Practical 1

Aim: Learn how to tabulate and summarize marketing data using R.

- **Clean and preprocess the marketing data.**
- **Generate a simple histogram plot to visualize data distribution.**
- **Use tabulation and summary functions to gain insights from the data.**
- **Interpret the findings and discuss the implications for marketing analysis.**

Data Cleaning in R

Data Cleaning in R is the process to transform raw data into consistent data that can be easily analyzed. It is aimed at filtering the content of statistical statements based on the data as well as their reliability. Moreover, it influences the statistical statements based on the data and improves your data quality and overall productivity.

Purpose of Data Cleaning

The following are the various purposes of data cleaning in R:

- Eliminate Errors
- Eliminate Redundancy
- Increase Data Reliability
- Delivery Accuracy
- Ensure Consistency
- Assure Completeness
- Standardize your approach

Overview of a typical data analysis chain

This section represents an overview of typical data analysis. Each rectangle in the figure represents data in a certain state while each arrow represents the activities needed to get from one state to the other. The first state (Raw data) is the data as it comes in. Raw data may lack headers, contain wrong data types, wrong category labels, unknown or unexpected character encoding, and so on. Once this preprocessing has taken place, data can be deemed Technically correct Data. That is, in this state data can be read into an R data. frame, with correct names, types, and labels, without further trouble. However, this does not mean that the values are error-free or complete. Consistent data is the stage where data is ready for statistical inference. It is the data that most statistical theories use as a starting point.

How to clean data in R

Here, this involves various steps, as from the initial raw data have to move toward the consistent and highly efficient data which is ready to be implemented as per the requirements and produces highly precise and accurate statistical results. The steps vary from data to data in this case the user should be aware of the date he/she is using for the results. As there are many characteristics and common symptoms of messy data which totally depend on the data used by the user for analysis.

Characteristics of clean data include data are:

- Free of duplicate rows/values
- Error-free (misspellings free)
- Relevant (special characters free)
- The appropriate data type for analysis
- Free of outliers (or only contain outliers that have been identified/understood)
- Follows a “tidy data” structure

Common symptoms of messy data:

- Special characters (e.g. commas in numeric values)
- Numeric values stored as text/character data types
- Duplicate rows
- Misspellings
- Inaccuracies
- White space
- Missing data
- Zeros instead of null values vary.

Implementation of Data Cleaning in R

For this, we will use inbuilt datasets(air quality datasets) which are available in

R.

Code:

```
head(airquality)
```

Output:

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Handling missing values in R

To handle the missing value we will check the columns of the datasets, if we found some missing data inside the columns then this generates the NA values as an output, which can be not good for every model. So let's check it using mean() methods.

Checking another column

Here we get the mean value of Wind Columns which means it doesn't have any

missing value in this column.

Code:

```
mean(airquality$Wind)
```

Output:

```
[1] 9.957516
>
```

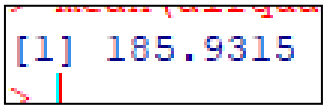
Handling NA values

Handling NA value using na.rm in both columns.

Code:

```
mean(airquality$Solar.R, na.rm = TRUE)
```

Output:



Data Cleaning Operation

After checking the summary of the dataset and we found the number on NA in two columns(Ozone and Solar.R)

Code:

summary(airquality)

Output:

```
     Ozone           Solar.R          Wind             Temp      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
 NA's   :37       NA's   :7                                       
     Month            Day      
 Min.   :5.000   Min.   : 1.0  
 1st Qu.:6.000   1st Qu.: 8.0  
 Median :7.000   Median :16.0  
 Mean   :6.993   Mean   :15.8  
 3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :9.000   Max.   :31.0  
```

tabulate() function in R Language is used to count the frequency of occurrence of an element in the vector. This function checks for each element in the vector and returns the number of times it occurs in the vector. It will create a vector of the length of the maximum element present in the vector.

