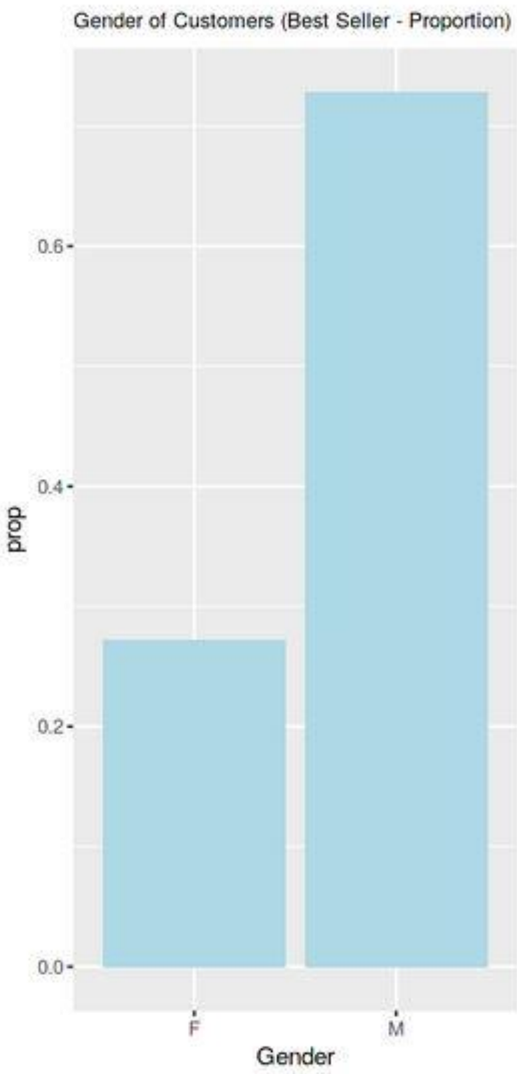


Figure 5. Proportion of customers Figure 6. Proportion of customers
Source: Own elaboration.

We can see that between the overall observation set, both purchasers of the best seller and purchasers of all products are roughly ~25% female and ~75% male. A slight difference does exist but it seems like we can generally conclude that our best seller does not cater to a specific gender. [Figure 5 and Figure 6]

4.3 Age

Lets begin examining Age by creating a table of each individual age group and their respective counts. [Table 13]

```
customers_age = dataset %>%
                    select(User_ID, Age) %>%
                    distinct() %>%
                    count(Age)
customers_age
```

Age	n
0-17	218
18-25	1069
26-35	2053
36-45	1167
46-50	531
51-55	481
55+	372

Table 13. Age
Source: Own elaboration.

We can see a dataset that shows the count of each Age category of customers at our store.

```
customers_age_vis = ggplot(data = customers_age) +
                      geom_bar(color = 'black', stat = 'identity',
mapping = aes(x = Age, y = n, fill = Age)) +
                      labs(title = 'Age of Customers') +
                      theme(axis.text.x = element_text(size = 10)) +
                      scale_fill_brewer(palette = 'Blues') +
                      theme(legend.position="none")
print(customers_age_vis)
```

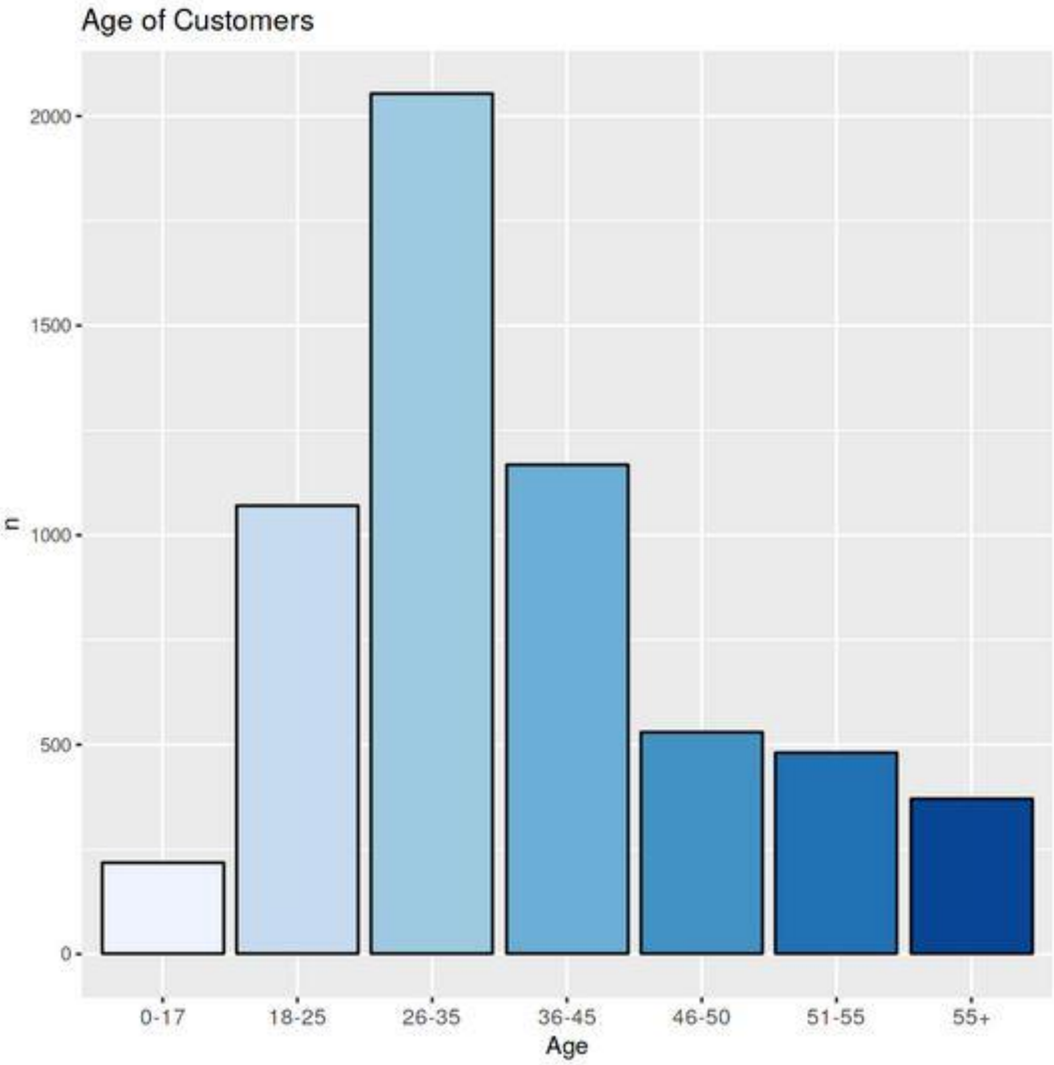


Figure 7. Age of customers
Source: Own elaboration.

We can also plot a similar chart depicting the distribution of age within our "best seller" category. This will show us if there is a specific age category that purchased the best selling product more than other shoppers. [Figure 7]

```
ageDist_bs  = ggplot(data = best_seller) +
                 geom_bar(color = 'black', mapping = aes(x = Age, y
= ..count.., fill = Age)) +
                 labs(title = 'Age of Customers (Best Seller)') +
                 theme(axis.text.x = element_text(size = 10)) +
                 scale_fill_brewer(palette = 'GnBu') +
                 theme(legend.position="none")
print(ageDist_bs)
```

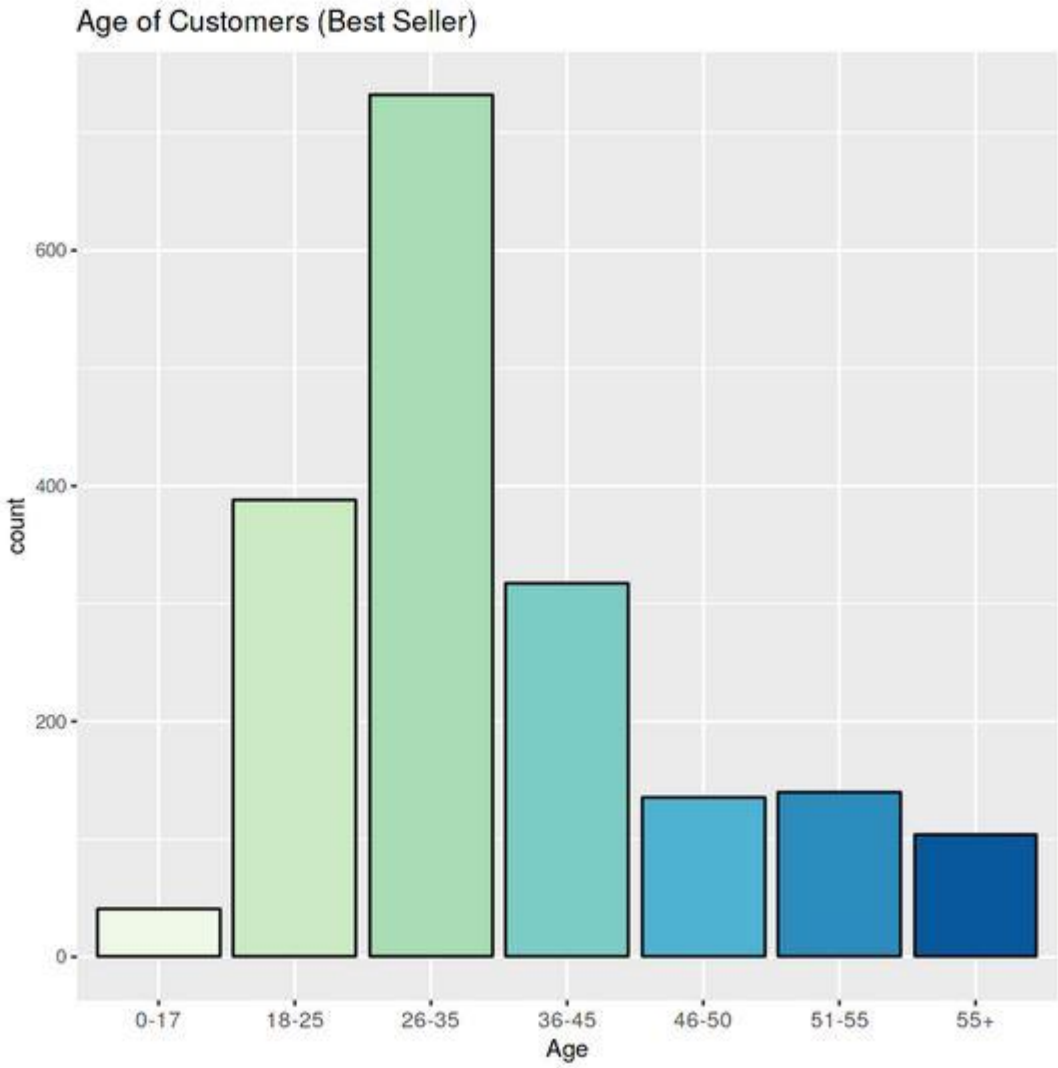
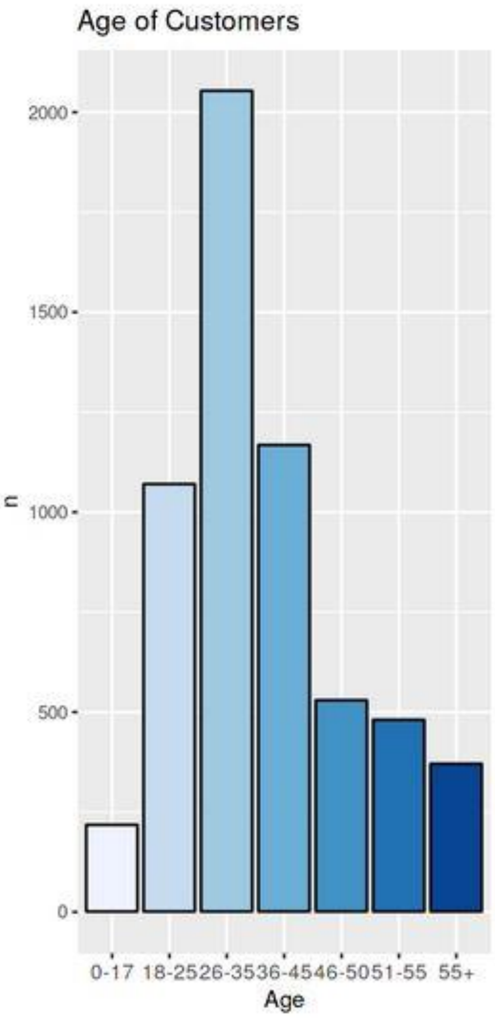


