Code ▾

Exploring Sales Trends, Product Performance, and Customer Segmentation in Online Retail

#Exploring Sales Trends, Product Performance, and Customer Segmentation in Online Retail

##Premise:

Despite the growing popularity of online retail, many businesses struggle to fully understand their customer behavior and preferences

In this project we aim to explore the retail dataset in order to:

1* Perform statistical analysis to identify key patterns and trends in customer purchasing behavior 2* Perform clustering of customers in terms of product purchases, geographical location etc. 3* Try to identify anomalous customer behavior

Thus enabling businesses to make data-driven decisions that can improve customer satisfaction and drive profitability

Dataset Info: This Online Retail II dataset (https://archive.ics.uci.edu/ml/datasets/Online+Retail+II) contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011.The company mainly sells unique all-occasion gift-ware. Many customers of the company are wholesalers. (source: UCI ML Repository)

##**PART 1: DATA PREPROCESSING** In this part, we clean our data and remove any missing values or outliers that may affect our results. We will also transform our data if necessary, such as taking logarithmic transformations to normalize the data. This will be followed by feature engineering and then the data is ready to be used for machine learning algorithms.

Let us install the required libraries first:

Hide

```
#install.packages("readr") # uncomment and run to install readr package
library(readr) # load readr package

#install.packages("ggplot2") # uncomment and run to install ggplot2 package
library(ggplot2) # load ggplot2 package

#install.packages("dplyr") # uncomment and run to install dplyr package
library(dplyr) # load dplyr package
```

```

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
#install.packages("moments") # uncomment and run to install moments package
library(moments) # load moments package

# Install readxl package
#install.packages("readxl")
library(readxl)

#install.packages("tidyr")
library(tidyr)
```

Hide

```
#reading the data
online_retail<- read_xlsx("C:\\Users\\hp\\Downloads\\online_retail.xlsx")
online_retail <- distinct(online_retail)
online_retail
```

Invoice <chr>	StockCo… <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45
489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45
489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45
489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45
489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45
489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45
489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45
489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45
489435	22350	CAT BOWL	12	2009-12-01 07:46
489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46

1-10 of 518,596 rows | 1-6 of 8 columns Previous **1** 2 3 4 5 6 ... 100 Next

Cool! Let’s find out if there are any missing values and duplicates:

Hide

```
# Check for missing values
online_retail %>%summarize_all(~sum(is.na(.)))
```

Invoice <int>	StockCode <int>	Description <int>	Quantity <int>	InvoiceDate <int>	Price <int>	Customer ID <int>	Country <int>
0	0	2928	0	0	0	107833	0

1 row

There seem to be some missing values. Let's remove them.

Hide

```
# Remove missing values
or_cleaned <- online_retail %>% drop_na()
or_cleaned %>%summarize_all(~sum(is.na(.)))
```

Invoice <int>	StockCode <int>	Description <int>	Quantity <int>	InvoiceDate <int>	Price <int>	Customer ID <int>	Country <int>
0	0	0	0	0	0	0	0

1 row

Hide

```
or_cleaned
```

Invoice <chr>	StockCo... <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45
489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45
489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45
489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45
489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45
489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45
489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45
489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45
489435	22350	CAT BOWL	12	2009-12-01 07:46
489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46

1-10 of 410,763 rows | 1-6 of 8 columns Previous **1** 2 3 4 5 6 ... 100 Next

Hide

```
str(or_cleaned)
```

```
tibble [410,763 × 8] (S3: tbl_df/tbl/data.frame)
 $ Invoice    : chr [1:410763] "489434" "489434" "489434" "489434" ...
 $ StockCode  : chr [1:410763] "85048" "79323P" "79323W" "22041" ...
 $ Description: chr [1:410763] "15CM CHRISTMAS GLASS BALL 20 LIGHTS" "PINK CHERRY LIGHTS" "WH
ITE CHERRY LIGHTS" "RECORD FRAME 7\" SINGLE SIZE" ...
 $ Quantity   : num [1:410763] 12 12 12 48 24 24 24 10 12 12 ...
 $ InvoiceDate: POSIXct[1:410763], format: "2009-12-01 07:45:00" "2009-12-01 07:45:00" "2009-
12-01 07:45:00" ...
 $ Price      : num [1:410763] 6.95 6.75 6.75 2.1 1.25 1.65 1.25 5.95 2.55 3.75 ...
 $ Customer ID: num [1:410763] 13085 13085 13085 13085 13085 ...
 $ Country    : chr [1:410763] "United Kingdom" "United Kingdom" "United Kingdom" "United Kin
gdom" ...
```

Based on the output of the function, we can see that the dataset is a tibble with 410,763 rows and 8 columns, which are:

1* Invoice: **character vector** representing the invoice number for each transaction. 2* StockCode: **character vector** representing the stock code for each item purchased. 3* Description: **character vector**r describing the item purchased. 4* Quantity: **numeric vector** representing the quantity of each item purchased. 5* InvoiceDate: **POSIXct object** representing the date and time of each transaction. 6* Price: **numeric vector** representing the price of each item. 7* Customer ID: **numeric vector** representing the ID of the customer who made each transaction. 8* Country: **character vector** representing the country where the transaction took place.

It is important to convert attributes like “invoice” and “InvoiceDate” into numeric and date time types. Converting the “Invoice” attribute to numeric is not necessary since it is an identifier and not a numerical value. However, converting “InvoiceDate” to date time type is important for time series analysis and other date-related operations.

Hide

```
or_cleaned$InvoiceDate <- as.POSIXct(or_cleaned$InvoiceDate, format = "%Y-%m-%d %H:%M:%S")
```

Hide

```
str(or_cleaned)
```

```
tibble [410,763 × 8] (S3: tbl_df/tbl/data.frame)
 $ Invoice    : chr [1:410763] "489434" "489434" "489434" "489434" ...
 $ StockCode  : chr [1:410763] "85048" "79323P" "79323W" "22041" ...
 $ Description: chr [1:410763] "15CM CHRISTMAS GLASS BALL 20 LIGHTS" "PINK CHERRY LIGHTS" "WH
ITE CHERRY LIGHTS" "RECORD FRAME 7\" SINGLE SIZE" ...
 $ Quantity   : num [1:410763] 12 12 12 48 24 24 24 10 12 12 ...
 $ InvoiceDate: POSIXct[1:410763], format: "2009-12-01 07:45:00" "2009-12-01 07:45:00" "2009-
12-01 07:45:00" ...
 $ Price      : num [1:410763] 6.95 6.75 6.75 2.1 1.25 1.65 1.25 5.95 2.55 3.75 ...
 $ Customer ID: num [1:410763] 13085 13085 13085 13085 13085 ...
 $ Country    : chr [1:410763] "United Kingdom" "United Kingdom" "United Kingdom" "United Kin
gdom" ...
```

Hide

```
# Function to identify outliers using Tukey's method
tukey_outliers <- function(x) {
  qnt <- quantile(x, probs = c(.25, .75), na.rm = TRUE)
  caps <- quantile(x, probs = c(.01, .99), na.rm = TRUE)
  fence <- IQR(x, na.rm = TRUE) * 1.5
  lw <- qnt[1] - fence
  uw <- qnt[2] + fence
  x[x < lw] <- NA
  x[x > uw] <- NA
  return(x)
}

# Identify Tukey outliers in each numeric column
or_cleaned_outliers <- or_cleaned %>%
  mutate(across(where(is.numeric), tukey_outliers))

# Summarize the number of outliers in each column
outlier_counts <- or_cleaned_outliers %>%
  summarise(across(where(is.numeric), ~ sum(is.na(.))))
outlier_counts
```

Quantity <int>	Price <int>	Customer ID <int>
27342	34703	0

1 row

Hide

```
# Remove Tukey outliers from each numeric column
or_cleaned_no_outliers <- or_cleaned_outliers %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))
or_cleaned_no_outliers
```

Invoice <chr>	StockCo… <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45
489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45
489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45
489434	22041	RECORD FRAME 7" SINGLE SIZE	4	2009-12-01 07:45
489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45
489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45
489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45
489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45
489435	22350	CAT BOWL	12	2009-12-01 07:46

Invoice <chr>	StockCo… <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46

1-10 of 410,763 rows | 1-6 of 8 columns Previous **1** 2 3 4 5 6 ... 100 Next

After identifying and handling missing values, duplicates, and outliers, the next step in data preprocessing is to perform feature scaling or normalization. This is done to bring all the features into the same range and to prevent certain features from dominating the others. It helps in improving the performance of machine learning algorithms and also reduces the computational time.

For doing this, we have to check the distribution of each attribute in our dataset. To check the distribution of each attribute in our dataset, we can create a histogram plot and/or a density plot.

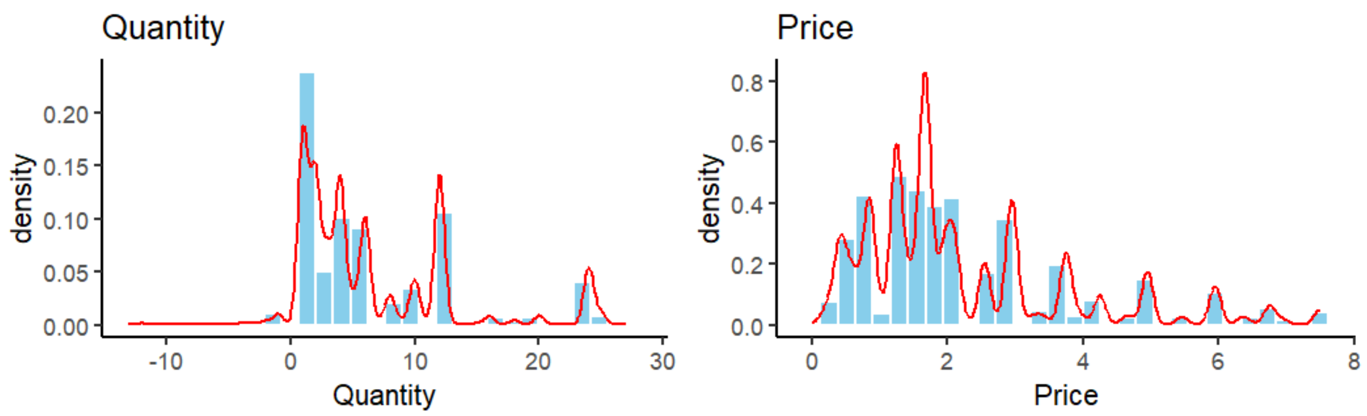
Hide

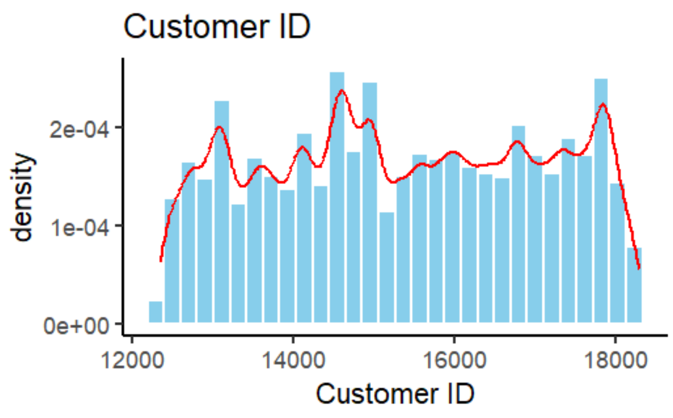
```
#install.packages("gridExtra")
library(gridExtra)
```

```


Attaching package: ‘gridExtra’

The following object is masked from ‘package:dplyr’:

    combine
```

Hide

```
# Identify numeric columns
numeric_cols <- or_cleaned_no_outliers %>% select_if(is.numeric)

# Create a list of histogram plots
hist_plots <- lapply(names(numeric_cols), function(x) {
  ggplot(or_cleaned_no_outliers, aes(x = !!sym(x))) +
    geom_histogram(aes(y = ..density..), fill = "skyblue", color = "white") +
    geom_density(alpha = 0.5, color = "red") +
    labs(title = x) +
    theme_classic()
})

# Arrange and print the plots
grid.arrange(grobs = hist_plots, ncol = 2)
```





Hide

```
format(100000, big.mark=",")
```

```
[1] "1e+05"
```

There are several types of scaling methods, and the choice of scaling method depends on the type of data and the requirements of the machine learning algorithm. Here are some commonly used scaling methods:

1* Min-Max Scaling: This method scales the data to a fixed range, typically between 0 and 1. It is useful when the data has a **uniform distribution**.

2* Standardization: This method scales the data to have zero mean and unit variance. It is useful when the data has a **normal distribution**.

3* Logarithmic Scaling: This method scales the data logarithmically. It is useful when the data has a **skewed distribution**.

We can also choose to scale only certain attributes while leaving others unscaled if they do not require it. In our case, Customer ID is not really in need of scaling. The other arrributes seem to be skewed so let us check their ranges and then choose to perform Logarithmic scaling on them both.

Hide

```
#checking range of Quantity column
range(or_cleaned_no_outliers$Quantity)
```

```
[1] -13  27
```

Hide

```
#checking range of Price column
range(or_cleaned_no_outliers$Price)
```

```
[1] 0.0 7.5
```

It is not possible for a quantity attribute to have negative values as it represents a physical quantity. So instead of scaling, we need to eliminate the customers who have a purchase quantity < 1.

For the price to be starting from 0 is not uncommon, as prices can often start from 0, especially in cases where a product or service is being offered for free.

Hide

```
#eliminating customers with purchase quantity < 1
or_cleaned_no_outliers_filtered <- or_cleaned_no_outliers %>%
  filter(Quantity >= 1)

#checking for missing values
or_cleaned_no_outliers %>%summarize_all(~sum(is.na(.)))
```

Invoice <int>	StockCode <int>	Description <int>	Quantity <int>	InvoiceDate <int>	Price <int>	Customer ID <int>	Country <int>
0	0	0	0	0	0	0	0

1 row

All in all, we did not need scaling at all.

After performing feature scaling or normalization, the next step in data preprocessing is usually feature engineering. Feature engineering involves creating new features or transforming existing ones to make them more informative for machine learning algorithms.

It is important to evaluate the need for feature engineering on a case-by-case basis. Feature engineering is not always necessary, but it can significantly improve the performance of machine learning models. In some cases, the raw data itself may be sufficient for building accurate machine learning models.

In our case, it is not really necessary to perform feature engineering.

##**PART 2: APPLICATIONS**

A recap of our goals:

A* Perform statistical analysis to identify key patterns and trends in customer purchasing behavior B* Perform clustering of customers in terms of product purchases, geographical location etc. C* Try to identify anomalous customer behavior

PART 2A. Statistical analysis to identify key patterns and trends in customer purchasing behavior

Let us get some basic central and dispersion tendencies first.

Hide

```
# Modify the Quantity column
or_cleaned_no_outliers_filtered$Quantity <- floor(or_cleaned_no_outliers_filtered$Quantity)
```

Hide

```
library(dplyr)

# Filter out non-numeric columns
numeric_cols <- or_cleaned_no_outliers_filtered %>% select_if(is.numeric)

# Define function to get summary statistics
get_tendencies <- function(x) {
  c(min = min(x, na.rm = TRUE),
    max = max(x, na.rm = TRUE),
    mean = mean(x, na.rm = TRUE),
    median = median(x, na.rm = TRUE),
    mode = names(sort(table(x), decreasing = TRUE))[1],
    range = max(x, na.rm = TRUE) - min(x, na.rm = TRUE),
    variance = var(x, na.rm = TRUE),
    sd = sd(x, na.rm = TRUE))
}

# Apply the function to all numeric columns
tendencies <- numeric_cols %>%
  summarise_all(get_tendencies)
```

```
Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in dplyr
1.1.0.
Please use `reframe()` instead.
When switching from `summarise()` to `reframe()`, remember that `reframe()` always returns an
ungrouped data frame and adjust accordingly.
```

Hide

```
# Rename the columns
tendencies <- tendencies %>%
  rename_with(~ paste0(., "_stats"), everything())

# Convert the result into a dataframe
tendencies<-as.data.frame(tendencies)

# add a new column named "Stat_Name"
tendencies <- tendencies %>%
  mutate(Stat_Name = c("min", "max", "mean", "median", "mode", "range", "variance", "sd"))

# reorder the columns so that "Stat_Name" appears at the beginning
tendencies <- tendencies %>%
  select(Stat_Name, everything())
tendencies
```

Stat_Name <chr>	Quantity_stats <chr>	Price_stats <chr>	Customer ID_stats <chr>
min	1	0	12346
max	27	7.5	18287

Stat_Name <chr>	Quantity_stats <chr>	Price_stats <chr>	Customer ID_stats <chr>
mean	6.62928210192715	2.23486997037262	15360.5793881983
median	4	1.69	15311
mode	1	1.25	14911
range	26	7.5	5941
variance	40.1906241778985	2.53421530742458	2824965.67844807
sd	6.33960757286274	1.59192189111922	1680.76342132022

8 rows

What can we infer from this?