Syntax: tabulate(x, nbins)

Parameters:

x: Numeric Vector

nbins: to control length of output vector

Example 1:

Code:

```
# R program to count frequency

# of elements in a vector


# Creating a vector

x1  <-  c(3, 5, 3, 7, 9, 4, 6)

x2 <-  c(-1, -4, 2.4, 6, -7)

# Calling tabulate() function

tabulate(x1)

tabulate(x2)
```

Output:

```
> tabulate(x1)
[1] 0 0 2 1 1 1 1 0 1
> tabulate(x2)
[1] 0 1 0 0 0 1
```

R – Histograms

We can create histograms in R Programming Language using the hist() function.

Syntax: hist(v, main, xlab, xlim, ylim, breaks, col, border)

Parameters:

- v: This parameter contains numerical values used in histogram.
- main: This parameter main is the title of the chart.
- col: This parameter is used to set color of the bars.
- xlab: This parameter is the label for horizontal axis.
- border: This parameter is used to set border color of each bar.
- xlim: This parameter is used for plotting values of x-axis.
- ylim: This parameter is used for plotting values of y-axis.
- breaks: This parameter is used as width of each bar.

Creating a simple Histogram in R

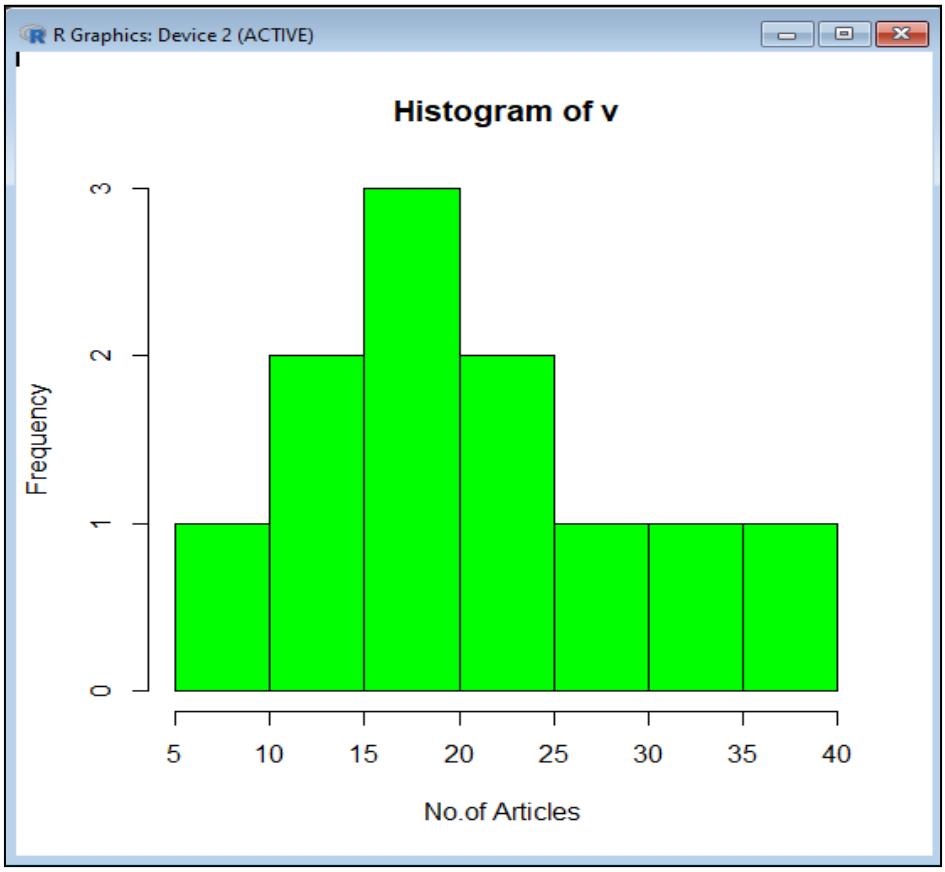
Creating a simple histogram chart by using the above parameter. This vector v is

plot using hist().