Figure 8. Age of customers best seller.
Source: Own elaboration.

It seems as though younger people (18-25 & 26-35) account for the highest number of purchases of the best selling product. Lets compare this observation to the overall dataset. [Figure 8]

```
grid.arrange(customers_age_vis, ageDist_bs, ncol=2)
```



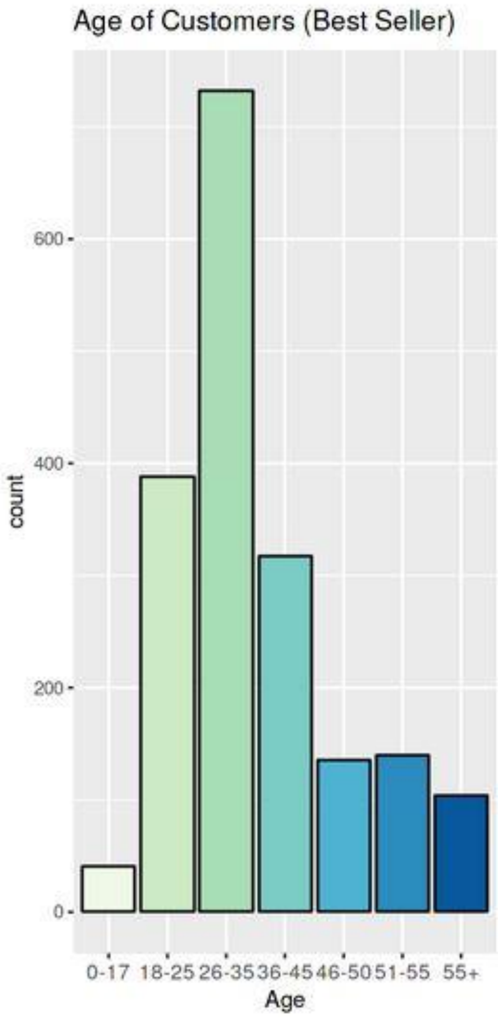


Figure 9. Age of customers best seler

Figure 10. Age of customers best seller

Source: Own elaboration.

We can see that there is some deviation with the proportion of customers grouped by age when comparing the best selling product to the overall dataset. It looks like older customers > Age 45 are buying the top seller slightly less than other products included in the overall dataset. [Figure 9, Figure 10]

4.4 Purchase

Now lets do some investigation regarding store customers and their purchases. We will start by computing the total purchase amount by user ID

```
customers_total_purchase_amount = dataset %>%
                                    group_by(User_ID) %>%
                                    summarise(Purchase_Amount = sum(Purchase))

head(customers_total_purchase_amount)
```

User_ID	Purchase_Amount
1000001	333481
1000002	810353
1000003	341635
1000004	205987
1000005	821001
1000006	379450

Table 14. Purchase Amount

Source: Own elaboration.

Now that we have grouped our purchases and grouped by User ID, we will sort and find our top spenders. [Table 14]

```
customers_total_purchase_amount = arrange(customers_total_purchase_amount,
desc((Purchase_Amount)))

head(customers_total_purchase_amount)
```

User_ID	Purchase_Amount
1004277	10536783
1001680	8699232
1002909	7577505
1001941	6817493
1000424	6573609
1004448	6565878

Table 15. Purchase Amount

Source: Own elaboration.

Looks like User ID 1004277 is our top spender. Lets use summary() to see other facets of our total customer spending data. [Table 15]

```
summary(customers_total_purchase_amount)
    User_ID          Purchase_Amount
 Min.   :1000001   Min.   :   44108
 1st Qu.:1001518   1st Qu.:  234914
 Median :1003026   Median :  512612
 Mean   :1003025   Mean   :  851752
 3rd Qu.:1004532   3rd Qu.: 1099005
 Max.   :1006040   Max.   :10536783
```

Table 16. Customers total purchase amount
Source: Own elaboration.

We can see an average total purchase amount of 851752, max total purchase amount of 10536783, min total purchase amount of 44108 and a median purchase amount of 512612.[Table 16]

Lets plot a chart showing the distribution of purchase amounts to see if purchases are normally distributed or contain some skewness. A density plot will show us where the highest number of similar purchase amounts rests in accordance to the entire customer base. It is important to note that Density charts graph the expected probability of values, given data as input, and then plot a line surrounding those values (estimation).

```
ggplot(customers_total_purchase_amount, aes(Purchase_Amount)) +
  geom_density(adjust = 1) +
```

```
  geom_vline(aes(xintercept=median(Purchase_Amount)),
             color="blue", linetype="dashed", size=1) +
  geom_vline(aes(xintercept=mean(Purchase_Amount)),
             color="red", linetype="dashed", size=1) +
  geom_text(aes(x=mean(Purchase_Amount), label=round(mean(Purchase_Amount)), y=1.2e-
06), color = 'red', angle=360,
            size=4, vjust=3, hjust=-.1) +
  geom_text(aes(x=median(Purchase_Amount),     label=round(median(Purchase_Amount)),
y=1.2e-06), color = 'blue', angle=360,
            size=4, vjust=0, hjust=-.1) +
  scale_x_continuous(name="Purchase   Amount",   limits=c(0,   7500000),   breaks   =
seq(0,7500000, by = 1000000), expand = c(0,0)) +
  scale_y_continuous(name="Density",  limits=c(0,  .00000125),  labels  =  scientific,
expand = c(0,0))
Warning message:
"Removed 3 rows containing non-finite values (stat_density)."
```

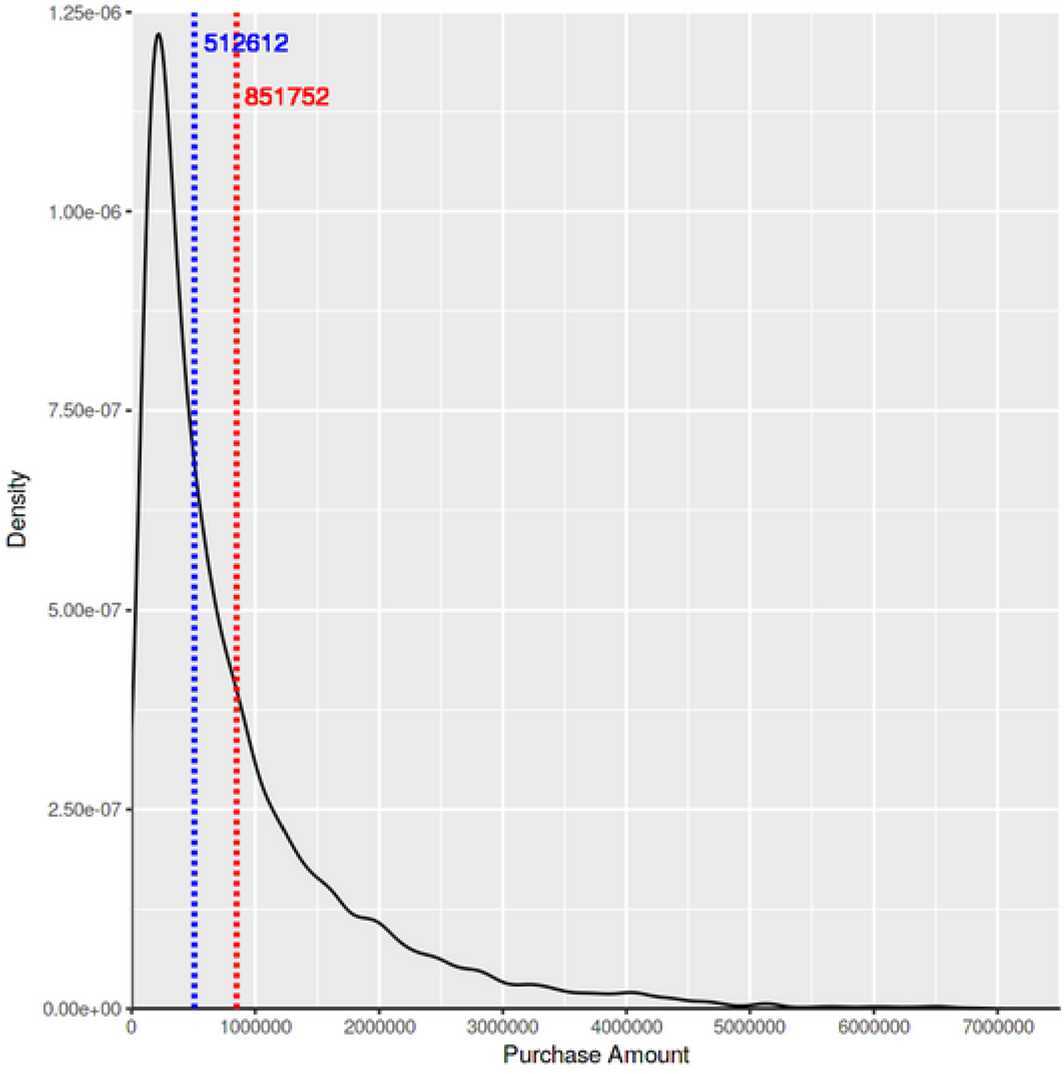


Figure 11. Purchase Amount to Density

Source: Own elaboration.