5941 customers shop at this site.

The average price of an item in this online site seems to be around 2.23 USD which seems pretty cheap!

The maximum amount of items that were sold in one order on this site were 27.

Most of the customers on the site purchase 1 item in a transaction.

The most frequently sold items correspond to the price 1.25 USD.

The most frequent customer of this site has a Customer ID of 14911.

TREND AND PATTERN EXTRACTION

Now that we have a clean dataset, let’s perform exploratory data analysis to identify patterns and trends in the data. Here are some examples of analyses we could perform:

1. Identify the top-selling products by quantity and revenue.
2. Determine which customers make the most purchases and spend the most money.
3. Analyze purchasing behavior by day of the week, month, or year.
4. Explore relationship between the number of products purchased and the total amount spent.

1) Identify the top-selling products by quantity and revenue.

To identify the top-selling products by quantity and revenue, we can use the group_by and summarise functions from the dplyr package. We will group the data by StockCode and calculate the total quantity and revenue for each product. Then we will arrange the results in descending order by quantity and revenue to find the top-selling products.

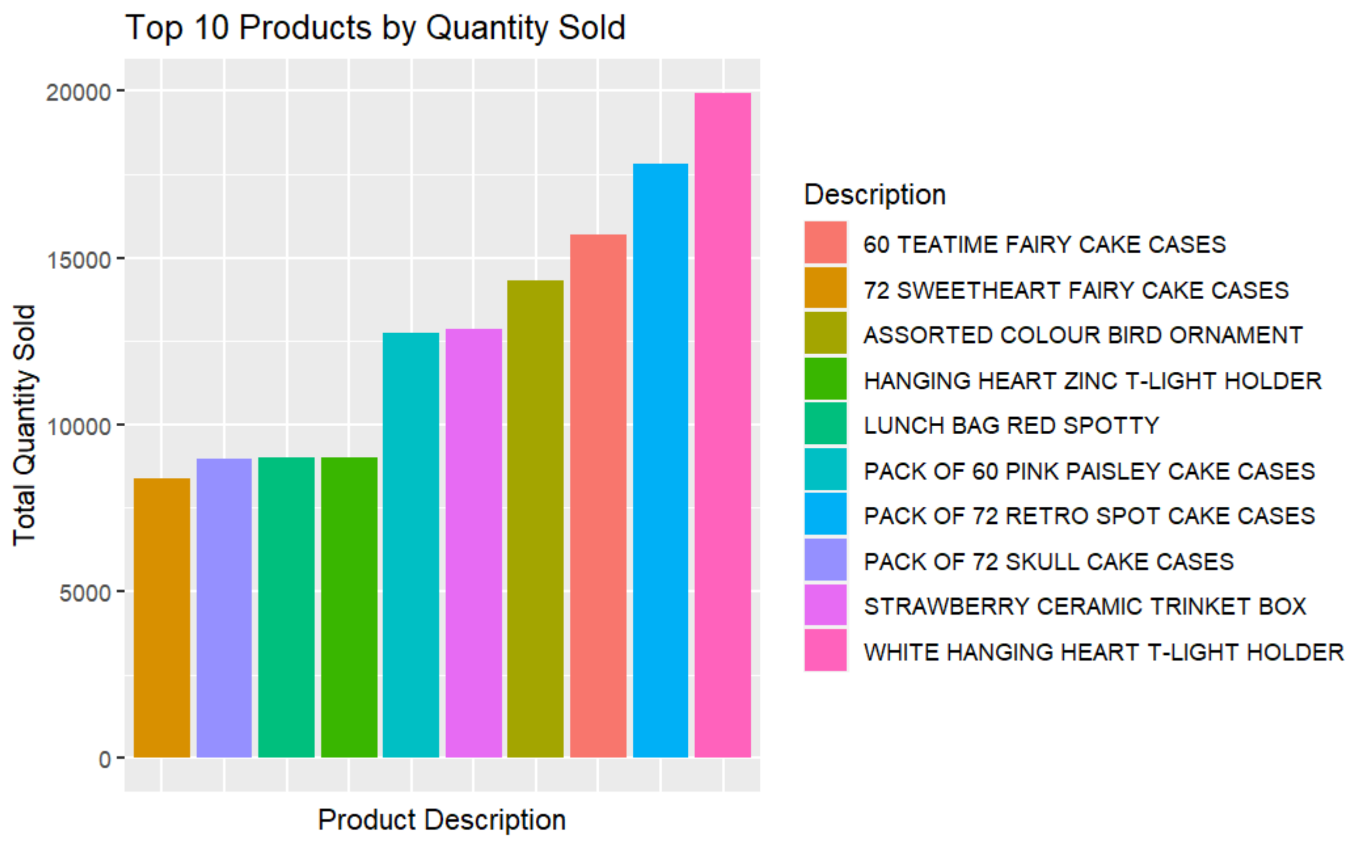
Hide

```
library(dplyr)
library(ggplot2)

# Group by StockCode and calculate total quantity and revenue
sales_by_product <- or_cleaned_no_outliers_filtered %>%
  group_by(StockCode, Description) %>%
  summarise(total_quantity = sum(Quantity),
            total_revenue = sum(Quantity * Price))
```

```
`summarise()` has grouped output by 'StockCode'. You can override using the `.groups` argumen
t.
```

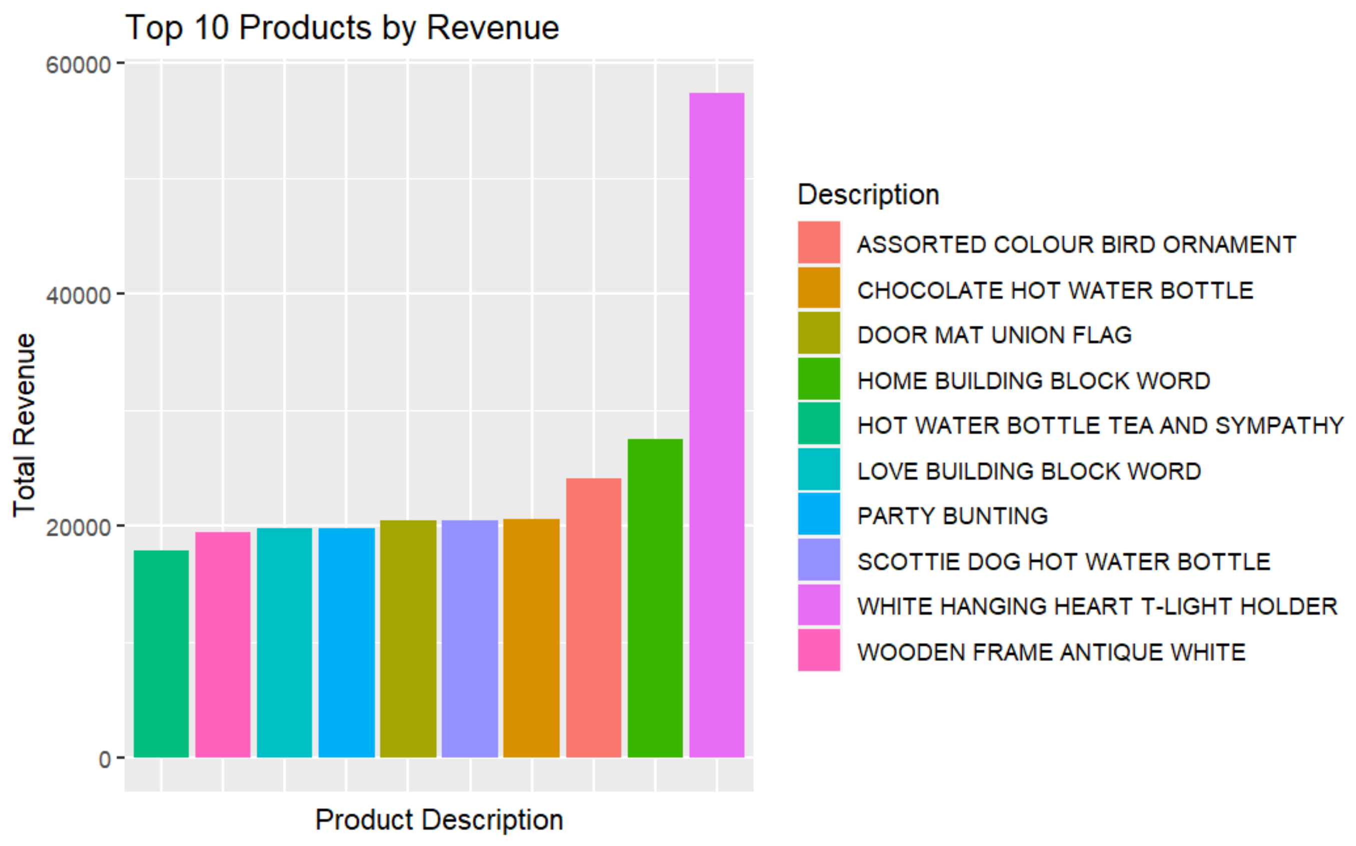
Hide

```
# Arrange by descending order of quantity and revenue to get top-selling products
top_quantity <- sales_by_product %>%
  arrange(desc(total_quantity)) %>%
  head(10)

top_revenue <- sales_by_product %>%
  arrange(desc(total_revenue)) %>%
  head(10)

# Visualize the results
ggplot(top_quantity, aes(x = reorder(Description, total_quantity), y = total_quantity, fill =
Description)) +
  geom_bar(stat = "identity") +
  xlab("Product Description") +
  ylab("Total Quantity Sold") +
  ggtitle("Top 10 Products by Quantity Sold") +  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



Hide

```
ggplot(top_revenue, aes(x = reorder(Description, total_revenue), y = total_revenue, fill = De
scription)) +
  geom_bar(stat = "identity") +
  xlab("Product Description") +
  ylab("Total Revenue") +
  ggtitle("Top 10 Products by Revenue") + theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
```



The product that was sold the most in the entirety of time and generated the highest amount of sales revenue in the dataset's given time period of one year is "White Hanging Heart T-Light Holder". About 20, 000 of them were sold.

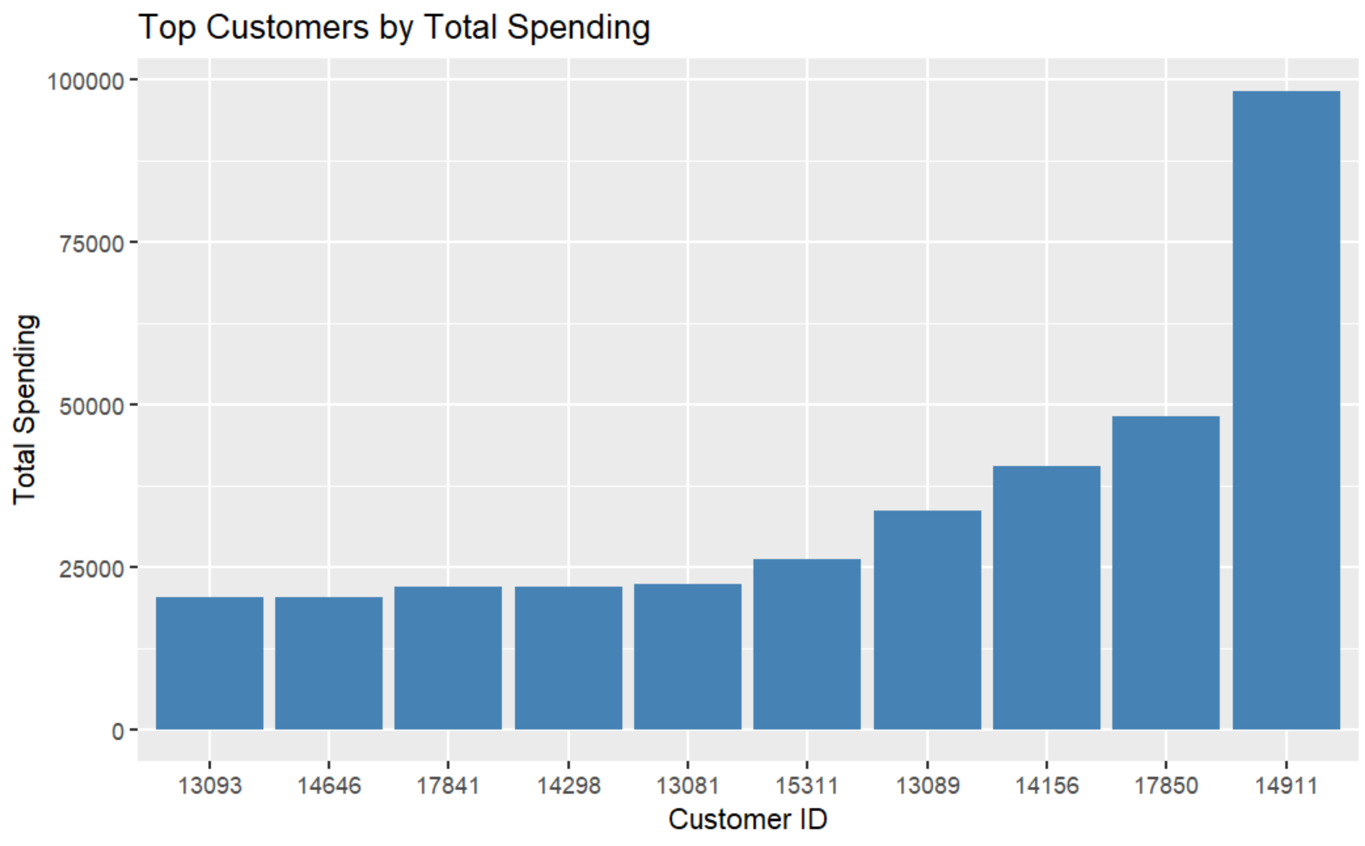
2) Determine which customers make the most purchases and spend the most money.

To determine which customers make the most purchases and spend the most money in the online retail dataset, we can perform the following steps:

Aggregate the data by customer ID to get the total number of purchases and total spending for each customer. Sort the data by total spending and total number of purchases to identify the customers who make the most purchases and spend the most money. Create visualizations to better understand the distribution of customer spending and purchasing behavior.

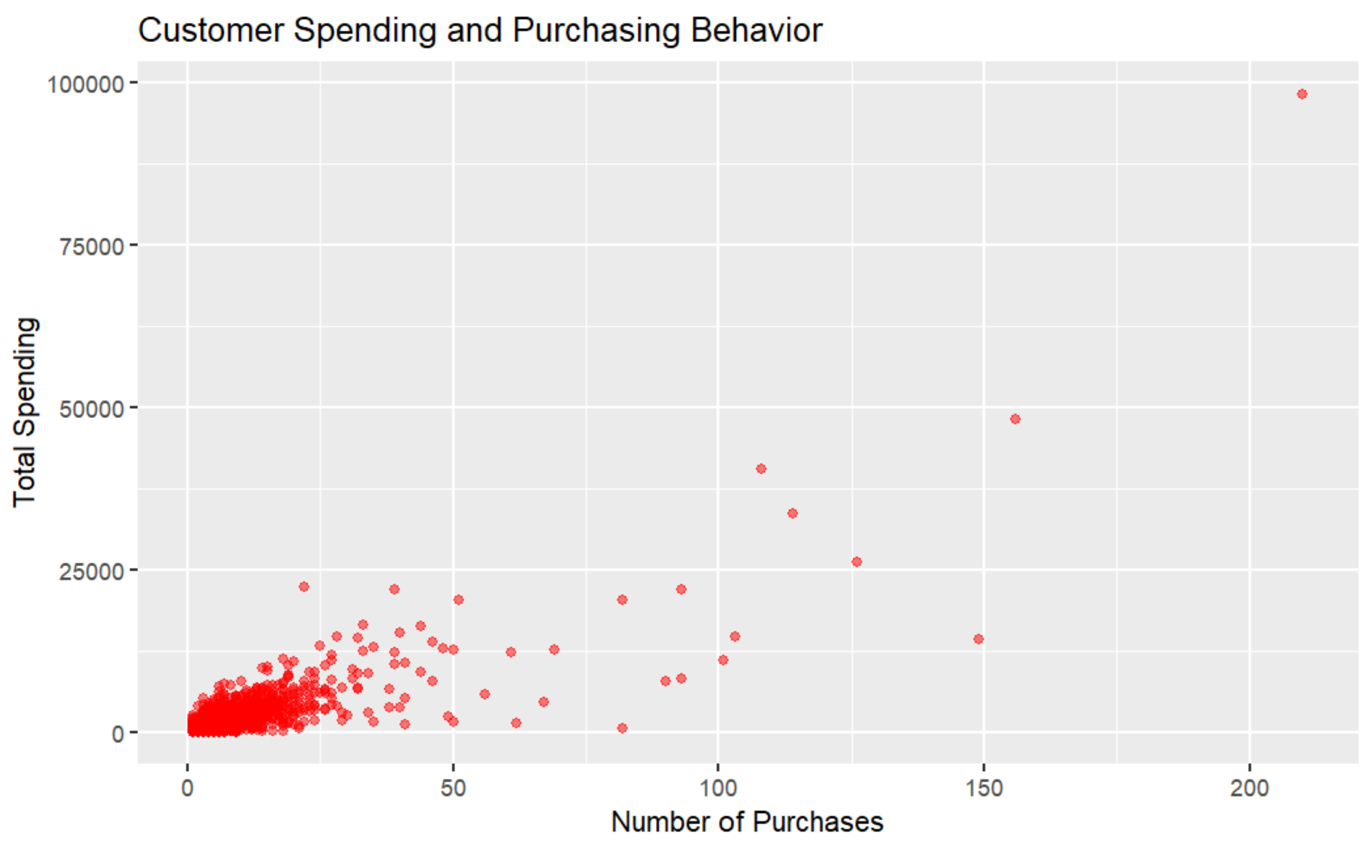
Hide