Example:

```
# Create data for the graph.
v <- c(19, 23, 11, 5, 16, 21, 32,
       14, 19, 27, 39)
# Create the histogram.
hist(v, xlab = "No.of Articles ",
     col = "green", border = "black")
```

Output:



Interpret the findings and discuss the implications for

Common issues in research analysis (And how to avoid them)

There are several issues common in research analysis. These issues are easily solvable. Below is a brief overview of some of the more general problems you'll encounter in your methods for analyzing data.

Problems

You may not have the skills needed for market data analysis, which results in choosing incorrect data collection methods and drawing biased conclusions about your target audience. You might miss the statistical significance of your market research data and inaccurately subgroup your data.

When conducting research, you might fail to use the best recording method due to undefined outcomes. This will compromise the reliability and validity of any data you collect and make presenting the data that much more challenging.

Solutions

To ensure your market data analysis is conducted effectively, you need a few key resources. These resources include a data analysis procedure and the best methods for analyzing data.

Data analysis procedure

When planning to incorporate research analysis into your business strategy, you need to create a procedure. The plan you create to evaluate any market research data you have collected will determine what actions and strategies you can take for your business's growth.

1. Define the problem

Use a problem statement to identify gaps in your marketing strategy. There are three types of market research for you to consider.

Exploratory: suggests solutions and new ideas

Descriptive: determines the extent of market variable impact

Causal: examines a cause and effect relationship

2. Develop a research plan

After you've defined the problem for your market research data, you're ready to develop a research plan. You'll gather a primary round of data or that which is most relevant to your

problem. Secondary data can provide additional information to support primary data, but may not be necessary, depending on your target.

To collect primary data, you'll use pilot testing, ideally on your target audience. This testing can be done through a variety of tools, including:

Questionnaires or surveys

Mechanical devices

Psychometrics

Qualitative measures

Collecting and analyzing data

There are a multitude of ways to collect data. You can conduct interviews, such as arranged, intercept, or in-depth variations to perform quality data collection. Quantitative data may better suit your needs, though.

Methods for analyzing data

Multiple regression is a statistical method used to explain a dependent variable's value, compared to changing independent variables. One example is comparing revenue to expenses for making said revenue.

Discriminant analysis classifies groups or products into categories to help find effective marketing strategies to act on.

Factor analysis determines connections between larger groups of market research data and identifies the strongest relationships. Factor analysis can help with understanding which things affect your target audience most directly.

Cluster analysis separates products into groups that overlap in distinct areas. When examining clustered research analysis, this will help you divide your customers into subgroups more accurately.

Conjoint analysis examines consumer preferences. It consists of utility functions and the relative importance of product or service relevance to your target audience.

Multidimensional scaling combines multiple quantitative methods to compare competitor brands' products or services.

Interpreting your research analysis

Once you've conducted the appropriate market data analysis, it's time to present yourfindings to create actionable items.

The best way to present big data to external clients is to determine how the research analyses connect to your business strategies. If you struggle to understand relativity, it might be helpful to consult a professional in business smart data, such as Kompass Solutions.

Segmenting and subgroups

Every business is unique and determining how you can best capitalize on potential B2B connections requires a level of understanding potentially beyond your professional scope.

Segmenting your audience into subgroups can provide the opportunity you need to create more impact in select target audiences. The more focused the impact, the more likely you areto align your business goals with your market.

Live data vs. tracked data

Data ages the second it's recorded, but that does not diminish its value for research analysis.It's important to recognize that tracked data provides important contextual information for live data.

Tracked data opens avenues for potential client leads while also making your live data more actionable more immediately. If you know the most important numbers of useful tracked data, you can use this to quickly analyze newer information provided by live data.

You'll be able to recognize patterns over time which might indicate possible new strategies to implement in your business goals. It will also create the opportunity for a high-level view of your data to see what is and is not working.