We see a very right (positive) skewed density plot with a long tail. This means that there are quite a few values that sit higher than the mean and that the highest density of values isn't a standardly distributed series. We see that the largest density of purchases is around the 250000 mark.[Figure 11]

4.5 Marital Status

Now examine the marital status of store customers.

```
dataset_maritalStatus = dataset %>%
                            select(User_ID, Marital_Status) %>%
                            group_by(User_ID) %>%
                            distinct()

head(dataset_maritalStatus)
```

User_ID	Marital_Status
1000001	0
1000002	0
1000003	0
1000004	1
1000005	1
1000006	0

Table 17. Marital status
Source: Own elaboration.

Note, we need to quickly change Marital_Status from a numeric variable to a categorical type.

```
dataset_maritalStatus$Marital_Status                                                    =
as.character(dataset_maritalStatus$Marital_Status)
typeof(dataset_maritalStatus$Marital_Status)
```
'character'

If we look back at the variable descriptions of the dataset, we don't have a clear guide for marital status. In other cases, it would be best to reach out to the provider of the data to be completely sure of what the values in a column represent but in this case, we will assume that 1 = married and 0 = single.[Table 17]

```
marital_vis = ggplot(data = dataset_maritalStatus) +
                    geom_bar(mapping = aes(x = Marital_Status, y =
..count.., fill = Marital_Status)) +
                    labs(title = 'Marital Status') +
                    scale_fill_brewer(palette = 'Pastel2')
print(marital_vis)
```

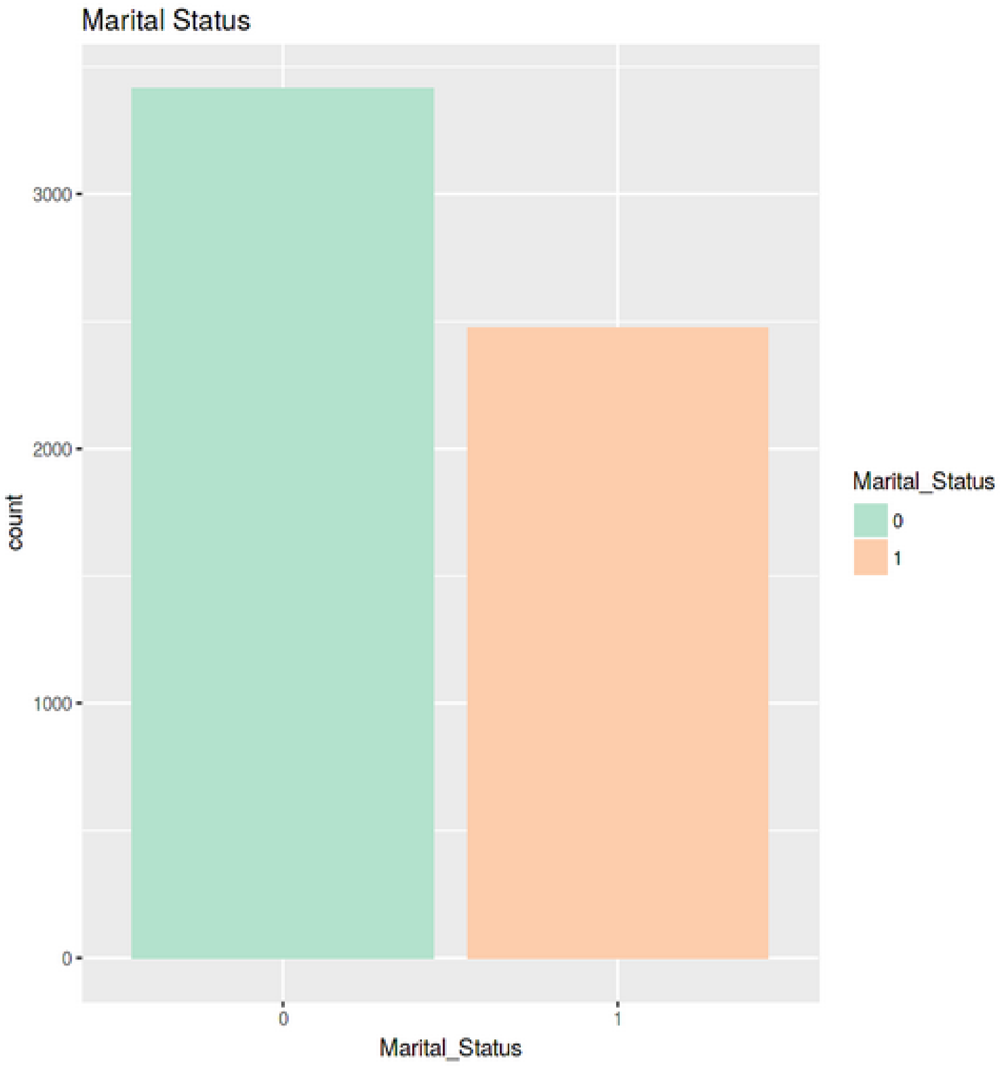


Figure 12. Marital status.

Source: Own elaboration.

It looks like most of our shoppers happen to be single or unmarried. Similar to our investigation of age groups, we can look at the makeup of Marital_Status in each City_Category. [Figure 12]

```
dataset_maritalStatus = dataset_maritalStatus %>%
                            full_join(customers_stay, by = 'User_ID')
head(dataset_maritalStatus)
```

User_ID	Marital_Status	City_Category	Stay_In_Current_City_Years
1000001	0	A	2
1000002	0	C	4+
1000003	0	A	3
1000004	1	B	2
1000005	1	A	1
1000006	0	A	1

Table 18. Full join (customers_stay, by = 'User_ID'

Source: Own elaboration.

```
maritalStatus_cities = dataset_maritalStatus %>%
                        group_by(City_Category, Marital_Status) %>%
                        tally()
head(maritalStatus_cities)
```

City_Category	Marital_Status	n
A	0	652
A	1	393
B	0	1004
B	1	703
C	0	1761
C	1	1378

Table.19 City Category
Source: Own elaboration.

```
ggplot(data = maritalStatus_cities, aes(x = City_Category, y = n, fill =
Marital_Status)) +
    geom_bar(stat = "identity", color = 'black') +
    scale_fill_brewer(palette = 2) +
    labs(title = "City + Marital Status",
```

```
y = "Total Count (Shoppers)", x = "City",  fill = "Marital Status")
```

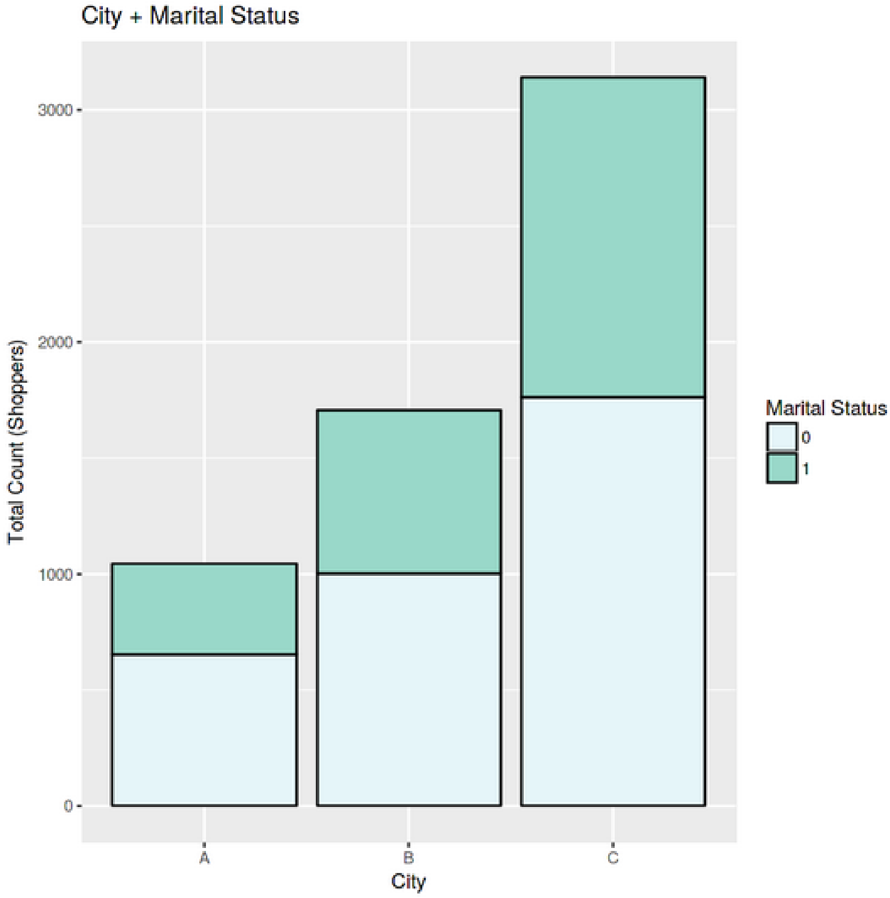


Figure 13. City
Source: Own elaboration.