```
library(dplyr)
library(ggplot2)

# Aggregate the data by customer ID to get the total number of purchases and total spending f
or each customer
customer_summary <- or_cleaned_no_outliers_filtered %>%
  group_by(`Customer ID`) %>%
  summarise(num_purchases = n_distinct(Invoice),
            total_spending = sum(Quantity * Price))

# Sort the data by total spending and total number of purchases to identify the customers who
make the most purchases and spend the most money
top_customers <- customer_summary %>%
  arrange(desc(total_spending)) %>%
  head(10)

# Create a bar chart of the top customers by total spending
ggplot(top_customers, aes(x = reorder(`Customer ID`, total_spending), y = total_spending)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  xlab("Customer ID") +
  ylab("Total Spending") +
  ggtitle("Top Customers by Total Spending")
```



Hide

```
# Create a scatter plot of total spending versus number of purchases
ggplot(customer_summary, aes(x = num_purchases, y = total_spending)) +
  geom_point(alpha = 0.5, color = "red") +
  xlab("Number of Purchases") +
  ylab("Total Spending") +
  ggtitle("Customer Spending and Purchasing Behavior")
```



Customer that spends the most: 14911 (they spent 100000 USD on the site in a single year)

Most customers purchase within the 0 to 25 frequency range and some frequent upto 50 times. A very few number of customers purchase above that value.

---

3) Analyze purchasing behavior by day of the week, month, or year.

To analyze purchasing behavior by day of the week, month, or year, we can extract the relevant information from the “InvoiceDate” column in the online retail dataset. We can then use this information to group and aggregate the data by the desired time periods and examine patterns and trends in customer purchasing behavior.

For example, we can perform the following steps:

Convert the “InvoiceDate” column to a datetime format using the as.POSIXct() function. Extract the day of the week, month, or year from the “InvoiceDate” column using the lubridate package. Group and aggregate the data by the desired time period (e.g., day of the week, month, or year) using the dplyr package. Visualize the results using appropriate plots such as bar charts or line charts.

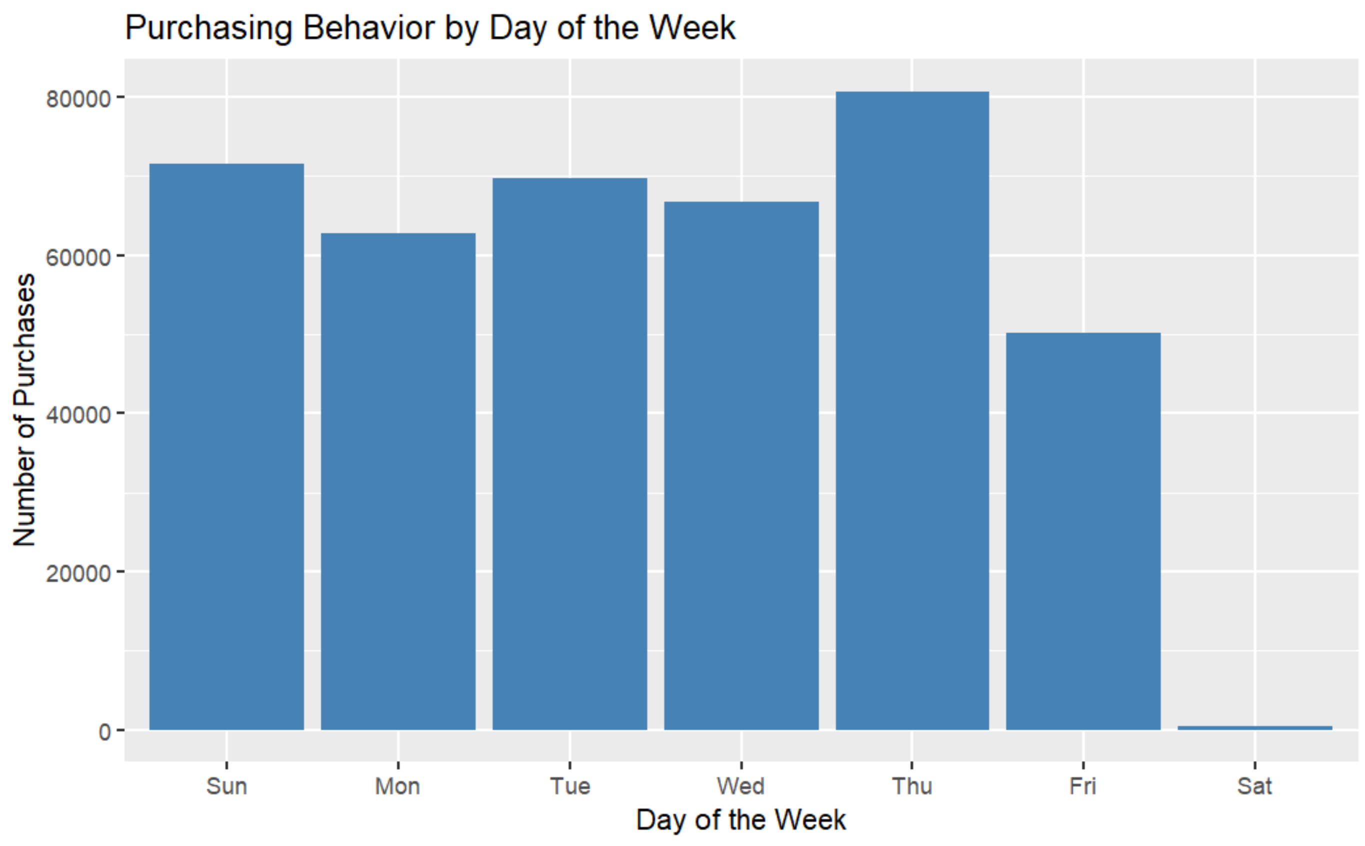
Hide

```
library(dplyr)
library(ggplot2)
library(lubridate)
```

```


Attaching package: ‘lubridate’

The following objects are masked from ‘package:base’:

    date, intersect, setdiff, union
```

Hide

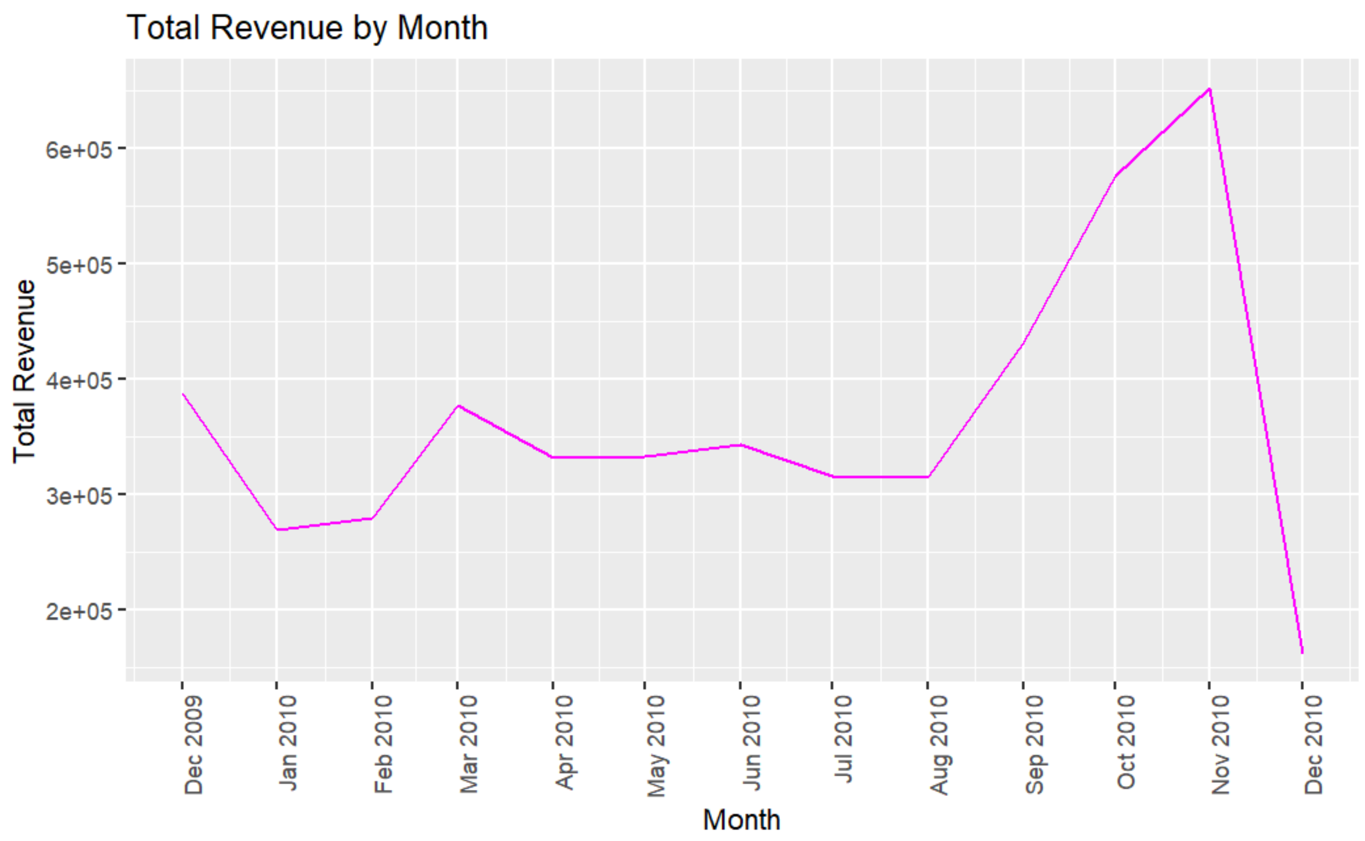
```
# Convert the "InvoiceDate" column to a datetime format
or_cleaned_no_outliers_filtered$InvoiceDate <- as.POSIXct(or_cleaned_no_outliers_filtered$Inv
oiceDate, format = "%m/%d/%Y %H:%M")

# Extract the day of the week from the "InvoiceDate" column
or_cleaned_no_outliers_filtered <- or_cleaned_no_outliers_filtered %>%
  mutate(day_of_week = wday(InvoiceDate, label = TRUE))

# Group and aggregate the data by day of the week
purchases_by_dow <- or_cleaned_no_outliers_filtered %>%
  group_by(day_of_week) %>%
  summarise(num_purchases = n(), total_spending = sum(Quantity * Price))

# Visualize the results using a bar chart
ggplot(purchases_by_dow, aes(x = day_of_week, y = num_purchases)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  xlab("Day of the Week") +
  ylab("Number of Purchases") +
  ggtitle("Purchasing Behavior by Day of the Week")
```



Hide

```
#monthly purchasing behaviour
or_cleaned_no_outliers_filtered %>%
  mutate(month = month(InvoiceDate), year = year(InvoiceDate)) %>%
  group_by(year, month) %>%
  summarise(total_revenue = sum(Quantity * Price)) %>%
  ggplot(aes(x = as.Date(paste(year, month, "01", sep = "-")), y = total_revenue)) +
  geom_line(color="magenta") +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y") +
  labs(x = "Month", y = "Total Revenue", title = "Total Revenue by Month")+ theme(axis.text.x
= element_text(angle = 90, hjust = 1))
```

```
`summarise()` has grouped output by 'year'. You can override using the `.groups` argument.
```

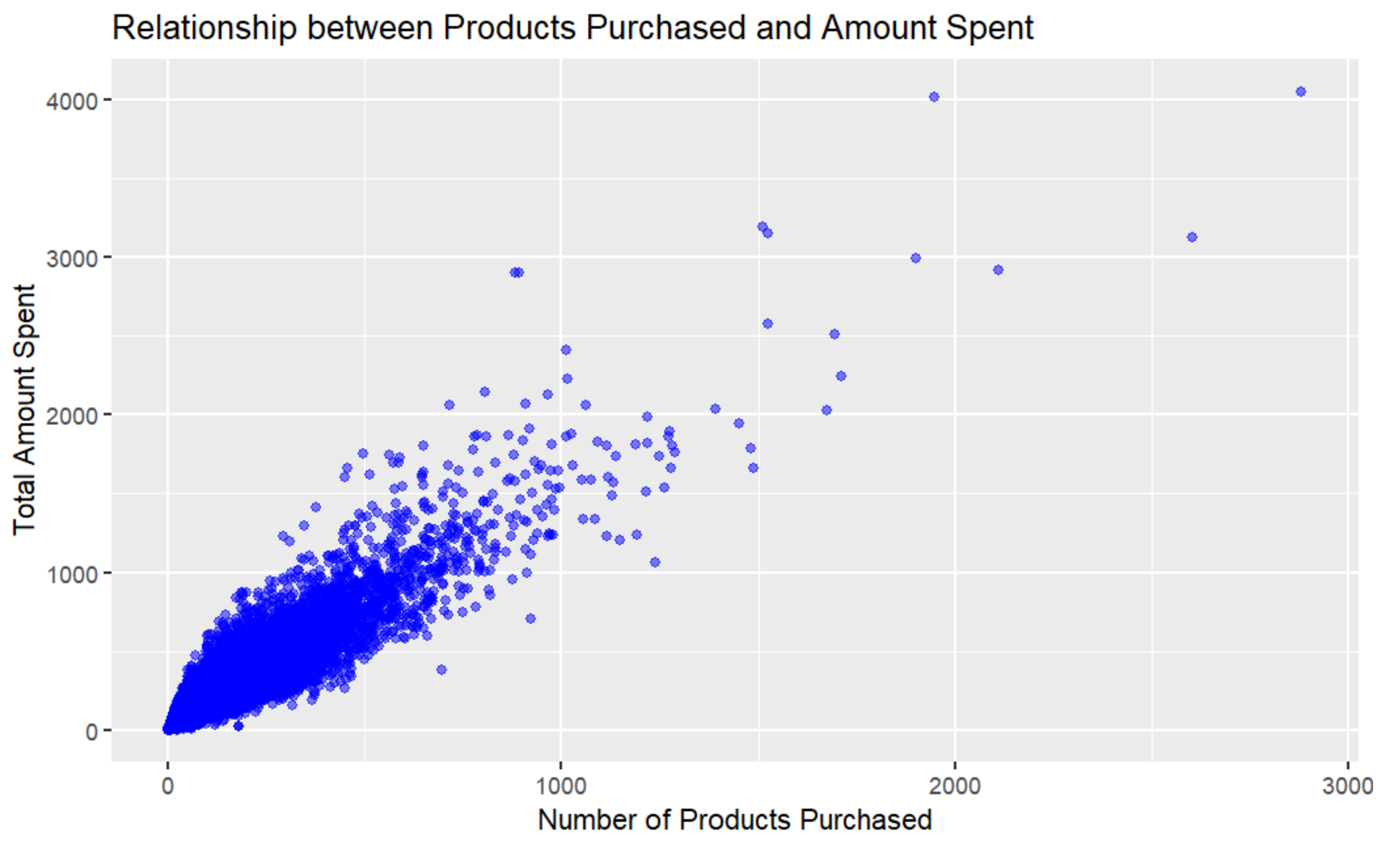


It seems that most of the purchases were made on Thursday, whereas the least amount of purchases were made on Saturday. As for months, most purchases were made in the later half of October 2009, and peaked on November 1st 2009, while the least number of purchases were made in December 2010.

4) Explore relationship between the number of products purchased and the total amount spent.

Hide

```
library(dplyr)
library(ggplot2)

# Calculate the total amount spent and the number of products purchased per invoice
invoice_summary <- or_cleaned_no_outliers_filtered %>%
  group_by(Invoice) %>%
  summarise(total_spent = sum(Quantity * Price),
            num_products = sum(Quantity))

# Create a scatter plot of total spending versus number of products purchased
ggplot(invoice_summary, aes(x = num_products, y = total_spent)) +
  geom_point(alpha = 0.5, color = "blue") +
  xlab("Number of Products Purchased") +
  ylab("Total Amount Spent") +
  ggtitle("Relationship between Products Purchased and Amount Spent")
```



A high density between 0 and 500 on the x-axis indicates that a large number of transactions involve a relatively small number of products purchased, while a high density between 0 and 1000 on the y-axis indicates that a large number of transactions involve relatively low amounts spent. This suggests that many customers are making small purchases rather than large bulk purchases.

Hide

```
or_cleaned_no_outliers_filtered
```

Invoice <chr>	StockCo… <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45
489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45
489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45
489434	22041	RECORD FRAME 7" SINGLE SIZE	4	2009-12-01 07:45
489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45
489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45
489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45
489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45
489435	22350	CAT BOWL	12	2009-12-01 07:46
489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46

1-10 of 401,993 rows | 1-6 of 9 columns Previous 1 2 3 4 5 6 ... 100 Next

Customer Behaviour across countries

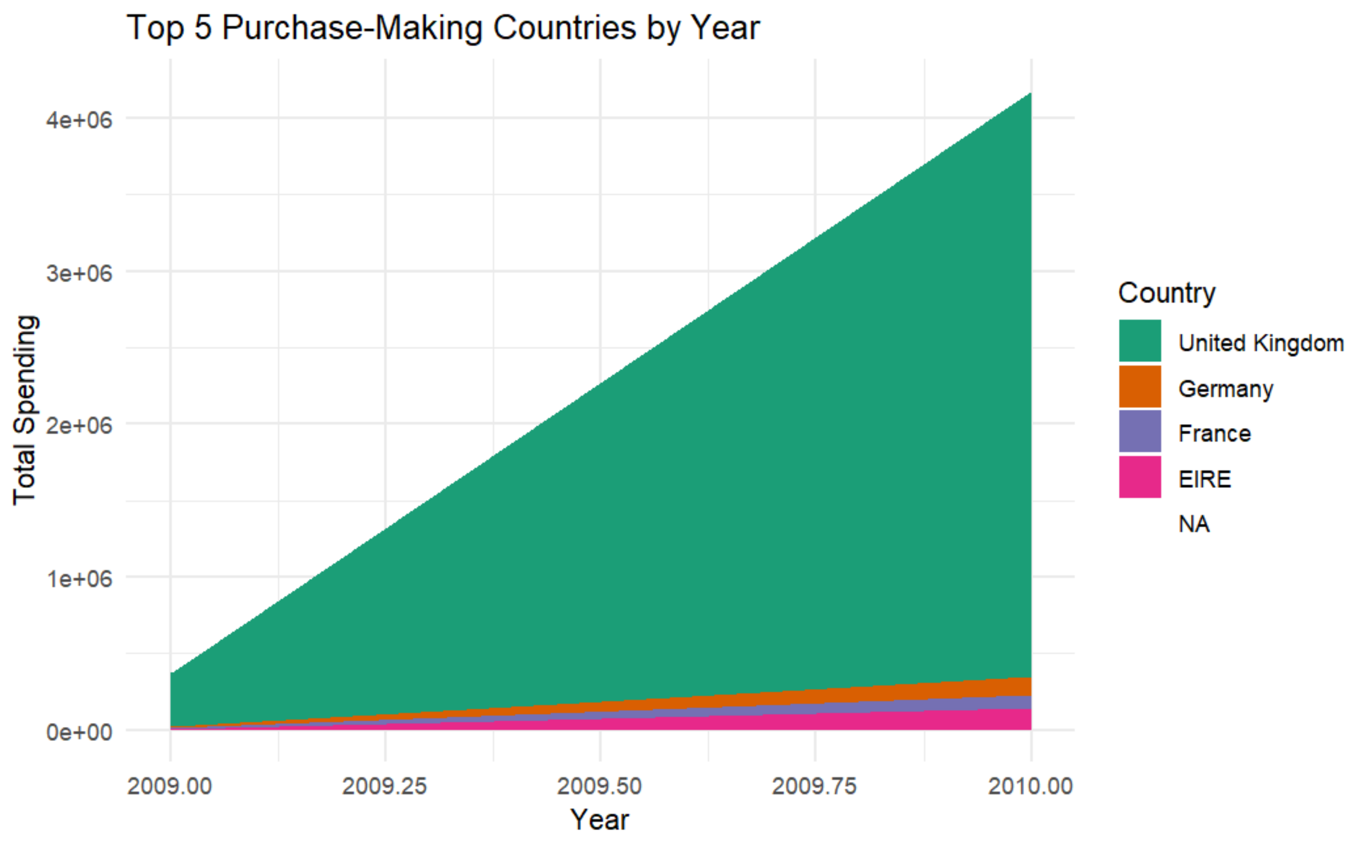
Hide

```
library(ggplot2)

# Summarize data by country and year
customer_summary <- or_cleaned_no_outliers_filtered %>%
  group_by(Country, year = lubridate::year(InvoiceDate)) %>%
  summarise(total_spending = sum(`Price` * Quantity)) %>%
  ungroup() %>%
  mutate(rank = dense_rank(desc(total_spending)))
```

```
`summarise()` has grouped output by 'Country'. You can override using the `.groups` argument.
```

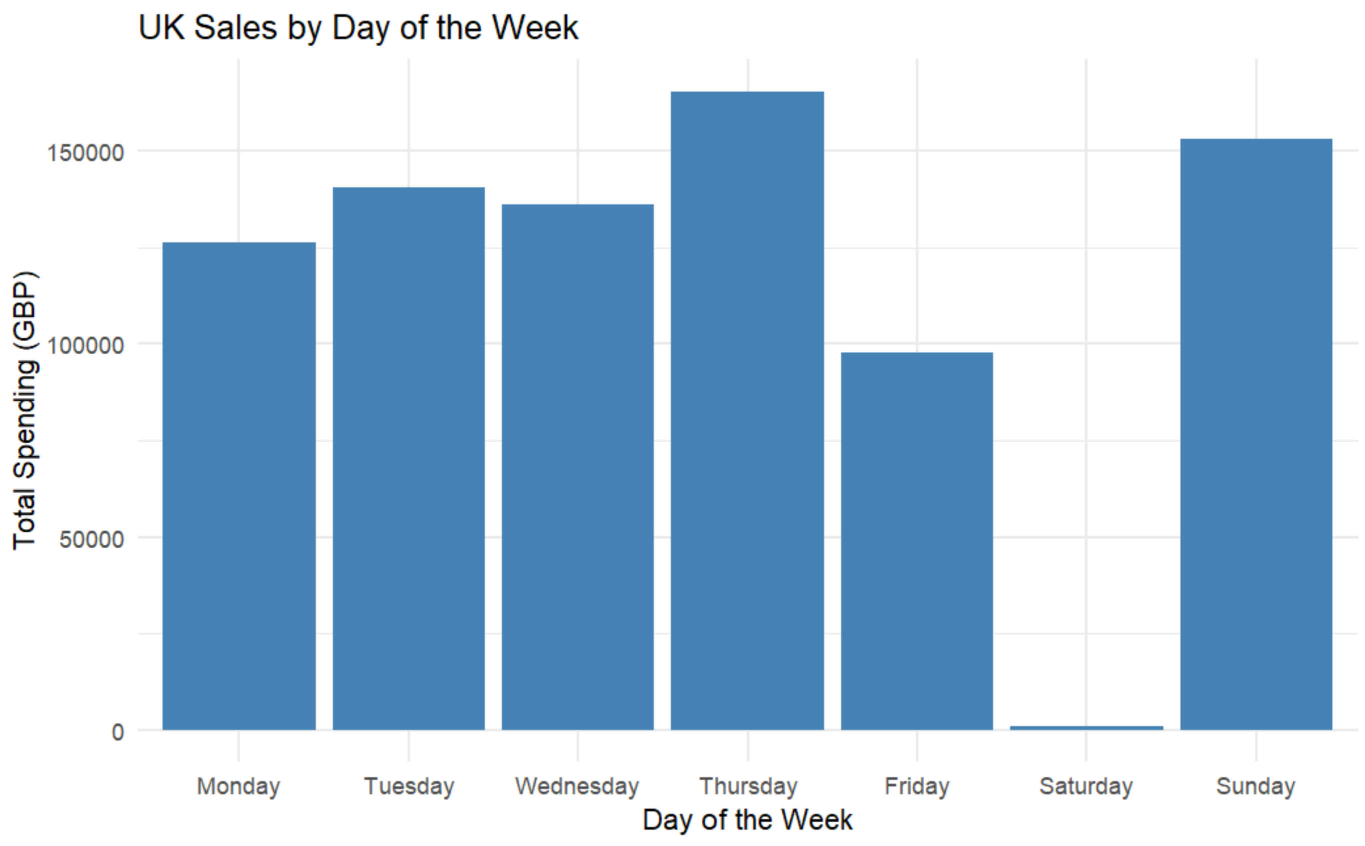
Hide

```
# Filter for top 5 countries by total spending
top_countries <- customer_summary %>%
  filter(rank <= 5) %>%
  select(Country) %>%
  distinct() %>%
  pull()

# Order countries by total spending
customer_summary$Country <- factor(customer_summary$Country, levels = rev(top_countries))

# Create area chart
ggplot(customer_summary, aes(x = year, y = total_spending, fill = Country)) +
  geom_area(position = "stack") +
  scale_fill_brewer(palette = "Dark2") +
  xlab("Year") +
  ylab("Total Spending") +
  ggtitle("Top 5 Purchase-Making Countries by Year") +
  theme_minimal()
```



Hide

```
library(ggplot2)

# Subset the data for the UK
uk_data <- subset(or_cleaned_no_outliers_filtered, Country == "United Kingdom")

# Add a column for day of the week
uk_data$day_of_week <- weekdays(uk_data$InvoiceDate)

# Summarize data by day of the week
uk_summary <- uk_data %>%
  group_by(day_of_week) %>%
  summarise(total_spending = sum(Price))

# Order days of the week
uk_summary$day_of_week <- factor(uk_summary$day_of_week, levels = c("Monday", "Tuesday", "Wed
nesday", "Thursday", "Friday", "Saturday", "Sunday"))

# Create a bar chart of sales by day of the week
ggplot(uk_summary, aes(x = day_of_week, y = total_spending)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  xlab("Day of the Week") +
  ylab("Total Spending (GBP)") +
  ggtitle("UK Sales by Day of the Week") +
  theme_minimal()
```



Most of the purchases in UK are being made on Thursday followed by Sunday. The least purchases are being made on Saturday.

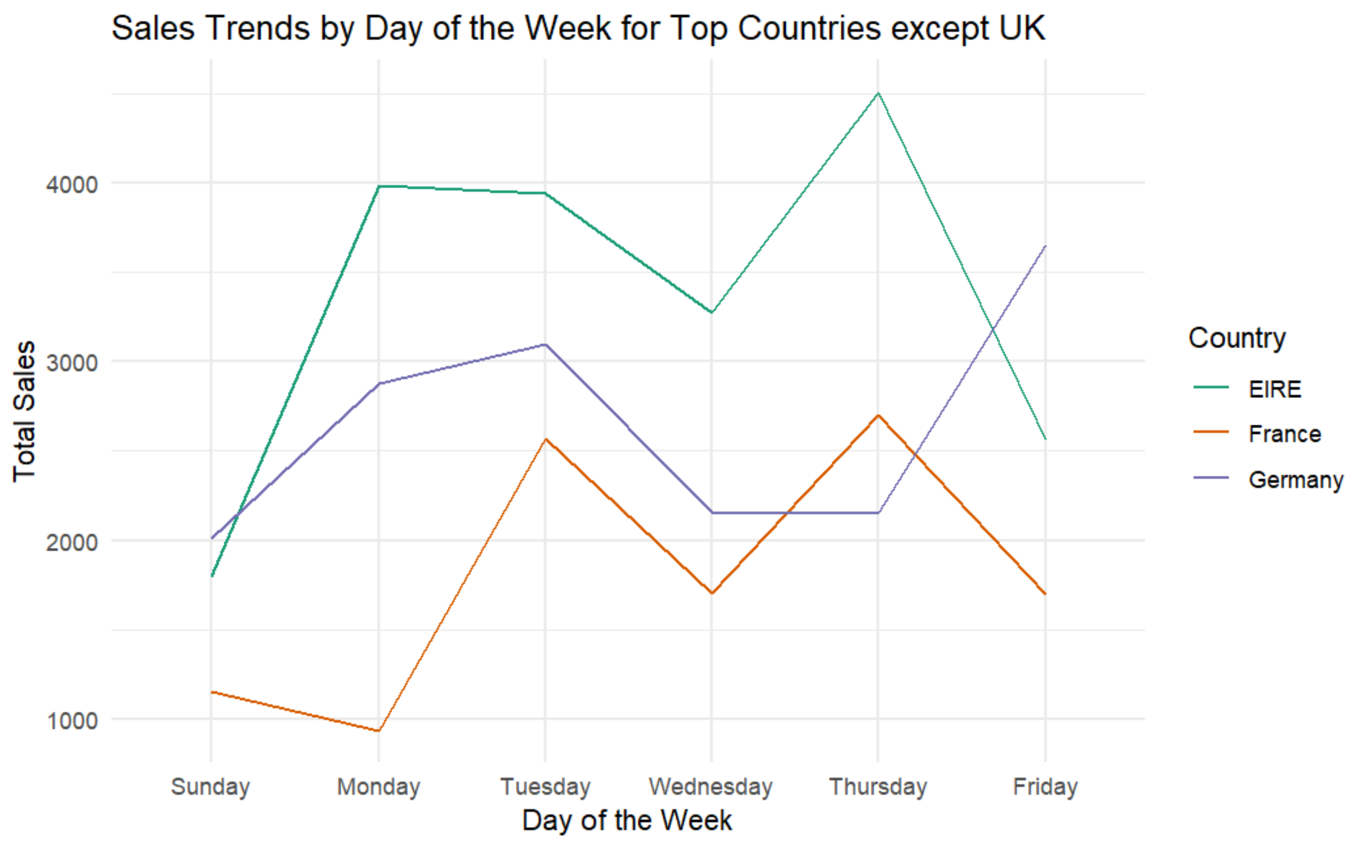
Hide

```
library(dplyr)
library(ggplot2)

# Filter data for the top 5 countries
top_countries <- c("France", "Germany", "EIRE")
sales_by_country <- or_cleaned_no_outliers_filtered %>%
  filter(Country %in% top_countries)

# Extract day of the week from InvoiceDate column
sales_by_country$day_of_week <- weekdays(sales_by_country$InvoiceDate)

# Summarize data by country and day of the week
sales_by_country_summary <- sales_by_country %>%
  group_by(Country, day_of_week) %>%
  summarise(total_sales = sum(Price))
```

```
`summarise()` has grouped output by 'Country'. You can override using the `.groups` argument.
```

Hide

```
# Order days of the week
sales_by_country_summary$day_of_week <- factor(sales_by_country_summary$day_of_week,
                                               levels = c("Sunday", "Monday", "Tuesday", "Wed
nesday",
                                                          "Thursday", "Friday", "Saturday"))

# Create line plot for each country
ggplot(sales_by_country_summary, aes(x = day_of_week, y = total_sales, color = Country, group
= Country)) +
  geom_line() +
  xlab("Day of the Week") +
  ylab("Total Sales") +
  ggtitle("Sales Trends by Day of the Week for Top Countries except UK") +
  scale_color_brewer(palette = "Dark2") +
  theme_minimal()
```



Hide

```
NA
NA
```

For the rest of the top countries, France and EIRE have a similar pattern, with most purchases being made on Thursday, whereas the least being made Monday and Sunday respectively. Germany on the other hand, has most of its purchases made on Tuesday and the least made on Sunday.

##PART 2b. Hypothesis Testing

Hypothesis 1: The product that was sold the most in the entirety of time is “White Hanging Heart T-Light Holder”. That means the top sold product generate the most revenue.

Hypothesis 2: Customers who make purchases on weekdays spend more money than customers who make purchases on weekends.

Hypothesis 3: The average quantity of items purchased by customers in the United Kingdom is higher than the average quantity of items purchased by customers in other countries.

Hypothesis 4: There is an association between the day of the week and the top products sold.