Here, we can see that out off all Cities, the highest proportion of single shoppers seems to be in City A. Now, lets investigate the Stay_in_Current_City distribution within each City_Category. [Figure 13]

```
Users_Age = dataset %>%
                select(User_ID, Age) %>%
                distinct()
head(Users_Age)
```

User_ID	Age
1000001	0-17
1000002	55+
1000003	26-35
1000004	46-50
1000005	26-35
1000006	51-55

Table 20. ID Age

Source: Own elaboration.

```
dataset_maritalStatus = dataset_maritalStatus %>%
                            full_join(Users_Age, by = 'User_ID')
head(dataset_maritalStatus)
```

User_ID	Marital_Status	City_Category	Stay_In_Current_City_Years	Age
1000001	0	A	2	0-17
1000002	0	C	4+	55+
1000003	0	A	3	26-35
1000004	1	B	2	46-50
1000005	1	A	1	26-35
1000006	0	A	1	51-55

Table 21. Data prom marital status for User Id
Source: Own elaboration.

```
City_A = dataset_maritalStatus %>%
            filter(City_Category == 'A')
City_B = dataset_maritalStatus %>%
            filter(City_Category == 'B')
City_C = dataset_maritalStatus %>%
            filter(City_Category == 'C')
head(City_A)
head(City_B)
head(City_C)
```

User_ID	Marital_Status	City_Category	Stay_In_Current_City_Years	Age
1000001	0	A	2	0-17
1000003	0	A	3	26-35
1000005	1	A	1	26-35
1000006	0	A	1	51-55
1000015	0	A	1	26-35
1000019	0	A	3	0-17
1000004	1	B	2	46-50
1000007	1	B	1	36-45
1000010	1	B	4+	36-45
1000018	0	B	3	18-25
1000021	0	B	0	18-25
1000023	1	B	3	36-45
1000002	0	C	4+	55+
1000008	1	C	4+	26-35
1000009	0	C	0	26-35
1000011	0	C	1	26-35
1000012	0	C	2	26-35
1000013	1	C	3	46-50

Table 22. Data prom City for User Id.
Source: Own elaboration.

```
City_A_stay_vis = ggplot(data = City_A, aes(x = Age, y = ..count.., fill = Age)) +
                              geom_bar(stat = 'count') +
                              scale_fill_brewer(palette = 8) +
                              theme(legend.position="none",       axis.text       =
element_text(size = 6)) +
                              labs(title = 'City A', y = 'Count', x = 'Age', fill =
'Age')
City_B_stay_vis = ggplot(data = City_B, aes(x = Age, y = ..count.., fill = Age)) +
                              geom_bar(stat = 'count') +
                              scale_fill_brewer(palette = 9) +
                              theme(legend.position="none",       axis.text       =
element_text(size = 6)) +
                              labs(title = 'City B', y = 'Count', x = 'Age', fill =
'Age')
City_C_stay_vis = ggplot(data = City_C, aes(x = Age, y = ..count.., fill = Age)) +
                              geom_bar(stat = 'count') +
                              scale_fill_brewer(palette = 11) +
```

```
theme(legend.position="none", axis.text = element_text(size = 6)) +
labs(title = 'City C', y = 'Count', x = 'Age', fill = 'Age')
grid.arrange(City_A_stay_vis, City_B_stay_vis, City_C_stay_vis, ncol = 3)
```

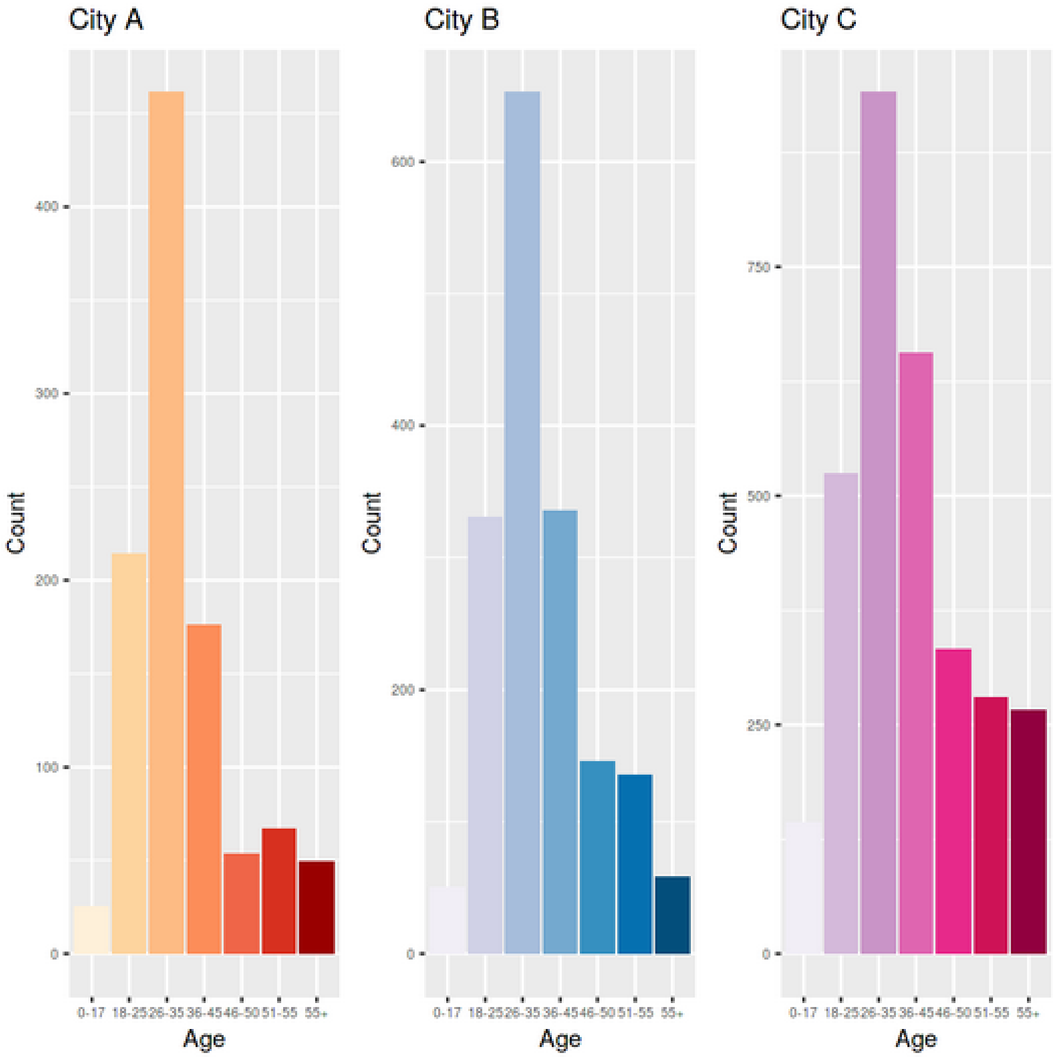


Figure 12. Count for Age

Source: Own elaboration.

It looks as though City A has less shoppers living there over the age of 45 compared to the other cities. This could be a factor in the resulting levels of Marital_Status within each individual city.[Figure 12]

4.5 Top Shoppers

Now we will investigate who our top shoppers were on Black Friday.

```
top_shoppers = dataset %>%
                count(User_ID, sort = TRUE)

head(top_shoppers)
```

User_ID	n
1001680	1025
1004277	978
1001941	898
1001181	861
1000889	822
1003618	766

Table 23. Data User Id

Source: Own elaboration.

Looks like User_ID 1001680 shows up the most on our master ledger of shopper data. Since each individual row represents a different transaction/product, it looks like this user made over 1000 total transactions! We can join together this top shoppers dataset with our total customer purchases dataset to see them combined.[Table 23]

```
top_shoppers =  top_shoppers %>%
                    select(User_ID, n) %>%
                    left_join(customers_total_purchase_amount, Purchase_Amount, by =
'User_ID')

head(top_shoppers)
```

User_ID	n	Purchase_Amount
1001680	1025	8699232
1004277	978	10536783
1001941	898	6817493
1001181	861	6387899
1000889	822	5499812
1003618	766	5961987

Table 24. Data User Id Purchase_Amount

Source: Own elaboration.

Now that we have joined the two tables together, we can see that although User_ID 1001680 has the highest number of total purchases, User_ID 1004277 has the highest Purchase_Amount as identified in our earlier charts as well. From here, we can also compute the average Purchase_Amount for each user.[Table 24]

```
top_shoppers = mutate(top_shoppers,
                      Average_Purchase_Amount = Purchase_Amount/n)

head(top_shoppers)
```

User_ID	n	Purchase_Amount	Average_Purchase_Amount
1001680	1025	8699232	8487.056
1004277	978	10536783	10773.807
1001941	898	6817493	7591.863
1001181	861	6387899	7419.163
1000889	822	5499812	6690.769
1003618	766	5961987	7783.273

Table 25. Data User Id Purchase Amount

Source: Own elaboration.

Now, we can sort according to Average_Purchase_Amount to see which customers, on average, are spending the most.[Table 25]

```
top_shoppers_averagePurchase = top_shoppers %>%
                                    arrange(desc(Average_Purchase_Amount))
```

```
head(top_shoppers_averagePurchase)
```

User_ID	n	Purchase_Amount	Average_Purchase_Amount
1005069	16	308454	19278.38
1003902	93	1746284	18777.25
1005999	18	330227	18345.94
1001349	23	417743	18162.74
1000101	65	1138239	17511.37
1003461	20	350174	17508.70

Table 26. Data User Id Purchase_Amount Average_Purchase_Amount

Source: Own elaboration.

Looks like User_ID 1005069 has the highest Average_Purchase_Amount and a total Purchase_Amount of 308454. User_ID 1003902 is right behind User_ID 1005069 in Average_Purchase_Amount, but has a much higher total Purchase_Amount of 1746284.

4.6 Occupation

We will analyze is the occupation of customers in our dataset.

```
customers_Occupation =  dataset %>%
                          select(User_ID, Occupation) %>%
                          group_by(User_ID) %>%
                          distinct() %>%
                          left_join(customers_total_purchase_amount, Occupation, by
= 'User_ID')

head(customers_Occupation)
```

User_ID	Occupation	Purchase_Amount
1000001	10	333481
1000002	16	810353
1000003	15	341635
1000004	7	205987
1000005	20	821001
1000006	9	379450

Table 27. Data Occupation and Purchase Amount

Source: Own elaboration.