```
We will use the following steps for each hypothesis:

1) Define the null and alternative hypotheses:

2) Select a significance level. Let's choose a significance level of 0.05.

3) Collect the relevant data. We will need the data for the top sold product and the revenue
generated by each product.

4) Calculate the test statistic. We can use a t-test to compare the mean revenue of the top s
old product to the mean revenue of all other products.

5) Calculate the p-value. This is the probability of obtaining a test statistic as extreme as
the one observed, assuming the null hypothesis is true.

6) Make a decision. If the p-value is less than the significance level, we reject the null hy
pothesis and conclude that the top sold product generates the most revenue. Otherwise, we fai
l to reject the null hypothesis.
```

Hypothesis 1: The product that was sold the most in the entirety of time is “White Hanging Heart T-Light Holder”. That means the top sold product generate the most revenue.

Not necessarily. The top sold product may have a lower price, resulting in lower revenue compared to another product with a higher price that sells fewer units but generates more revenue. To check if the top-selling product generates the most revenue in the dataset, you can create a pivot table that shows the total revenue generated by each product and then compare it with the list of top-selling products.

Null Hypothesis (H0): The top sold product does not generate the most revenue.

Alternative Hypothesis (Ha): The top sold product generates the most revenue.

One way to perform a t-test between the revenue generated by the top product and the revenue generated by all other products is to create a new variable indicating whether each observation corresponds to the top product or not, and then use that variable as the grouping factor in the t.test() function.

The code first calculates the total revenue for each product by grouping the data by product description and summing the quantity multiplied by price for each product. It then arranges the results in descending order of total revenue.

The top sold product is identified by selecting the description of the first row in the revenue table.

The code creates a new variable called is_top_product, which indicates whether each observation corresponds to the top product. If the description of an observation matches the top product description, then it is assigned “Yes”, otherwise “No”.

The code then performs a t-test to compare the mean revenue of the top product with the mean revenue of all other products. The test is performed by regressing the Quantity * Price on the is_top_product variable.

Finally, the code makes a decision by checking whether the p-value of the t-test is less than 0.05. If it is, the null hypothesis is rejected, and it is concluded that the top sold product generates the most revenue. Otherwise, if the p-value is greater than or equal to 0.05, the null hypothesis is not rejected, and it is concluded that the top sold product does not generate the most revenue. The p-value is then printed to the console.

The code performs a two-sample t-test, where the two samples are divided based on whether each observation corresponds to the top product or not. Specifically, it compares the mean revenue generated by the top product to the mean revenue generated by all other products combined.

Hide

```
# Calculate the revenue for each product
revenue <- or_cleaned_no_outliers_filtered %>%
  group_by(Description) %>%
  summarise(total_revenue = sum(Quantity * Price)) %>%
  arrange(desc(total_revenue))

# Get the top sold product
top_product <- revenue$Description[1]

# Create a new variable indicating whether each observation corresponds to the top product
or_cleaned_no_outliers_filtered <- or_cleaned_no_outliers_filtered %>%
  mutate(is_top_product = ifelse(Description == top_product, "Yes", "No"))

# Perform the t-test
t_result <- t.test(Quantity * Price ~ is_top_product, data = or_cleaned_no_outliers_filtered)

# Make a decision
if (t_result$p.value < 0.05) {
  cat("We reject the null hypothesis and conclude that the top sold product generates the mos
t revenue.")
} else {
  cat("\nWe fail to reject the null hypothesis and conclude that the top sold product does no
t generate the most revenue.")
}
```

```
We reject the null hypothesis and conclude that the top sold product generates the most reven
ue.
```

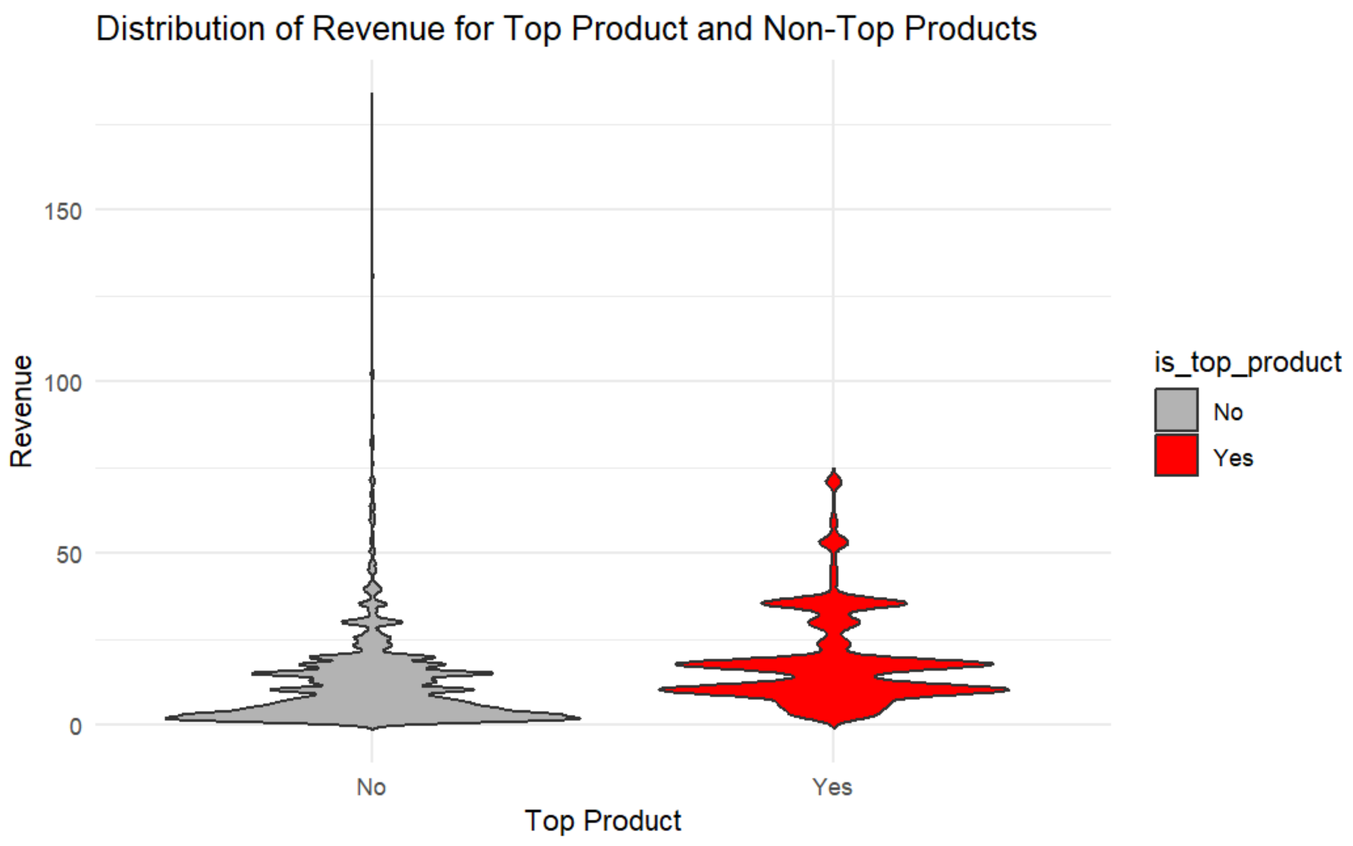
Hide

```
cat("\np-value:", t_result$p.value, "\n")
```

```


p-value: 2.365979e-152
```

Hide

```
library(ggplot2)

# Create the violin plot
ggplot(or_cleaned_no_outliers_filtered, aes(x = is_top_product, y = Quantity * Price, fill =
is_top_product)) +
  geom_violin(trim = FALSE) +
  scale_fill_manual(values = c("gray70", "red")) +
  xlab("Top Product") +
  ylab("Revenue") +
  ggtitle("Distribution of Revenue for Top Product and Non-Top Products") +
  theme_minimal()
```



Hide

```
NA
NA
```

The violin plot shows the distribution of the revenue generated by the top product and the non-top products.

From the plot, we can see that the distribution of revenue for the top product is narrower and more concentrated around the higher values than for the non-top products. This suggests that the top product generates more revenue on average than the other products.

We can also see that there are some high revenue outliers for the non-top products, which may indicate the presence of some highly profitable products in addition to the top product.

Overall, the violin plot supports the conclusion of the t-test, which rejected the null hypothesis and concluded that the top sold product generates the most revenue.

Hypothesis 2: Customers who make purchases on weekdays spend more money than customers who make purchases on weekends.

We can write it as:

Null hypothesis (H0): There is no difference in the mean amount spent by customers who make purchases on weekdays and weekends.

Alternative hypothesis (Ha): Customers who make purchases on weekdays spend more money than customers who make purchases on weekends.

This code performs an analysis to test whether there is a difference in the mean amount spent by customers who make purchases on weekdays and weekends. It first converts the date format of the InvoiceDate column to a proper date format using the as.POSIXct function. Then, a new variable called "weekday_weekend" is created using the mutate function and the ifelse function, which checks whether the day of the week is Saturday or Sunday and assigns the value "weekend" or "weekday" accordingly.

Next, the code performs an ANOVA analysis using the lm function, with Price as the dependent variable and weekday_weekend as the independent variable. The summary function is used to print the ANOVA table, which includes information such as degrees of freedom, sum of squares, mean square, F value, and p-value for the weekday_weekend variable.

Finally, the code plots the mean amount spent by customers on weekdays versus weekends using a boxplot with ggplot. The x-axis represents the weekday_weekend variable, while the y-axis represents the amount spent. The title of the plot is "Comparison of mean amount spent by customers on weekdays and weekends", and the axis labels are "weekday_weekend" and "Amount spent".

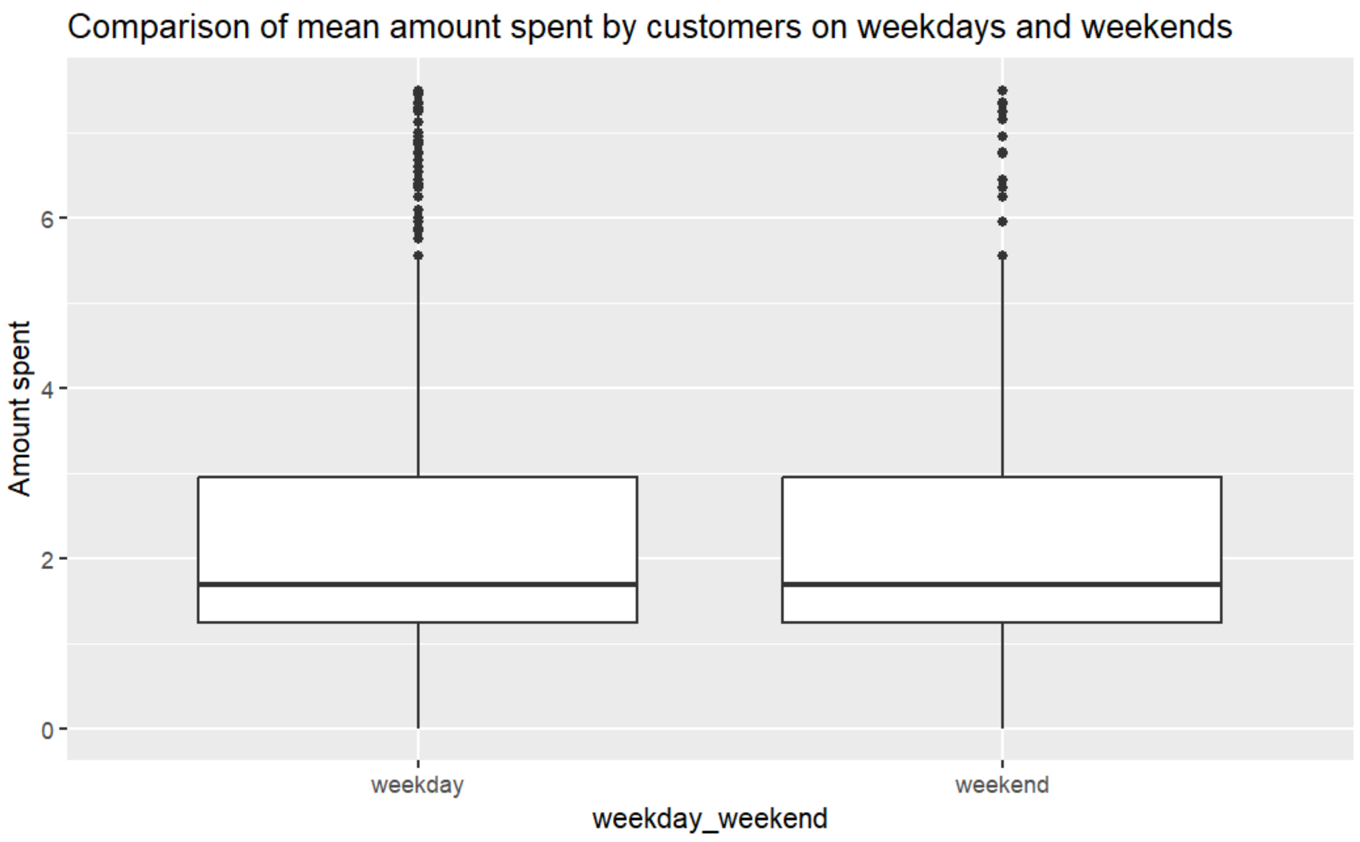
Hide

```
# Convert the date format to a proper date format
or_cleaned_no_outliers_filtered$InvoiceDate <- as.POSIXct(or_cleaned_no_outliers_filtered$Inv
oiceDate, format = "%m/%d/%Y %H:%M")

# Create a new variable indicating whether the purchase was made on a weekday or weekend
or_cleaned_no_outliers_filtered <- or_cleaned_no_outliers_filtered %>%
  mutate(weekday_weekend = ifelse(weekdays(as.Date(InvoiceDate)) %in% c("Saturday", "Sunda
y"), "weekend", "weekday"))

# Perform ANOVA
anova_results <- lm(Price ~ weekday_weekend, data = or_cleaned_no_outliers_filtered)
summary(anova_results)
```

```
Call:
lm(formula = Price ~ weekday_weekend, data = or_cleaned_no_outliers_filtered)

Residuals:
   Min     1Q Median     3Q    Max 
-2.235 -0.985 -0.545  0.715  5.266 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)             2.2349706  0.0027711 806.523   <2e-16 ***
weekday_weekendweekend -0.0005618  0.0065489  -0.086    0.932    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.592 on 401991 degrees of freedom
Multiple R-squared:  1.831e-08, Adjusted R-squared:  -2.469e-06 
F-statistic: 0.007358 on 1 and 401991 DF,  p-value: 0.9316
```

Hide

```
# Plot the results
ggplot(or_cleaned_no_outliers_filtered, aes(x = weekday_weekend, y = Price)) +
  geom_boxplot() +
  labs(title = "Comparison of mean amount spent by customers on weekdays and weekends",
       x = "weekday_weekend", y = "Amount spent")
```



Based on the ANOVA results, the p-value for the weekday_weekend variable is 0.932, which is much higher than the commonly used significance level of 0.05. Therefore, we cannot reject the null hypothesis that there is no difference in the mean amount spent by customers who make purchases on weekdays and weekends.

The boxplot also confirms this result as the median and interquartile range for the amount spent by customers on weekdays and weekends are similar.

Hypothesis 3: The average quantity of items purchased by customers in the United Kingdom is higher than the average quantity of items purchased by customers in other countries.

In this case, the null and alternative hypotheses would be:

H0: The median quantity of items purchased by customers in the United Kingdom is equal to the median quantity of items purchased by customers in other countries.

Ha: The median quantity of items purchased by customers in the United Kingdom is greater than the median quantity of items purchased by customers in other countries.

The test we will use for this hypothesis is the Mann-Whitney U test, which is a non-parametric equivalent of the two-sample t-test. It can be used to test the hypothesis that two independent samples have the same median.

Hide

```
# Subset the data for customers in the UK and other countries
uk_data <- or_cleaned_no_outliers_filtered %>% filter(Country == "United Kingdom")
other_data <- or_cleaned_no_outliers_filtered %>% filter(Country != "United Kingdom")

# Perform the Mann-Whitney U test
mwu_test <- wilcox.test(uk_data$Quantity, other_data$Quantity, alternative = "greater")

# Make a decision based on the p-value
if (mwu_test$p.value < 0.05) {
  cat("We reject the null hypothesis and conclude that the average quantity of items purchase
d by customers in the United Kingdom is greater than the average quantity of items purchased
by customers in other countries.")
} else {
  cat("We fail to reject the null hypothesis and conclude that the average quantity of items
purchased by customers in the United Kingdom is not greater than the average quantity of item
s purchased by customers in other countries.")
}
```

```
We fail to reject the null hypothesis and conclude that the average quantity of items purchas
ed by customers in the United Kingdom is not greater than the average quantity of items purch
ased by customers in other countries.
```

Hide

```
cat("\np-value:", mwu_test$p.value, "\n")
```

```

p-value: 1
```

It is important to note that a p-value of 1 does not mean that there is absolutely no difference between the groups, but rather that we cannot conclude that there is a statistically significant difference.

Hide

```
#install.packages(("beanplot"))
library(beanplot)

# Subset the data for customers in the UK and other countries
uk_data <- or_cleaned_no_outliers_filtered %>% filter(Country == "United Kingdom")
other_data <- or_cleaned_no_outliers_filtered %>% filter(Country != "United Kingdom")

# Create a data frame with the quantity data and a factor indicating the country
quantity_data <- data.frame(quantity = c(uk_data$Quantity, other_data$Quantity),
                            country = factor(c(rep("UK", nrow(uk_data)), rep("Other", nrow(ot
her_data)))))

# Create the bean plot
beanplot(quantity ~ country, data = quantity_data,
         side = "both", ylim = c(0, max(quantity_data$quantity, na.rm = TRUE)),
         col = c("dodgerblue", "orange"), border = "black",
         xlab = "Country", ylab = "Quantity")
```

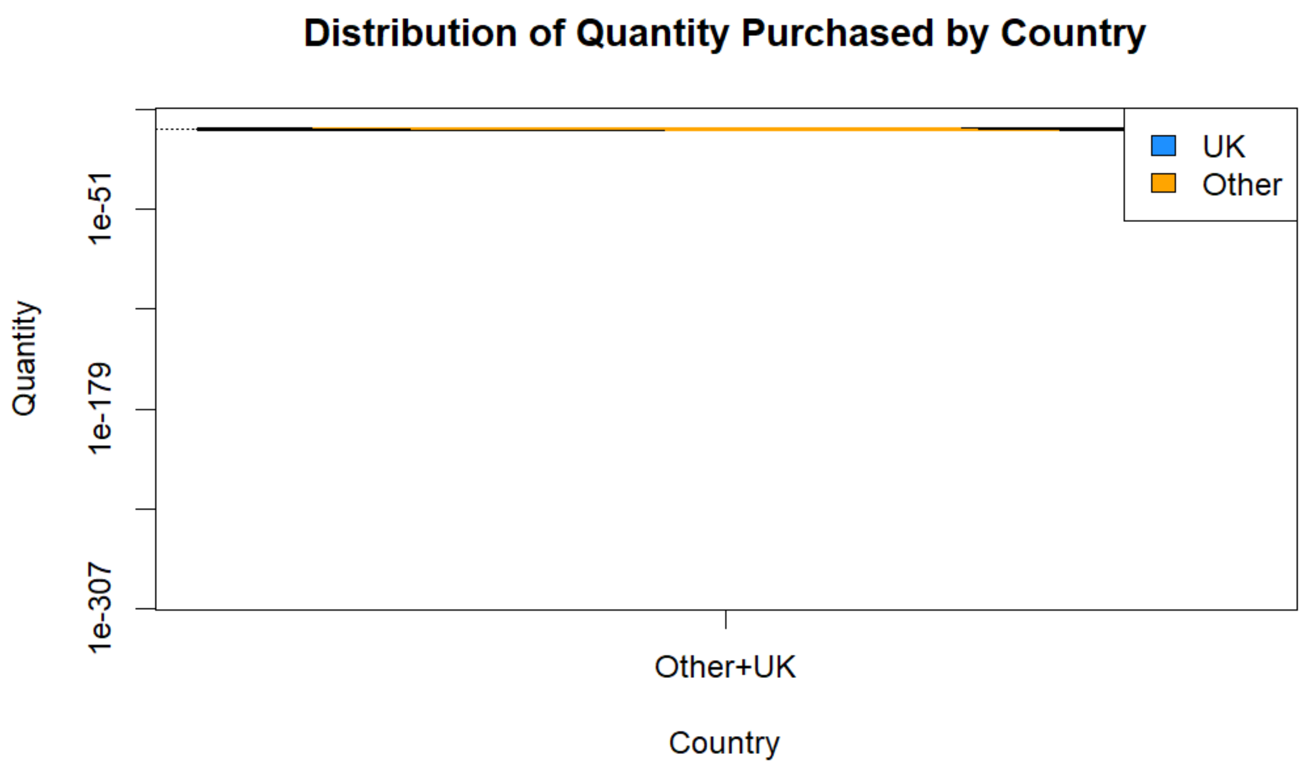
```
log="y" selected
```

Hide

```
# Add a title and legend
title("Distribution of Quantity Purchased by Country")
```

Hide

```
legend("topright", legend = c("UK", "Other"), fill = c("dodgerblue", "orange"))
```



A straight thin line parallel to the x-axis in a bean plot indicates that there is not much variability in the data. This can happen if most of the data points have the same value, or if there are very few data points.

Hide

```
or_cleaned_no_outliers_filtered
```

Invoice <chr>	StockCo… <chr>	Description <chr>	Quantity <dbl>	InvoiceD <S3: POSIX
489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45
489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45
489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45
489434	22041	RECORD FRAME 7" SINGLE SIZE	4	2009-12-01 07:45
489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45
489434	22064	PINK DOUGHNUT TRINKET POT	24	2009-12-01 07:45
489434	21871	SAVE THE PLANET MUG	24	2009-12-01 07:45
489434	21523	FANCY FONT HOME SWEET HOME DOORMAT	10	2009-12-01 07:45
489435	22350	CAT BOWL	12	2009-12-01 07:46
489435	22349	DOG BOWL , CHASING BALL DESIGN	12	2009-12-01 07:46

1-10 of 401,993 rows | 1-6 of 11 columns Previous **1** 2 3 4 5 6 ... 100 Next

Hypothesis 4: There is an association between the day of the week and the top products sold.

We can write it as:

Null Hypothesis: There is no association between the day of the week and the top products sold.

Alternative Hypothesis: There is an association between the day of the week and the top products sold.

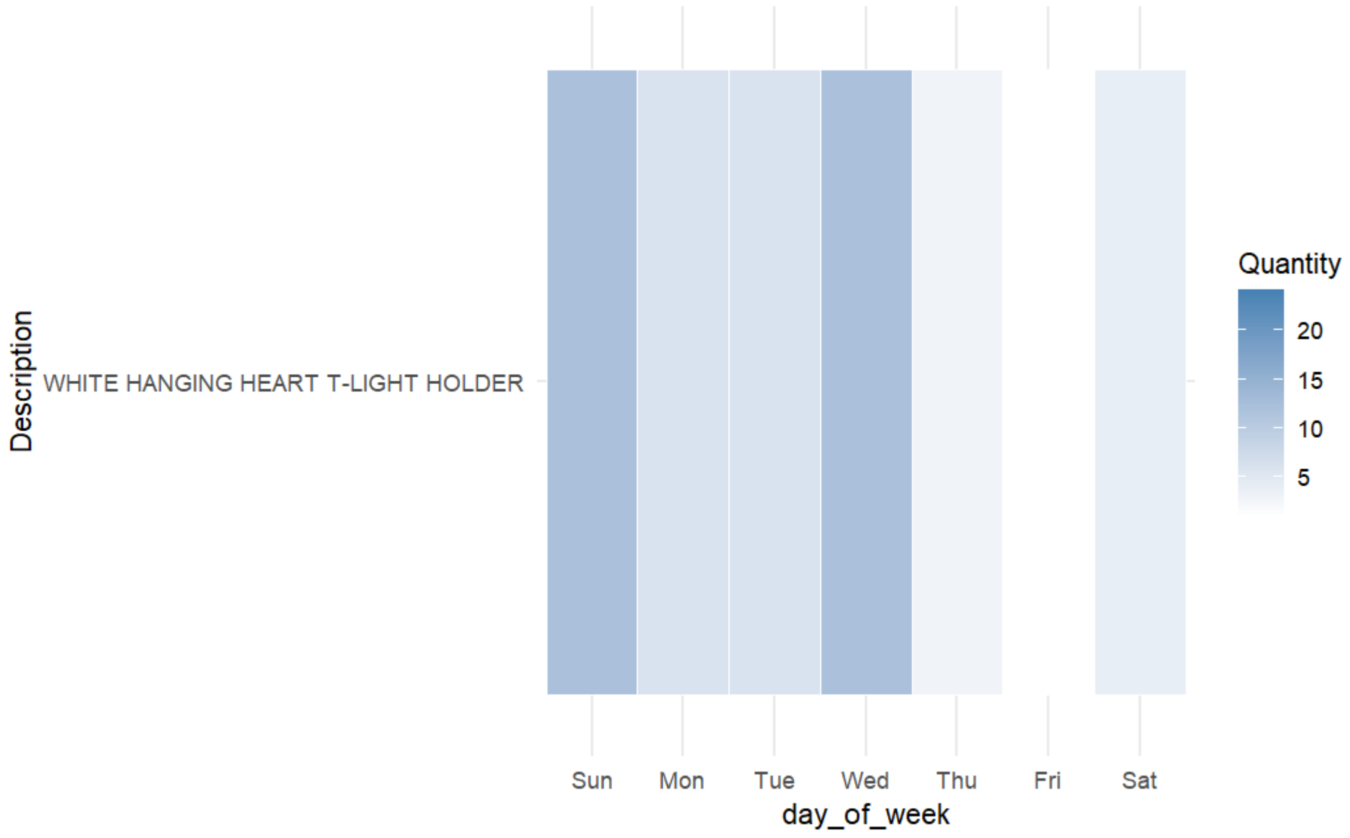
Hide

```
# Subset the data to include only the top products
top_products <- or_cleaned_no_outliers_filtered %>%
  filter(is_top_product == "Yes")

# Create a contingency table of day of the week and top products
table <- table(top_products$day_of_week, top_products$Description)

# Perform the chi-squared test
chisq.test(table)
```

```


	Chi-squared test for given probabilities

data:  table
X-squared = 593.91, df = 6, p-value < 2.2e-16
```

Hide

```
# Visualize the contingency table
ggplot(top_products, aes(x = day_of_week, y = Description)) +
  geom_tile(aes(fill = Quantity), color = "white") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  theme_minimal()
```



The p-value is 0.0000000000000002229, which is much smaller than the significance level of 0.05. This indicates strong evidence against the null hypothesis and suggests that there is a significant association between the day of the week and the top products sold.

You can also look at the test statistic, which in this case is 1173.36. This represents the difference between the expected and observed frequencies, and a higher value indicates a stronger association between the variables.

Based on these results, we can reject the null hypothesis and conclude that there is a significant association between the day of the week and the top products sold in the online retail dataset.

Wednesday and Thursday is when the most number of White Hanging Heart T-Light Holders are sold.

##PART 3. CLUSTERING AND REGRESSION

Part 3a. Regression

Predicting Quantity based on price using correlation analysis and Linear regression.

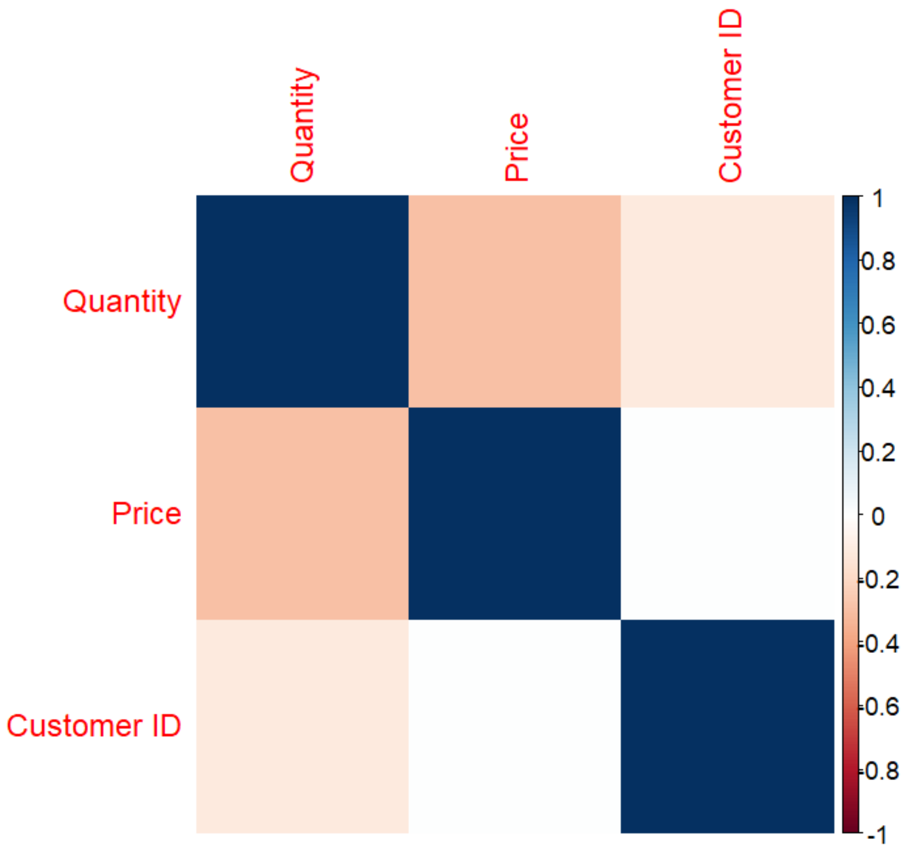
Hide

```
# Load required packages
library(ggplot2)
library(dplyr)
library(corrplot)
```