Now that we have our dataset necessary, we can group together the total Purchase_Amount for each Occupation identifier. We will then convert Occupation to a charater data type [Table 27].

```
totalPurchases_Occupation = customers_Occupation %>%
                              group_by(Occupation) %>%
                              summarise(Purchase_Amount = sum(Purchase_Amount)) %>%
                              arrange(desc(Purchase_Amount))

totalPurchases_Occupation$Occupation                                              =
as.character(totalPurchases_Occupation$Occupation)
typeof(totalPurchases_Occupation$Occupation)

head(totalPurchases_Occupation)
'character'
```

Occupation	Purchase_Amount
4	657530393
0	625814811
7	549282744
1	414552829
17	387240355
12	300672105

Table 28. Data Occupation and Purchase Amount
Source: Own elaboration.

Now, lets plot each occupation and their total [Table 28]Purchase_Amount:

```
occupation = ggplot(data = totalPurchases_Occupation) +
                  geom_bar(mapping = aes(x = reorder(Occupation, -Purchase_Amount),
y = Purchase_Amount, fill = Occupation), stat = 'identity') +
                  scale_x_discrete(name="Occupation", breaks = seq(0,20, by = 1),
expand = c(0,0)) +
                  scale_y_continuous(name="Purchase Amount ($)", expand = c(0,0),
limits = c(0, 750000000)) +
                  labs(title = 'Total Purchase Amount by Occupation') +
                  theme(legend.position="none")
print(occupation)
```

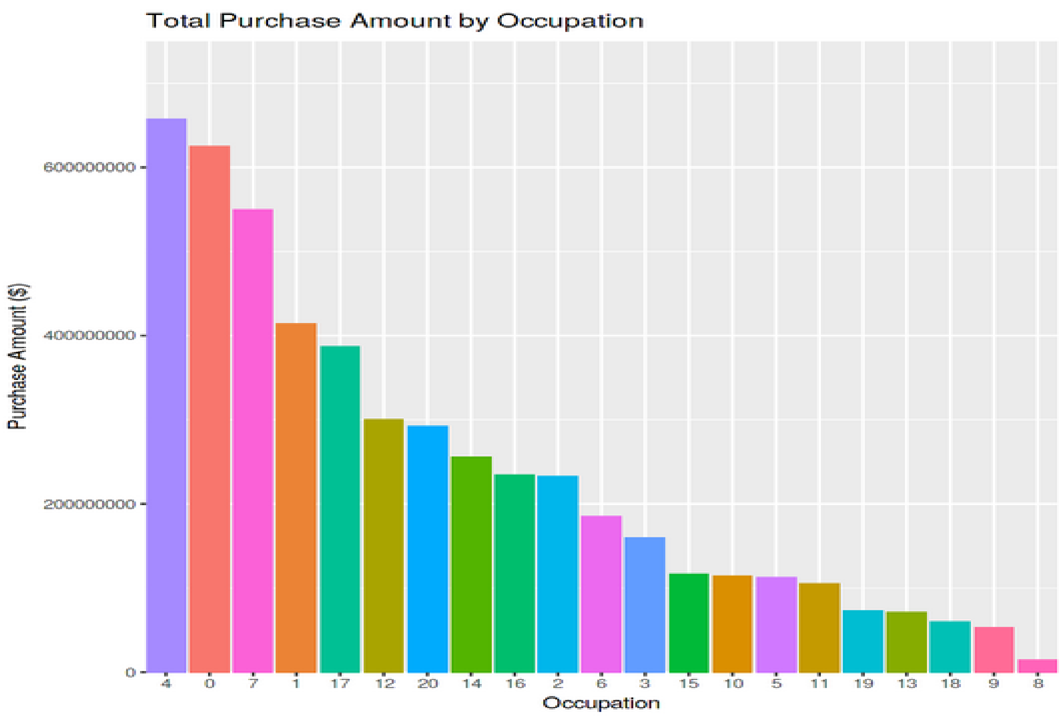


Figure 13. Total purchase amount by occupation
Source: Own elaboration.

Looks like customers labeled as Occupation 4 spent the most at our store on Black Friday, with customers of Occupation 0 + 7 closely behind. Here, if a key was given, we could use that information to classify our shoppers accordingly.[Figure 13]

4.7 Apriori (Association Rule Learning)

Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.[11]
Now lets use a machine learning algorithim called Apriori to make some association rules regarding customer purchases. We will be using the arules package.

Before we begin, lets elaborate on the idea of Association Rule Learning. In its simplest form, Association Rule Learning attempts to predict customer transactions. In other words, the algorithm

[11] https://en.wikipedia.org/wiki/Apriori_algorithm

solves the problem, "People who bought ----- also bought ----- ." This can prove to be extemely useful for retailers who aim to optimize product placement in stores and promotional campaigns.[12]
In the case of our store on Black Friday, implementing an effective product placement strategy can prove to optimize sales of products normally bought together. For example, lets say that our store was to have a sale on TVs. It would be smart to place HDMI Cables alongside these TVs because those items are usually purchased together. On the other hand, it may also prove to be smart to place them far apart so that customers need to walk throughout the entire store while searching for their desired item, where another product may catch their eye along the way.

The Apriori algorithm specifically aims to maximize the likelihood someone performs/purchases something given knowledge about their prior actions.

To begin, lets import the libraries we wil be using for this section if not done so already.

```
library(arules)
library(arulesViz)
library(tidyverse)
Loading required package: grid
```

The arules package was developed specifically to deal with Association Rule and Frequent Itemset mining. In order to begin our analysis, we must retrieve the necessary data from the original dataset and then apply the correct formatting.

```
# Data Preprocessing
# Getting the dataset into the correct format
customers_products = dataset %>%
                        select(User_ID, Product_ID) %>%   # Selecting the columns we
will need
                        group_by(User_ID) %>%             # Grouping by "User_ID"
                        arrange(User_ID) %>%              # Arranging by "User_ID"
                        mutate(id = row_number()) %>%     # Defining a key column for
each "Product_ID" and its corresponding "User_ID" (Must do this for spread() to work
properly)
                        spread(User_ID, Product_ID) %>%    # Converting our dataset
from tall to wide format, and grouping "Product_IDs" to their corresponding "User_ID"
                        t()                                # Transposing the dataset
from columns of "User_ID" to rows of "User_ID"

# Now we can remove the Id row we created earlier for spread() to work correctly.
customers_products = customers_products[-1,]
```

Now, in order for the Apriori algorithm to work correctly, we need to convert the customers_products table into a sparse matrix. Unfortunately, Apriori doesn't take strings or text as input, but rather 1 + 0. (Binary Format) This means that we must allocate a column for each individual product and then if a

[12] https://en.wikipedia.org/wiki/Apriori_algorithm

User_ID contains that product, it will be marked as a 1. On the other hand, if the User_ID does not contain that Product_ID, it wil be marked with a 0.

In order to do so, we need to use the arules library as described above and import the table as a .csv file. From there, we can use the arules function, "read.transactions()" to get our sparse matrix.

```
write.csv(customers_products, file = 'customers_products.csv')

customersProducts = read.transactions('customers_products.csv', sep = ',',
rm.duplicates = TRUE) # remove duplicates with rm.duplicates
distribution of transactions with duplicates:

items
  46  126  163  202  258  272  285  307  310  316  319  327  330  334  340  344
   1    1    1    1    1    1    1    1    1    1    1    2    1    1    1    1
 345  348  354  357  373  393  402  408  419  437  441  449  450  452  454  456
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
 459  465  466  467  475  476  477  481  487  491  495  498  507  523  524  526
   1    1    2    2    1    1    1    1    2    2    3    2    1    1    2    1
 527  528  530  531  532  533  535  537  538  539  540  545  546  548  549  553
   2    1    1    2    1    1    1    1    2    3    1    1    1    3    1    1
 554  555  556  558  563  566  567  570  572  574  575  577  578  580  583  584
   2    1    1    2    1    3    1    3    2    4    1    2    3    2    1    1
 586  588  589  590  591  592  593  594  595  597  598  601  602  604  607  608
   2    3    5    1    1    1    1    1    3    2    1    1    1    1    2    1
 610  612  613  614  615  616  617  618  619  620  623  625  632  633  634  635
   1    2    1    2    1    1    2    1    3    2    1    1    6    1    5    2
 638  640  641  642  643  644  645  646  647  648  653  654  657  658  659  661
   2    2    1    2    1    2    3    1    4    1    1    3    2    1    1    2
 662  663  664  665  666  667  668  669  670  671  672  674  676  677  678  679
   1    3    1    1    2    1    2    4    2    1    3    1    1    2    3    3
 681  682  683  685  686  687  688  689  690  691  692  694  695  697  698  699
   2    2    4    4    5    2    2    1    1    1    1    2    1    1    1    1
 700  702  703  704  705  706  707  708  709  710  712  713  714  715  716  717
   2    1    3    1    1    2    2    3    2    2    4    5    2    2    1    1
 718  719  720  721  722  723  724  725  726  727  728  729  730  732  733  734
   2    3    3    5    2    1    1    2    5    2    1    3    3    2    1    5
 735  736  737  738  739  740  741  742  743  744  745  747  748  749  750  751
   6    2    4    6    3    4    1    6    7    4    6    1    1    5    5    3
 752  753  754  755  756  757  758  759  760  761  763  764  765  766  767  768
   2    3    4    6    6    2    6    2    1    5    3    4    2    3    2    5
 769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
   2    2    3    1    3    4    2    2    3    3    2    4    7    3    4    5
 785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
   5    3    3    6    5    5    2    3    5    3    8    5    5    9    3    4
 801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
   4    7    4    3    4    5    7    5    4    5    3    2    6    6    3   11
 817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832
   5   10    6    6    4    7    7    2    5    4    7    5    5    4    5    4
 833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848
   3    5    4   11    5    5    4    9    7    6    4    7    9   11    4    6
 849  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864
  10    6   10    7   12   16   11    8    7    4   12    9   11   11    9    6
 865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880
  11   10    7    6    5   12    6    7    8   11    9    9    8    7    5    4
 881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896
  15   13   12    8    4    6   12   15   13   10   11   13    6   21    7   14
 897  898  899  900  901  902  903  904  905  906  907  908  909  910  911  912
   9    7   11   18    5   14   10    9   19   15   10   17   18   23    8   19
 913  914  915  916  917  918  919  920  921  922  923  924  925  926  927  928
  15   12   18   21   17   12   11   13   13   12   20   20   16   13   15   17
 929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944
  27   22   20   28   18   14   20   20   20   14   22   30   23   23   21   20
 945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960
```

```
  25   19   30   31   30   24   27   25   40   30   31   16   29   30   32   48
 961  962  963  964  965  966  967  968  969  970  971  972  973  974  975  976
  27   27   24   30   26   35   43   30   51   49   40   41   36   32   36   38
 977  978  979  980  981  982  983  984  985  986  987  988  989  990  991  992
  43   41   42   37   49   44   51   57   55   40   53   56   63   39   58   50
 993  994  995  996  997  998  999 1000 1001 1002 1003 1004 1005 1006 1007 1008
  58   77   74   72   72   84   74   66   77   85   93   79   94  118  122  104
1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019
 121  113  120   78   77   55   37   20    7    5    1
```

Table 29. Sparse matrix customers products.
Source: Own elaboration.

Before we implement the Apriori algorithm to our problem, lets take a look at our newly created sparse matrix.[Table 29]

In numerical analysis and scientific computing, a sparse matrix or sparse array is a matrix in which most of the elements are zero. By contrast, if most of the elements are nonzero, then the matrix is considered dense. The number of zero-valued elements divided by the total number of elements (e.g., m × n for an m × n matrix) is called the sparsity of the matrix (which is equal to 1 minus the density of the matrix). Using those definitions, a matrix will be sparse when its sparsity is greater than 0.5.[13]

```
summary(customersProducts)
transactions as itemMatrix in sparse format with
 5892 rows (elements/itemsets/transactions) and
 10539 columns (items) and a density of 0.008768598

most frequent items:
P00265242 P00110742 P00025442 P00112142 P00057642   (Other)
     1858      1591      1586      1539      1430    536489

element (itemset/transaction) length distribution:
sizes
   6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21
   1    5    7   20   37   55   77   78  120  113  121  104  122  118   94   79
  22   23   24   25   26   27   28   29   30   31   32   33   34   35   36   37
  93   85   77   66   74   84   72   72   74   77   58   50   58   39   63   56
  38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53
  53   40   55   57   51   44   49   37   42   41   43   38   36   32   36   41
  54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69
  40   49   51   30   43   35   26   30   24   27   27   48   32   30   29   16
  70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85
  31   30   40   25   27   24   30   31   30   19   25   20   21   23   23   30
  86   87   88   89   90   91   92   93   94   95   96   97   98   99  100  101
  22   14   20   20   20   14   18   28   20   22   27   17   15   13   16   20
 102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117
  20   12   13   13   11   12   17   21   18   12   15   19    8   23   18   17
 118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133
  10   15   19    9   10   14    5   18   11    7    9   14    7   21    6   13
 134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149
  11   10   13   15   12    6    4    8   12   13   15    4    5    7    8    9
 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165
   9   11    8    7    6   12    5    6    7   10   11    6    9   11   11    9
 166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181
  12    4    7    8   11   16   12    7   10    6   10    6    4   11    9    7
```

[13] https://en.wikipedia.org/wiki/Sparse_matrix

```
 182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197 
   4    6    7    9    4    5    5   11    4    5    3    4    5    4    5    5 
 198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213 
   7    4    5    2    7    7    4    6    6   10    5   11    3    6    6    2 
 214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229 
   3    5    4    5    7    5    4    3    4    7    4    4    3    9    5    5 
 230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245 
   8    3    5    3    2    5    5    6    3    3    5    5    4    3    7    4 
 246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261 
   2    3    3    2    2    4    3    1    3    2    2    5    2    3    2    4 
 262  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278 
   3    5    1    2    6    2    6    6    4    3    2    3    5    5    1    1 
 280  281  282  283  284  285  286  287  288  289  290  291  292  293  295  296 
   6    4    7    6    1    4    3    6    4    2    6    5    1    2    3    3 
 297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312 
   1    2    5    2    1    1    2    5    3    3    2    1    1    2    2    5 
 313  315  316  317  318  319  320  321  322  323  325  326  327  328  330  331 
   4    2    2    3    2    2    1    1    3    1    2    1    1    1    1    2 
 333  334  335  336  337  338  339  340  342  343  344  346  347  348  349  351 
   1    1    1    1    2    2    5    4    4    2    2    3    3    2    1    1 
 353  354  355  356  357  358  359  360  361  362  363  364  366  367  368  371 
   3    1    2    4    2    1    2    1    1    3    1    2    1    1    2    3 
 372  377  378  379  380  381  382  383  384  385  387  390  391  392  393  400 
   1    1    4    1    3    2    1    2    1    2    2    2    5    1    6    1 
 402  405  406  407  408  409  410  411  412  413  415  417  418  421  423  424 
   1    2    3    1    2    1    1    2    1    2    1    1    2    1    1    1 
 427  428  430  431  432  433  434  435  436  437  439  441  442  445  447  448 
   1    2    3    1    1    1    1    1    5    3    2    1    1    2    3    2 
 450  451  453  455  458  459  462  467  469  470  471  472  476  477  479  480 
   1    4    2    3    1    3    1    2    1    1    2    1    1    3    1    1 
 485  486  487  488  490  492  493  494  495  497  498  499  501  502  518  527 
   1    3    2    1    1    1    1    2    1    1    2    1    2    1    1    2 
 530  534  538  544  548  549  550  558  559  560  566  569  571  573  575  576 
   3    2    2    1    1    1    1    2    2    1    1    1    1    1    1    1 
 584  588  606  617  623  632  652  668  671  677  680  681  685  691  695  698 
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    2 
 706  709  715  718  740  753  767  823  862  899  979 1025 1026 
   1    1    1    1    1    1    1    1    1    1    1    1    1 

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   6.00   26.00   54.00   92.41  115.00 1026.00 
```

Table 30. Transactions as item Matrix.
Source: Own elaboration.

Here, we can see that there are 5892 rows (elements/itemsets/transactions) and 10539 columns (items) in our sparse matrix. With this sumary function, we get a density of 0.008768598 in our matrix. The density tells us that we have 0.9% non-zero values (1) in our sparse matrix and 99.1% zero (0) values.

Also, as we discovered in our Exploratory Data Analysis, the summary() function also gives us the most frequent items that customers purchased and just to be sure, we can cross reference what we discovered earlier in the analysis. Lets list out what our sparse matrix gave us.[Table 30]

- P00265242 = 1858
- P00110742 = 1591
- P00025442 = 1586

- P00112142 = 1539
- P00057642 = 1430
- (Other) = 536489

Now we can compare it to what we discovered earlier.

"Looks like our top 5 best sellers are (by product ID)"

- P00265242 = 1858
- P00110742 = 1591
- P00025442 = 1586
- P00112142 = 1539
- P00057642 = 1430

Looks like our sparce matrix is accurate to what we discovered earlier. It is important to ensure that all data is being transfered correctly in every step of the analysis. This ensures repeatability and easy debugging should an error occur.

Continue to examine our sparse matrix.

```
summary(customersProducts)
transactions as itemMatrix in sparse format with
 5892 rows (elements/itemsets/transactions) and
 10539 columns (items) and a density of 0.008768598

most frequent items:
P00265242 P00110742 P00025442 P00112142 P00057642   (Other)
     1858      1591      1586      1539      1430    536489

element (itemset/transaction) length distribution:
sizes
   6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21
   1    5    7   20   37   55   77   78  120  113  121  104  122  118   94   79
  22   23   24   25   26   27   28   29   30   31   32   33   34   35   36   37
  93   85   77   66   74   84   72   72   74   77   58   50   58   39   63   56
  38   39   40   41   42   43   44   45   46   47   48   49   50   51   52   53
  53   40   55   57   51   44   49   37   42   41   43   38   36   32   36   41
  54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69
  40   49   51   30   43   35   26   30   24   27   27   48   32   30   29   16
  70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85
  31   30   40   25   27   24   30   31   30   19   25   20   21   23   23   30
  86   87   88   89   90   91   92   93   94   95   96   97   98   99  100  101
  22   14   20   20   20   14   18   28   20   22   27   17   15   13   16   20
 102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117
  20   12   13   13   11   12   17   21   18   12   15   19    8   23   18   17
 118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133
  10   15   19    9   10   14    5   18   11    7    9   14    7   21    6   13
 134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149
  11   10   13   15   12    6    4    8   12   13   15    4    5    7    8    9
 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165
   9   11    8    7    6   12    5    6    7   10   11    6    9   11   11    9
 166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181
  12    4    7    8   11   16   12    7   10    6   10    6    4   11    9    7
```

```
 182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197
   4    6    7    9    4    5    5   11    4    5    3    4    5    4    5    5
 198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213
   7    4    5    2    7    7    4    6    6   10    5   11    3    6    6    2
 214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229
   3    5    4    5    7    5    4    3    4    7    4    4    3    9    5    5
 230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245
   8    3    5    3    2    5    5    6    3    3    5    5    4    3    7    4
 246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261
   2    3    3    2    2    4    3    1    3    2    2    5    2    3    2    4
 262  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278
   3    5    1    2    6    2    6    6    4    3    2    3    5    5    1    1
 280  281  282  283  284  285  286  287  288  289  290  291  292  293  295  296
   6    4    7    6    1    4    3    6    4    2    6    5    1    2    3    3
 297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312
   1    2    5    2    1    1    2    5    3    3    2    1    1    2    2    5
 313  315  316  317  318  319  320  321  322  323  325  326  327  328  330  331
   4    2    2    3    2    2    1    1    3    1    2    1    1    1    1    2
 333  334  335  336  337  338  339  340  342  343  344  346  347  348  349  351
   1    1    1    1    2    2    5    4    4    2    2    3    3    2    1    1
 353  354  355  356  357  358  359  360  361  362  363  364  366  367  368  371
   3    1    2    4    2    1    2    1    1    3    1    2    1    1    2    3
 372  377  378  379  380  381  382  383  384  385  387  390  391  392  393  400
   1    1    4    1    3    2    1    2    1    2    2    2    5    1    6    1
 402  405  406  407  408  409  410  411  412  413  415  417  418  421  423  424
   1    2    3    1    2    1    1    2    1    2    1    1    2    1    1    1
 427  428  430  431  432  433  434  435  436  437  439  441  442  445  447  448
   1    2    3    1    1    1    1    1    5    3    2    1    1    2    3    2
 450  451  453  455  458  459  462  467  469  470  471  472  476  477  479  480
   1    4    2    3    1    3    1    2    1    1    2    1    1    3    1    1
 485  486  487  488  490  492  493  494  495  497  498  499  501  502  518  527
   1    3    2    1    1    1    1    2    1    1    2    1    2    1    1    2
 530  534  538  544  548  549  550  558  559  560  566  569  571  573  575  576
   3    2    2    1    1    1    1    2    2    1    1    1    1    1    1    1
 584  588  606  617  623  632  652  668  671  677  680  681  685  691  695  698
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    2
 706  709  715  718  740  753  767  823  862  899  979 1025 1026
   1    1    1    1    1    1    1    1    1    1    1    1    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   6.00   26.00   54.00   92.41  115.00 1026.00 
```

Table 31. Transactions as item Matrix.
Source: Own elaboration.