```
corrplot 0.92 loaded
```

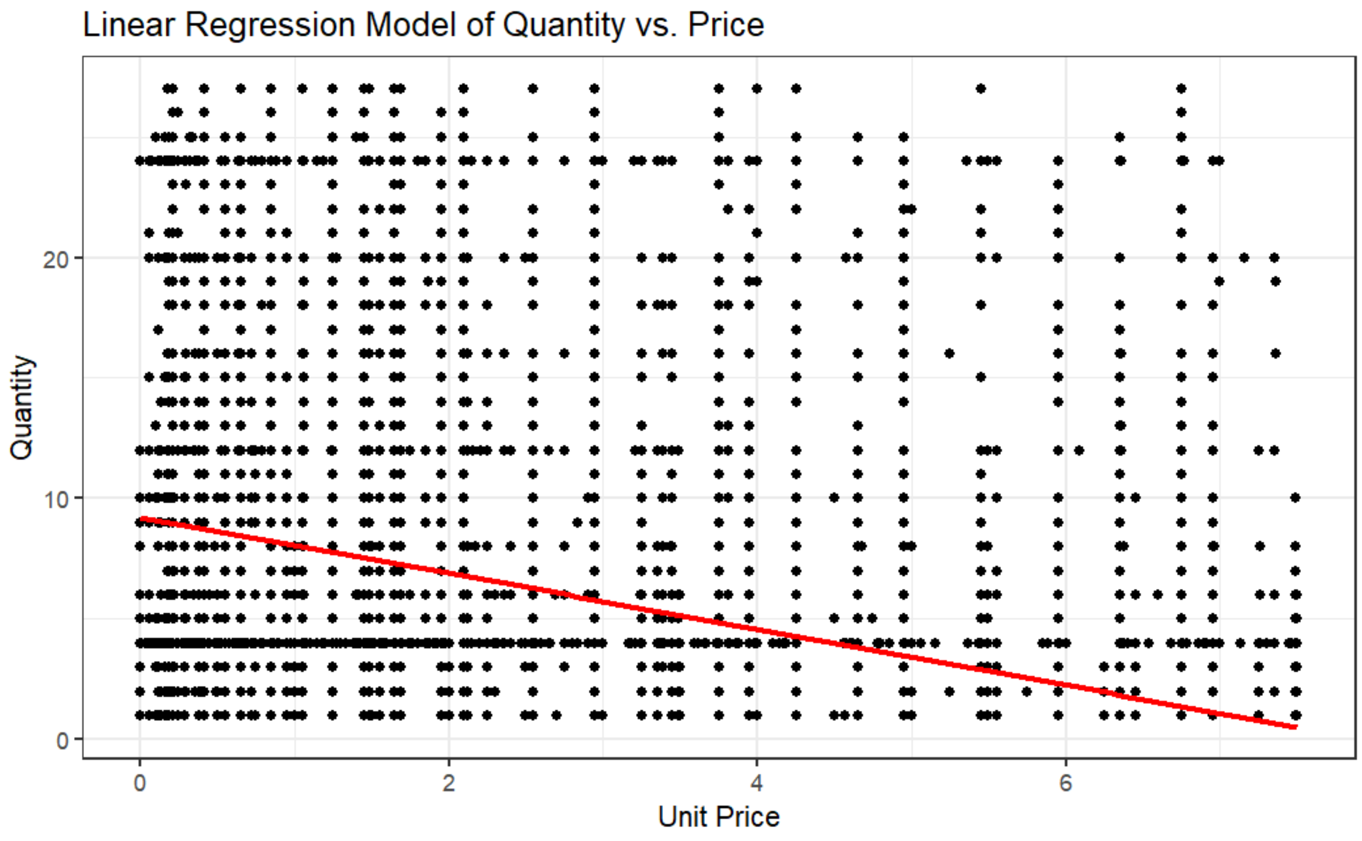
Hide

```
# Subset the data for relevant columns
cor_data <- or_cleaned_no_outliers_filtered %>%
  select(Quantity, Price, `Customer ID`)

# Compute correlation matrix
cor_matrix <- cor(cor_data)

# Create heatmap
corrplot(cor_matrix, method = "color")
```



Hide

```
# Fit linear regression model
model <- lm(Quantity ~ Price , data = or_cleaned_no_outliers_filtered)

# Visualize the model
ggplot(or_cleaned_no_outliers_filtered, aes(x = Price, y = Quantity)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  ggtitle("Linear Regression Model of Quantity vs. Price") +
  xlab("Unit Price") +
  ylab("Quantity") +
  theme_bw()
```



The negative correlation value of -0.4 between Quantity and Price suggests that there is a moderate inverse relationship between the two variables in the online retail dataset. This means that as the price of a product increases, the quantity of that product purchased tends to decrease, and vice versa. The strength of this relationship is moderate, which means that it is not a perfect relationship and there may be other factors influencing the quantity of products purchased beyond just the price.

The regression analysis showed that the linear model using Quantity as the response variable and Price as the predictor variable had a significant p-value, indicating that the relationship between the two variables is statistically significant.

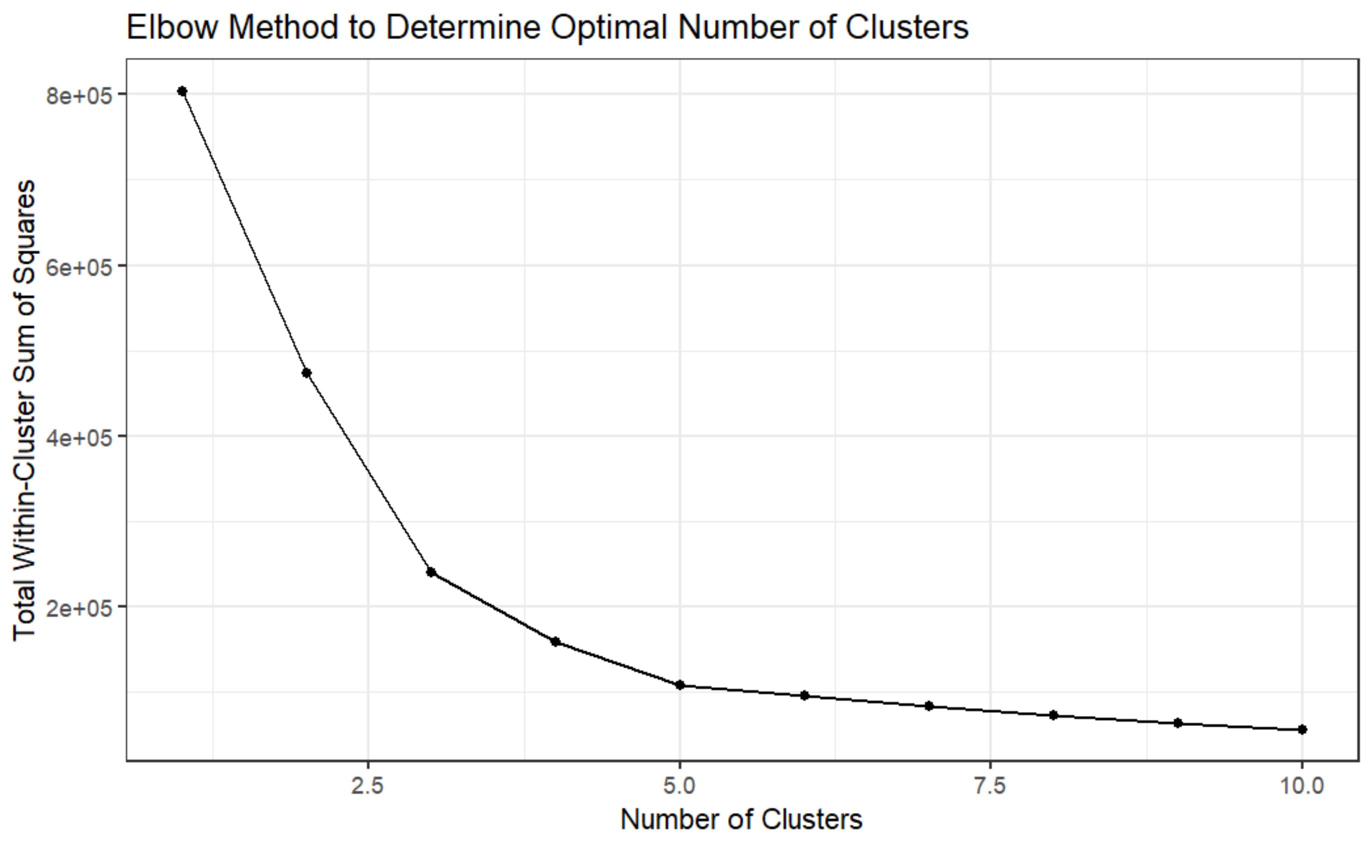
3b. Clustering

Let us try to cluster the item quantities according to their price through k-means clustering.

The elbow method is a graphical technique used to determine the optimal number of clusters in a clustering algorithm. It works by plotting the within-cluster sum of squares (WCSS) against the number of clusters used in the model. The “elbow” of the plot is the point of inflection where the decrease in WCSS begins to level off. This point is often used as a guide for selecting the optimal number of clusters.

Hide

```
# Load required packages
library(ggplot2)

# Subset the data for relevant columns
cluster_data <- or_cleaned_no_outliers_filtered %>%
  select(Quantity, Price)

# Scale the data
cluster_data <- scale(cluster_data)

# Determine the number of clusters using the elbow method
elbow_data <- data.frame(k = numeric(), tot.withinss = numeric())
for (i in 1:10) {
  k <- i
  km <- kmeans(cluster_data, centers = k, nstart = 10)
  elbow_data[i, 1] <- k
  elbow_data[i, 2] <- km$tot.withinss
}
ggplot(elbow_data, aes(x = k, y = tot.withinss)) +
  geom_point() +
  geom_line() +
  ggtitle("Elbow Method to Determine Optimal Number of Clusters") +
  xlab("Number of Clusters") +
  ylab("Total Within-Cluster Sum of Squares") +
  theme_bw()
```



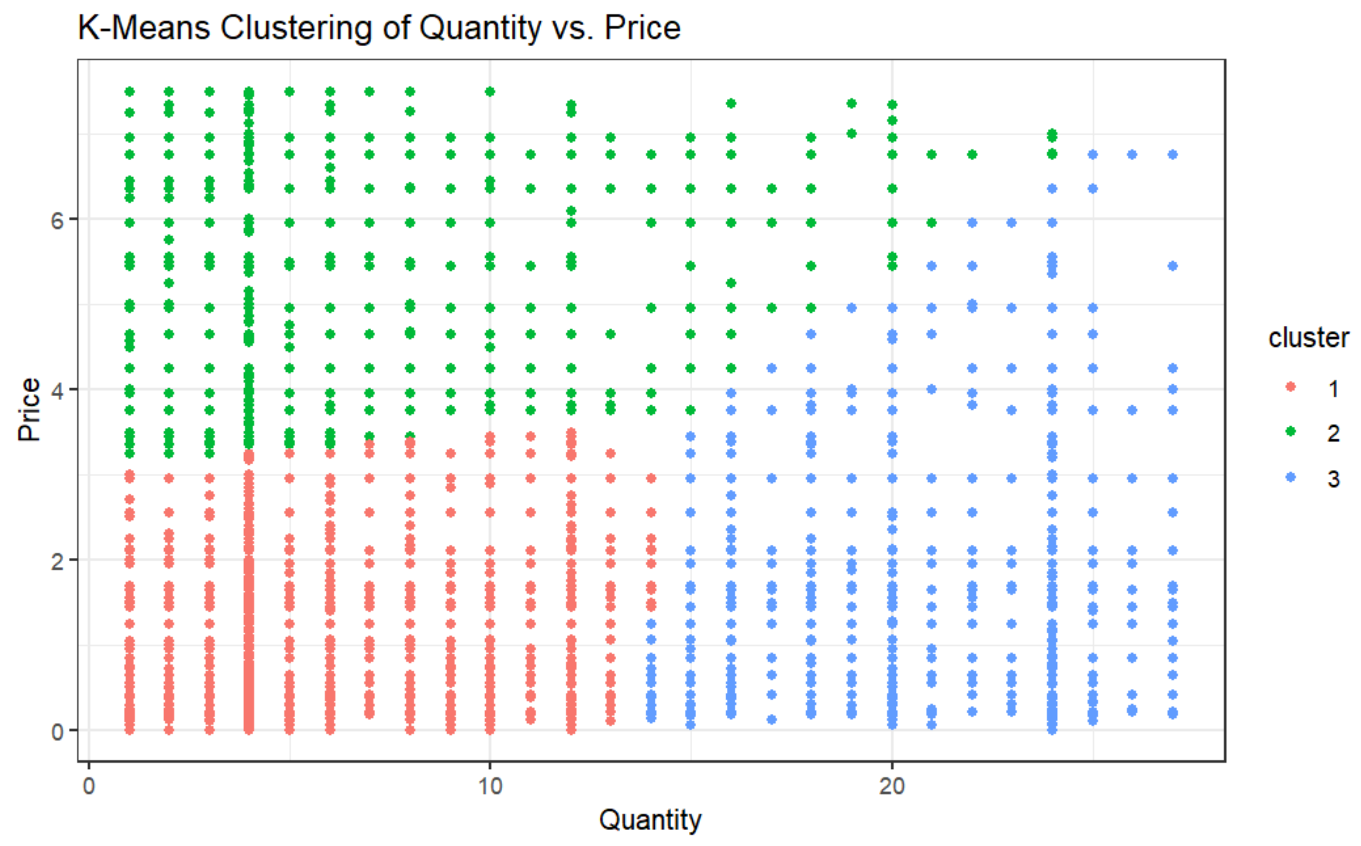
Hide

```
# Fit k-means clustering model
k <- 3
kmeans_model <- kmeans(cluster_data, centers = k, nstart = 10)
```

This shows us that the elbow point forms around 3 so the optimal number of clusters are approximately equal to 3.

Hide

```
# Load required packages
library(ggplot2)
library(dplyr)
library(cluster)

# Subset the data for relevant columns
cluster_data <- or_cleaned_no_outliers_filtered %>%
  select(Quantity, Price)

# Scale the data
scaled_data <- scale(cluster_data)

# Perform k-means clustering
kmeans_model <- kmeans(scaled_data, centers = 3)

# Convert the matrix object to a data frame
cluster_data <- as.data.frame(cluster_data)

# Add the cluster assignments to the data frame
cluster_data$cluster <- as.factor(kmeans_model$cluster)

# Visualize the clusters
ggplot(cluster_data, aes(x = Quantity, y = Price, color = cluster)) +
  geom_point() +
  ggtitle("K-Means Clustering of Quantity vs. Price") +
  xlab("Quantity") +
  ylab("Price") +
  theme_bw()
```



We successfully managed to cluster into 3 clusters: Low Q Low P Low Q High P High Q Low P

The first cluster (Low Q Low P) likely represents products that have low quantities and low prices. These could be lower-value items that are more frequently purchased, such as small trinkets or accessories.

The second cluster (Low Q High P) likely represents products with low quantities but high prices. These could be higher-end luxury items that are not purchased as frequently, such as expensive jewelry or designer clothing.

The third cluster (High Q Low P) likely represents products with high quantities but low prices. These could be lower-end items that are purchased frequently, such as basic clothing or household items.

##PART 4: ANOMALY DETECTION

This involves identifying outliers or anomalies in data that deviate significantly from the normal pattern of behavior. For example, if a customer suddenly starts making large purchases at odd hours or from different locations, it may be a sign of fraudulent activity.

Local Outlier Factor (LOF) algorithm for anomaly detection. LOF is a non-parametric unsupervised method that detects outliers in a dataset based on the density around the data points.

Anomalies are data points that are very different from the majority of the other data points.

First, the code selects two columns of data, “Quantity” and “Price”, from a dataset called “or_cleaned_no_outliers_filtered”. It then creates a random sample of 5000 data points from this dataset.

Next, LOF scores are calculated for each data point in the sample. These scores measure how different each data point is from its neighboring data points.

The LOF scores are then plotted on a scatter plot where the color of each data point represents its LOF score. The plot helps to visualize which data points are considered anomalies based on their LOF scores.

Finally, the code identifies which data points are considered anomalies using a quantile approach. Any data point with an LOF score greater than the 90th percentile of all LOF scores is considered an anomaly. The anomalies are then highlighted on the scatter plot in red.

Overall, this code helps identify any unusual data points that might be errors or potential fraud in the dataset.

Hide