The "element (itemset/transaction) length distribution" gives us a distribution of the number of items in a customers (User) basket and underneath it we can see more information including the quartile and mean information. In this case, we see a mean of 92.41, which means that on average, each customer purchased 92.41 items. In this case, since we are aware of a few customers who purchased over ~1000 items, it may be useful to use the median value of 54.00 items instead since the mean can be heavily affected by outlier values.

To get a clearer picture of the items, lets create an item frequency plot which is included in the arules package.[Table 31]

```
itemFrequencyPlot(customersProducts, topN = 25)     # topN is limiting to the top 50
products
```

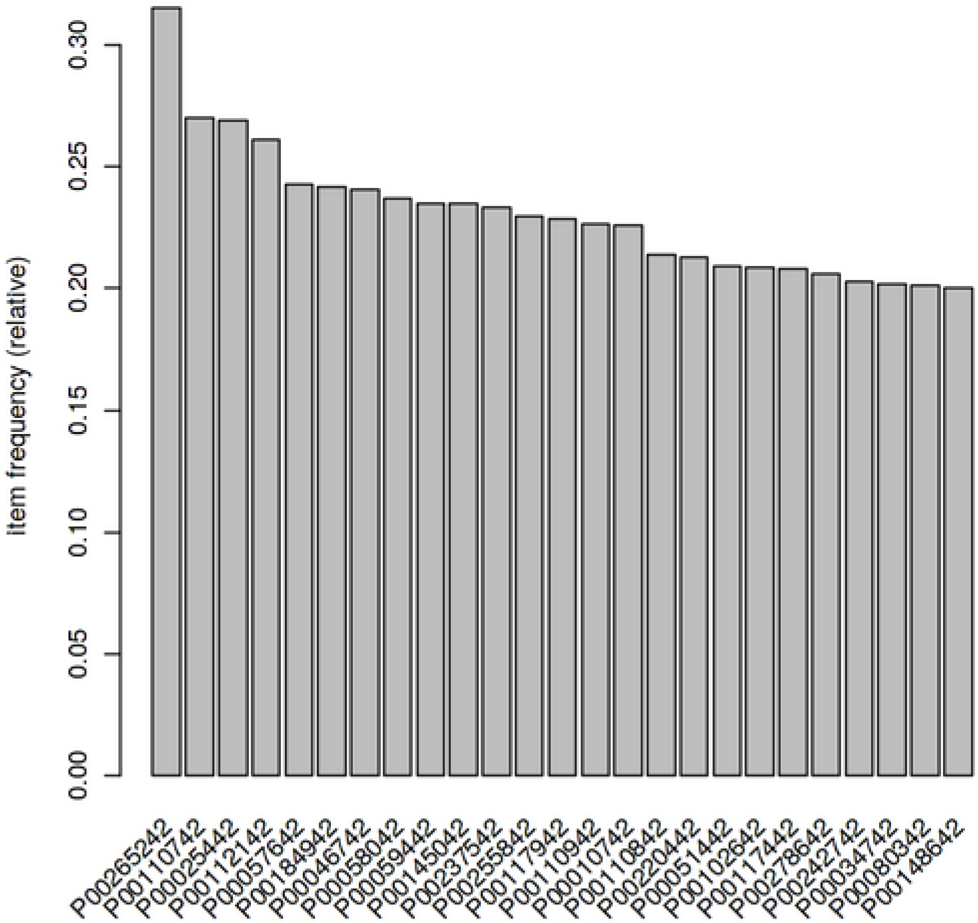


Figure 14. Frequency Plot customers Products
Source: Own elaboration.

Now we begin training the association rule model.

Our first step will be to set our parameters. The first parameters we will set are the support and confidence. The support value is derived from the frequency of a specific item within the dataset. When we set our support value, we are setting a minimum number of transactions necessary for our rules to take effect.

- Support: Our support value wil be the minimum number of transactions necessary divided by the total number of transactions.

- As described by summary(customersProducts), we have a total number of unique customer transactions of 5892.
- From our dataset, lets assume that we want to choose a product which was purchased by at least 50 different customers.
- With these two values established, we can compute the support value with simple division. (50/5892) = .008486083

The second parameter we will take into consideration will be the confidence. The confidence value determines how often a rule is to be found true. In other words, the minimum strength of any rule is a limit we place when setting our minimum confidence value.

The default confidence value in the apriori() function is 0.80 or 80%, so we can begin with that number and then adjust the parameters to applicable results.

- Confidence: We can determine our confidence value by first starting with the default value and adjusting accordingly.

    - With more domain knowledge, and with Product_IDs referencing items with recognizable names, the Confidence value can be easily changed to see different, and more relevant, results.
    - In our case, we will start with a value and then lower the confidence to see different rules.

```
rules = apriori(data = customersProducts,
               parameter = list(support = 0.008, confidence = 0.80, maxtime = 0)) #
maxtime = 0 will allow our algorithim to run until completion with no time limit
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       0   0.008      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 47

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10539 item(s), 5892 transaction(s)] done [0.10s].
sorting and recoding items ... [2099 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [19.52s].
writing ... [7 rule(s)] done [0.47s].
creating S4 object  ... done [0.24s].
```

It looks like apriori has created 7 rules in accordance to our specified parameters.

```
"writing ... [7 rule(s)] done [0.48s]."
```

Now we can examine our results to get a better idea of how our algoritm worked.

```
inspect(sort(rules, by = 'lift'))
lhs            rhs            support confidence     lift count
[1] {P00032042,
P00057642,
P00102642,
P00145042} => {P00270942} 0.008655804  0.8793103 4.540663    51
[2] {P00025442,
P00031042,
P00034742,
P00255842} => {P00145042} 0.008486083  0.8064516 3.433246    50
[3] {P00003242,
P00130742,
P00237542} => {P00145042} 0.008316361  0.8032787 3.419738    49
[4] {P00006942,
P00251242,
P00277642} => {P00145042} 0.009674134  0.8028169 3.417773    57
[5] {P00034042,
P00112442,
P00112542} => {P00110742} 0.008146640  0.8135593 3.012880    48
[6] {P00127642,
P00165442,
P00277442} => {P00110742} 0.008316361  0.8032787 2.974807    49
[7] {P00051442,
P00112142,
P00112542,
P00270942} => {P00110742} 0.008146640  0.8000000 2.962665    48
```

Table 32. Transactions apriori for data customersProducts
Source: Own elaboration.

We present the association rules created by our apriori algorithm. Let's take a look at rule number 1.

We see a few values listed and we will go through them individually.

- The first value lhs, corresponds to a grouping of items which the algorithm has pulled from the dataset.
- The second value, rhs, corresponds to the value predicted by apriori to be purchased with items in the "lhs" category.
- The third value, support is the number of transactions including that specific set of items divided by the total number of transactions. (As described earlier when we chose the parameters for Apriori.)
- The fourth value, confidence is the % chance in which a rule will be upheld.
- The fifth value, lift gives us the independance/dependence of a rule. It takes the confidence value and its relationship to the entire dataset into account.
- The sixth and final value, count is the number of times a rule occured during the implementation of Apriori on our data.

Now, lets visualize these rules using the arulesViz package.

```
plot(rules, method = 'graph')
```

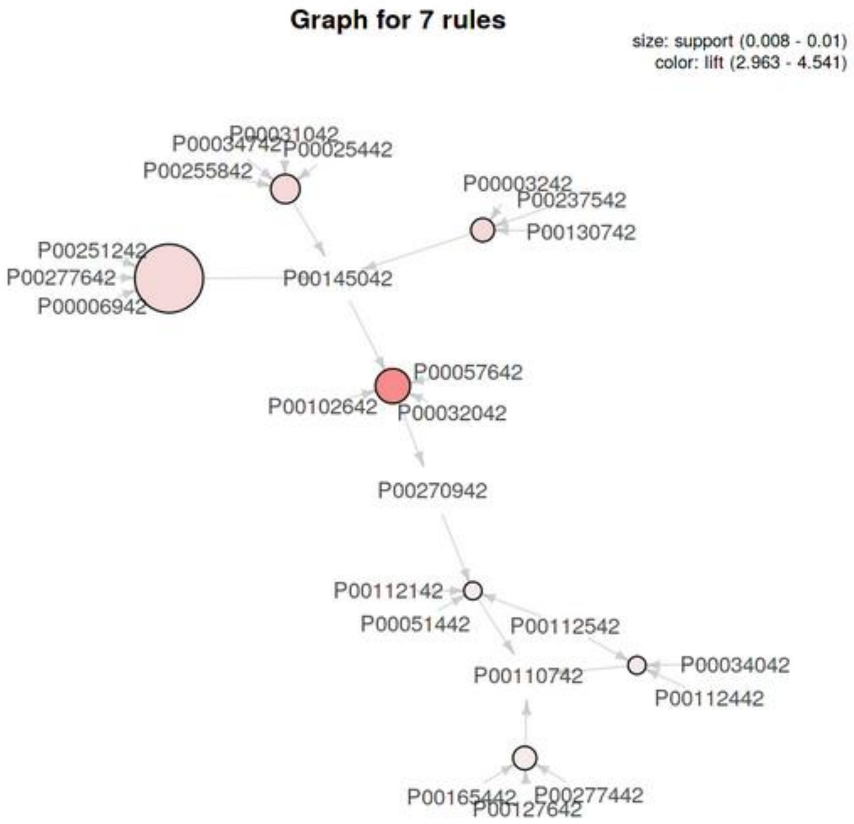


Figure 15. Graph present 7 rules.
Source: Own elaboration.

In a visualization of our association rules. Arrows pointing from items to rule vertices indicate LHS (Grouped) items and arrows from rules to items indicates the RHS (Rule Item).

The size of the bubbles indicate the support with larger bubbles representing a higher support value. Fill color represents the lift values, with darker colors representing higher lifts.

Lets now try modifying some of the parameters for the Apriori algotrithm and see the results. This process would prove to be more intuitive if given a key for each corresponding Product_ID, so will only implement the algorithm once more.

This time, we will decrease our confidence value to 75% and keep our support value the same (0.008).

```
ules = apriori(data = customersProducts,
               parameter = list(support = 0.008, confidence = 0.75, maxtime = 0))
Apriori
```

```
Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
       0.75    0.1    1 none FALSE            TRUE       0   0.008      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 47

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10539 item(s), 5892 transaction(s)] done [0.10s].
sorting and recoding items ... [2099 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [19.37s].
writing ... [171 rule(s)] done [0.46s].
creating S4 object  ... done [0.25s].
```

Now that we have decreased the minimum confidence value to 75%, we have a total of 171 rules.

```
writing ... [171 rule(s)] done [0.50s].
```

This is much higher number of rules compared to our previous rule list which only contained 7. This should now give us more interesting rules to examine.

```
inspect(head(sort(rules, by = 'lift'))) # limiting to the top 6 rules
lhs            rhs             support confidence     lift count
[1] {P00221142,
P00249642} => {P00103042} 0.008146640  0.7619048 8.030667    48
[2] {P00002142,
P00103042,
P00147942} => {P00221442} 0.008146640  0.7500000 6.045144    48
[3] {P00032042,
P00057642,
P00102642,
P00145042} => {P00270942} 0.008655804  0.8793103 4.540663    51
[4] {P00062842,
P00127242,
P00243942} => {P00044442} 0.008486083  0.7575758 4.061544    50
[5] {P00030842,
P00057942,
P00355142} => {P00114942} 0.008486083  0.7936508 4.024260    50
[6] {P00030842,
P00147742,
P00303342} => {P00044442} 0.008146640  0.7500000 4.020928    48
```

Table 33. Limiting to the top 6 rules●
Source: Own elaboration.

We can now see that we now have a new set of rules and the rule with the highest lift value has also changed.

Rule number 1 shows that Customers who bought items P00221142 and P00249642 will also purchase item P00103042 ~76% of the time, given a support of 0.008.

```
plot(rules, method = 'graph', max = 25)
Warning message:
"plot: Too many rules supplied. Only plotting the best 25 rules using
'support' (change control parameter max if needed)"
```

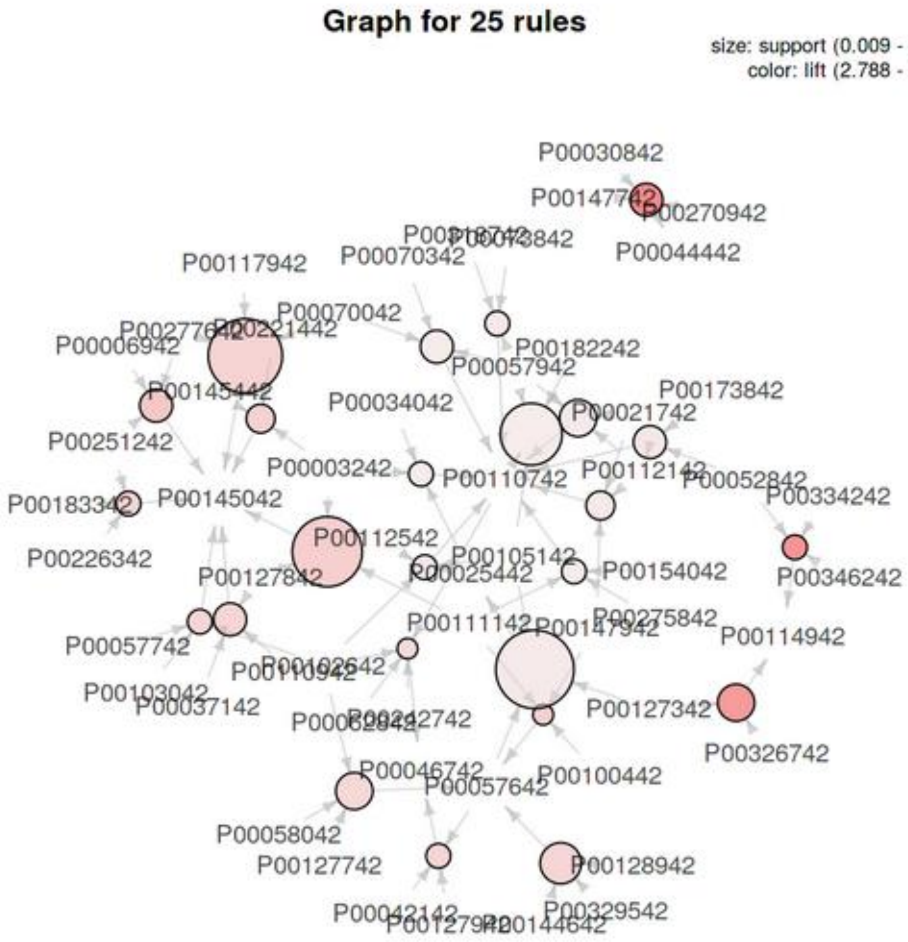


Figure 16. Graph present the 25 rules
Source: Own elaboration.

Now that we have more that 7 rules, this visualization becomes alot more difficult to interpret.

Instead, we can create a matrix and have a similar plot and clearer interpretation.

```
plot(rules, method = 'grouped', max = 25)
```

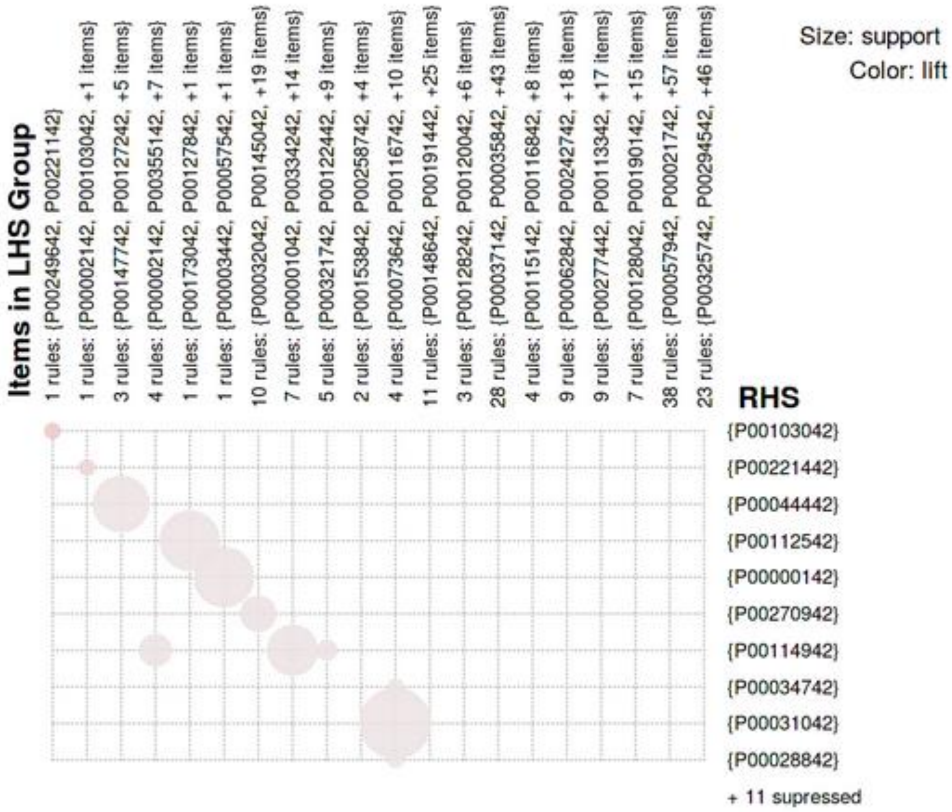


Figure 17. Graph LHS.
Source: Own elaboration.

In this visualization, we can see that we have our LHS on top and on the right hand side, the corresponding RHS. The size of the bubbles represents the support value of the rule and the fill / color represents the lift.

5. Conclussions

1. After the analisys of this data, we can confirm the male customers have a higher average spending then the female.

2. We can the gender of the customer are we rejection becouse the most of customers are men.

3. The category of age of customers we can see is 26-35. From this data and analyses now we can confirm the hypotheses.

References

1. Chang, E., Burns, L. D., & Francis, S. K. (2004) (Abstract)

2. Oxford Dictionaries- Black Friday

3. https://en.wikipedia.org/wiki/Black_Friday_(shopping)

4. https://www.kaggle.com/mehdidag/black-friday

5. https://en.wikipedia.org/wiki/Ggplot2

6. https://www.history.com/news/whats-the-real-history-of-black-friday

7. https://en.oxforddictionaries.com/explore/why-is-day-after-thanksgiving-black-friday/

8. https://www.cnn.com/2018/11/21/business/black-friday-history/index.html

9.https://journals.sagepub.com/doi/abs/10.1177/0887302X0402200404#articleCitationDownloadContainer

10. https://en.wikipedia.org/wiki/Apriori_algorithm

11. https://en.wikipedia.org/wiki/Association_rule_learning

Packages

1. https://www.tidyverse.org/

2. https://cran.r-project.org/web/packages/scales/scales.pdf

3. https://cran.r-project.org/web/packages/arules/arules.pdf

4. https://cran.r-project.org/web/packa[1] https://plot.ly/r/

5. https://www.numpy.org/

6. https://pandas.pydata.org/

7. https://matplotlib.org/

8. ges/arulesViz/vignettes/arulesViz.pdf

Images

Tables

Figures