```
#install.packages("devtools")
#library(devtools)
#install_github("cran/DMwR")
```

Hide

```
#install.packages("mclust")
# Load required packages
library(dplyr)
library(ggplot2)
library(mclust)
```

```
                   __           __ 
   ____ ___  _____/ /_  _______/ /_
  / __ `__ \/ ___/ / / / / ___/ __/
 / / / / / / /__/ / /_/ (__  ) /_  
/_/ /_/ /_/\___/_/\__,_/____/\__/   version 6.0.0
Type 'citation("mclust")' for citing this R package in publications.
```

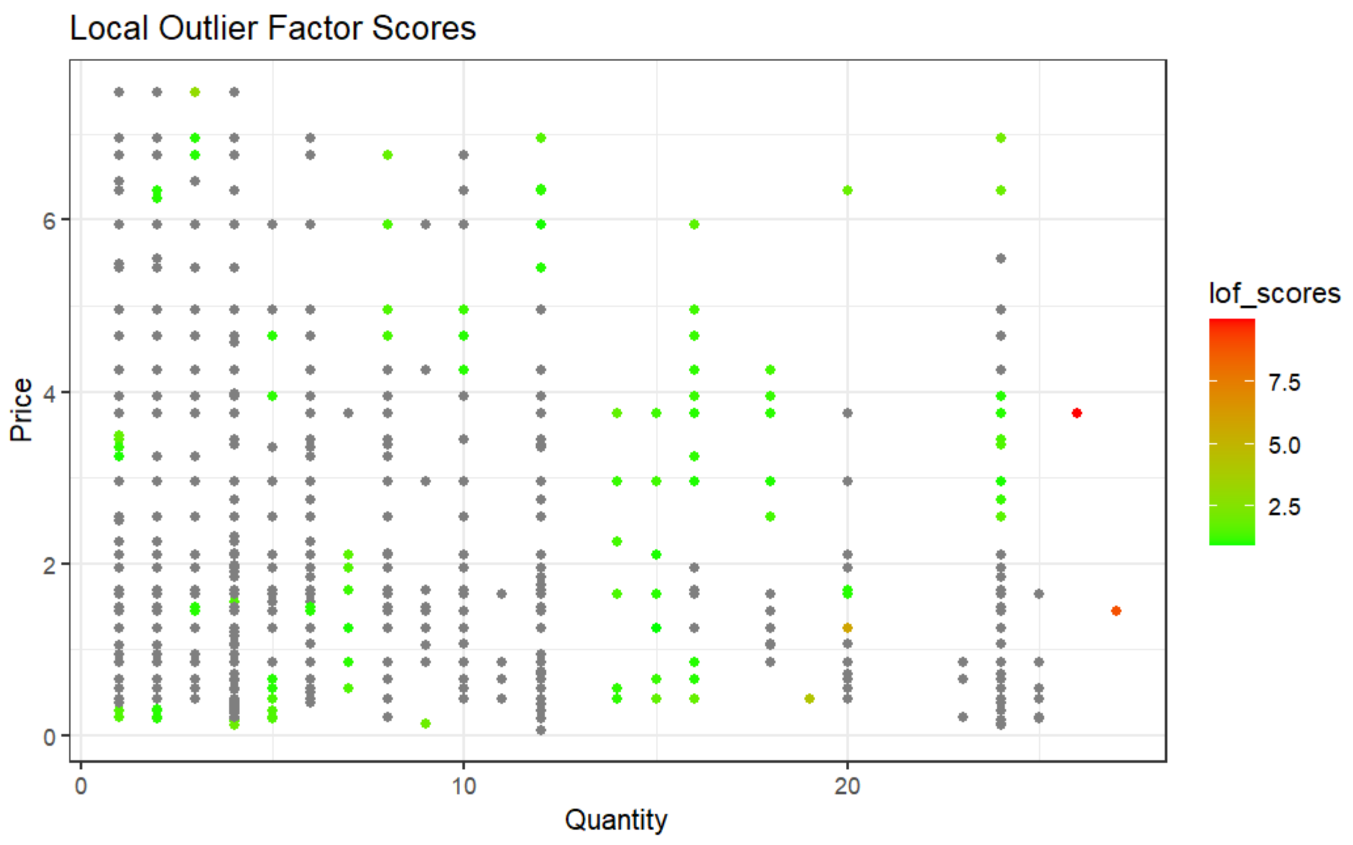
Hide

```
library(DMwR)
```

```
Loading required package: lattice
Loading required package: grid
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

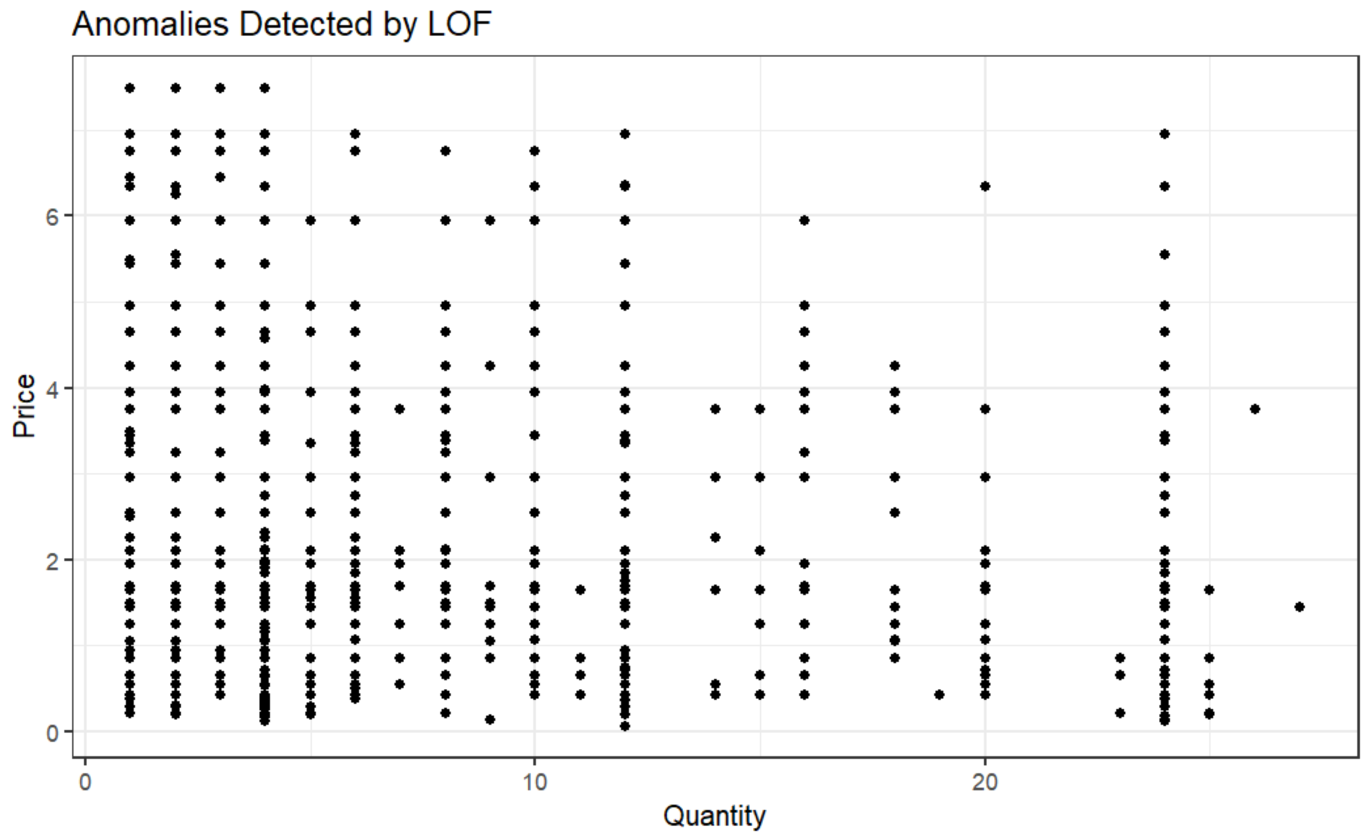
Hide

```r
# Subset the data for relevant columns
anomaly_data <- or_cleaned_no_outliers_filtered %>%
  select(Quantity, Price)

# Create a random sample of the data
set.seed(123)
anomaly_data_sample <- anomaly_data %>% sample_n(5000)

# Compute LOF scores
lof_scores <- lofactor(anomaly_data_sample, k = 5)

# Visualize LOF scores
ggplot(anomaly_data_sample, aes(x = Quantity, y = Price, color = lof_scores)) +
  geom_point() +
  ggtitle("Local Outlier Factor Scores") +
  xlab("Quantity") +
  ylab("Price") +
  scale_color_gradient(low = "green", high = "red") +
  theme_bw()
```



Hide

```
# Identify anomalies
anomalies <- anomaly_data[which(lof_scores > quantile(lof_scores, 0.90, na.rm = TRUE)), ]


# Visualize anomalies
ggplot(anomaly_data_sample, aes(x = Quantity, y = Price)) +
  geom_point() +
  geom_point(data = anomalies, color = "red", size = 4) +
  ggtitle("Anomalies Detected by LOF") +
  xlab("Quantity") +
  ylab("Price") +
  theme_bw()
```

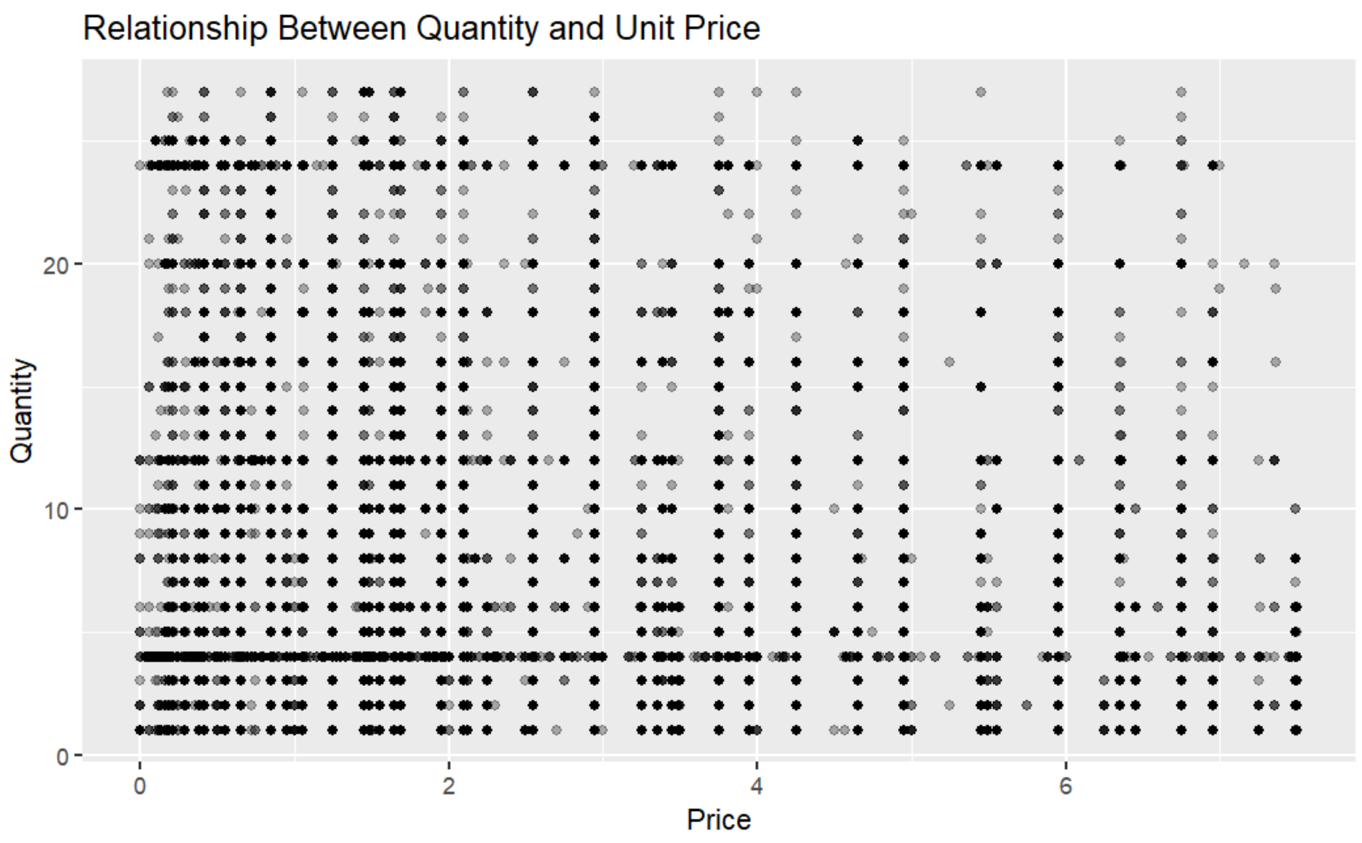


The first plot shows the LOF scores for each data point, with anomalies represented by higher scores (in red) and normal data points represented by lower scores (in green). We can notice that customers buying greater than 20 items of prices around 1.5 USD and 3.75 USD have been marked as anomalies. It may or may not be true as items of such small cost may be trinkets like earrings and small lights that can be purchased in lots of quantities. So this anomaly identification might or might not be true.

The second plot has no red points, it could mean that the LOF algorithm did not detect any outliers in the original dataset. This is highly possible as we eliminated most of the outliers in the beginning.

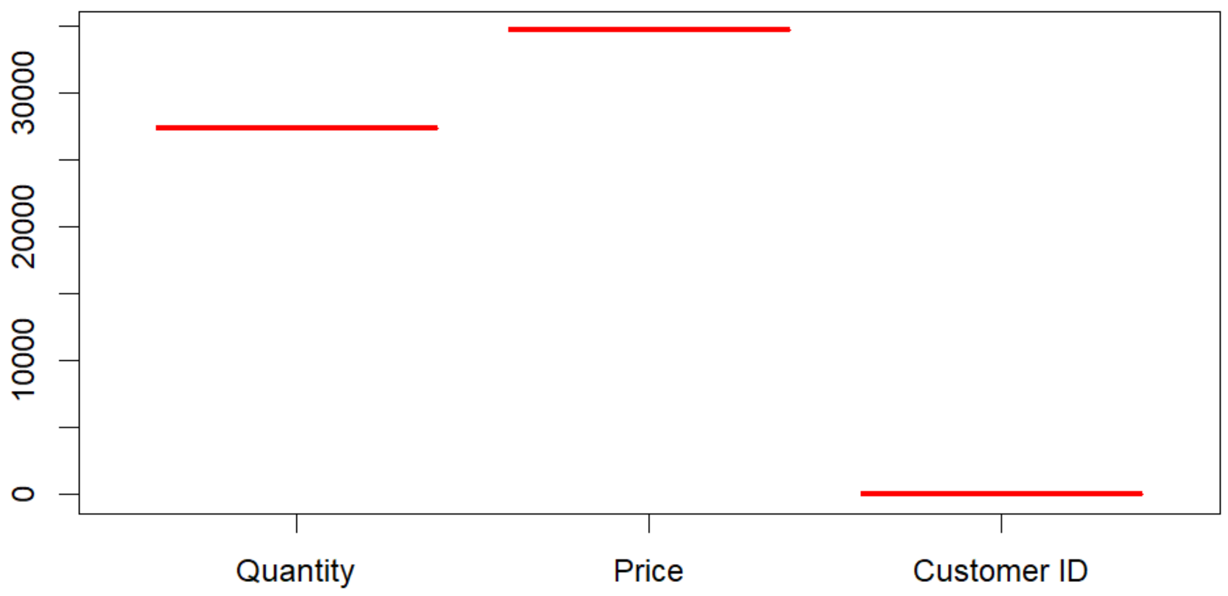
Hide

```
# Create the scatterplot
ggplot(or_cleaned_no_outliers_filtered, aes(x = Price, y = Quantity)) +
  geom_point(alpha = 0.3) +
  labs(title = "Relationship Between Quantity and Unit Price",
       x = "Price",
       y = "Quantity")
```



Hide

```
boxplot(outlier_counts, border="red")
```


0
10000
20000
30000
Quantity
Price
Customer ID