Practical 2:

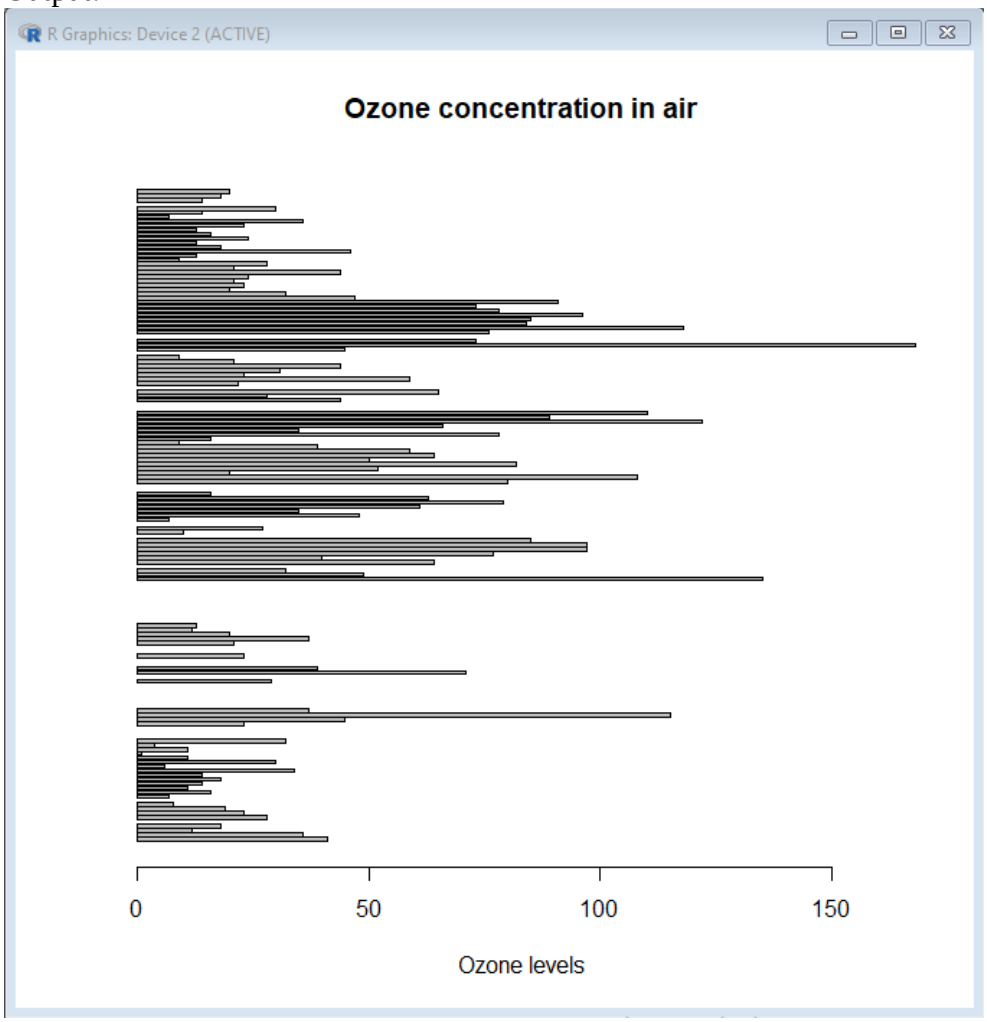
- **Understand the key elements of data visualization.**
- **Create various visualizations such as histograms, scatter plots, line plots, and bar charts using the ggplot() function in R.**
- **Apply appropriate visualization techniques to effectively communicate marketing insights.**

Bar Plot

Example 1:
Code:

```
# Horizontal Bar Plot for
> # Ozone concentration in air
> barplot(airquality$Ozone,
+ main='Ozone concentration in air',
+ xlab='Ozone levels',horiz=TRUE)
```

Output:



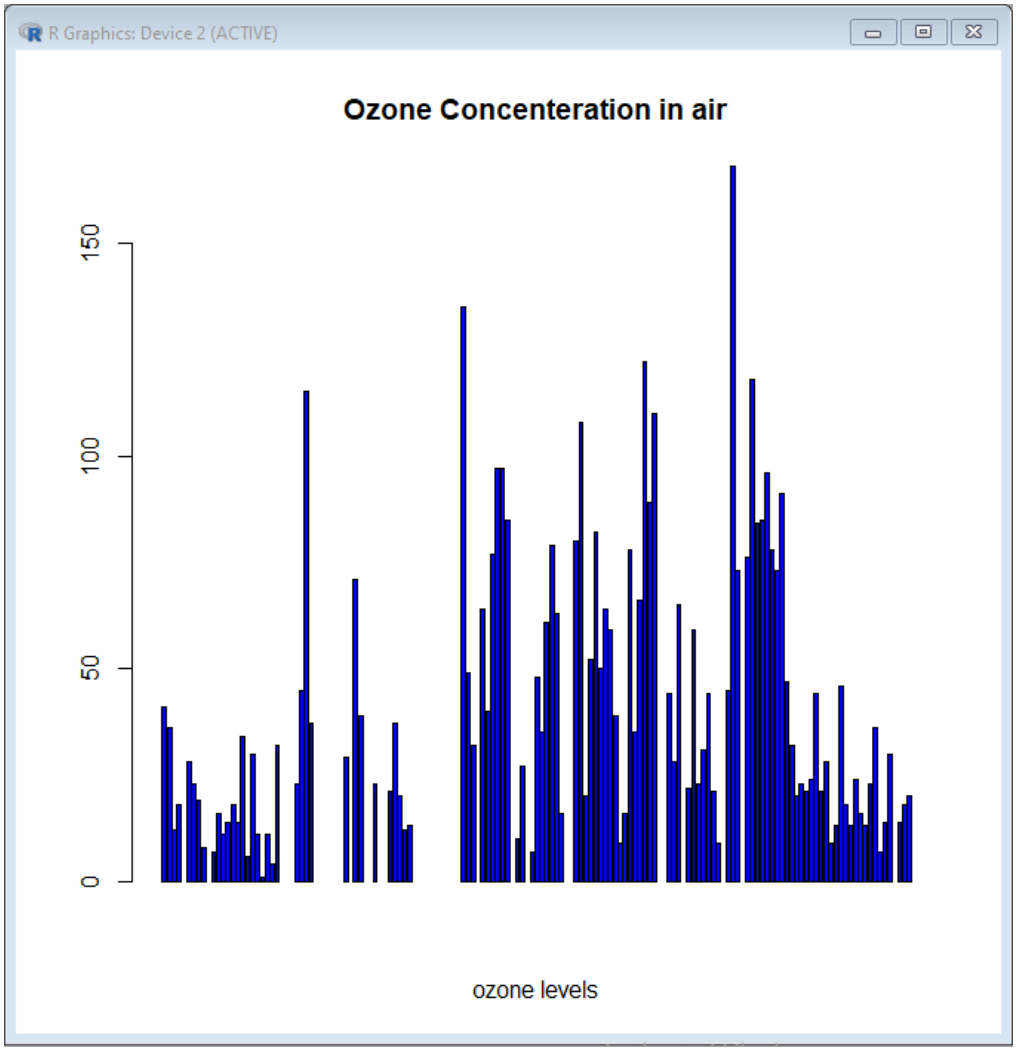
Example 2:
Code:

```
> # Vertical Bar Plot for
```

```
> # Ozone concentration in air
```

```
> barplot(airquality$Ozone, main = 'Ozone Concenteration in air',
+ xlab = 'ozone levels', col ='blue', horiz = FALSE)
```

Output:



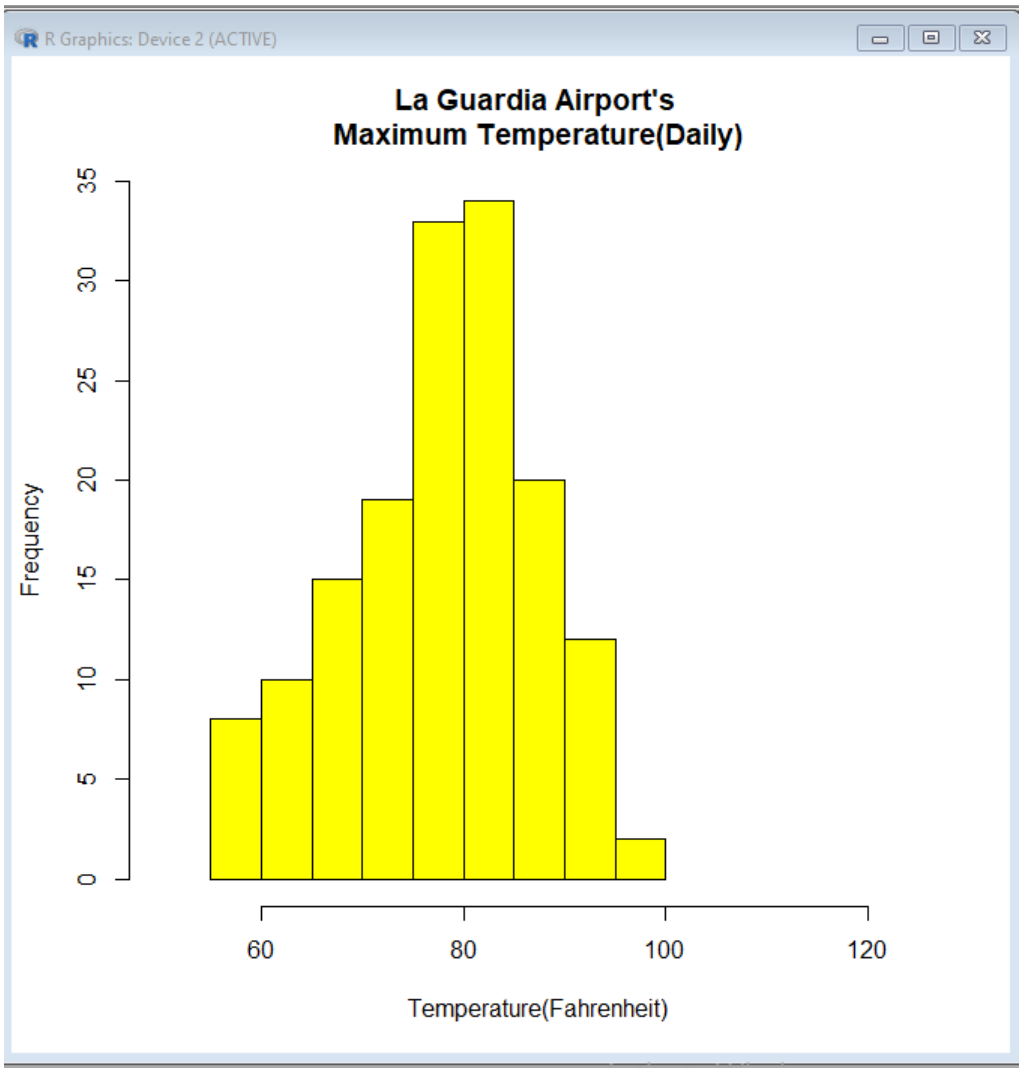
Histogram

Example 1:

Code:

```
> # Histogram for Maximum Daily Temperature
> data(airquality)
>
> hist(airquality$Temp, main ="La Guardia Airport's\
+ Maximum Temperature(Daily)",
+ xlab ="Temperature(Fahrenheit)",
+ xlim = c(50, 125), col ="yellow",
+ freq = TRUE)
```

Output:



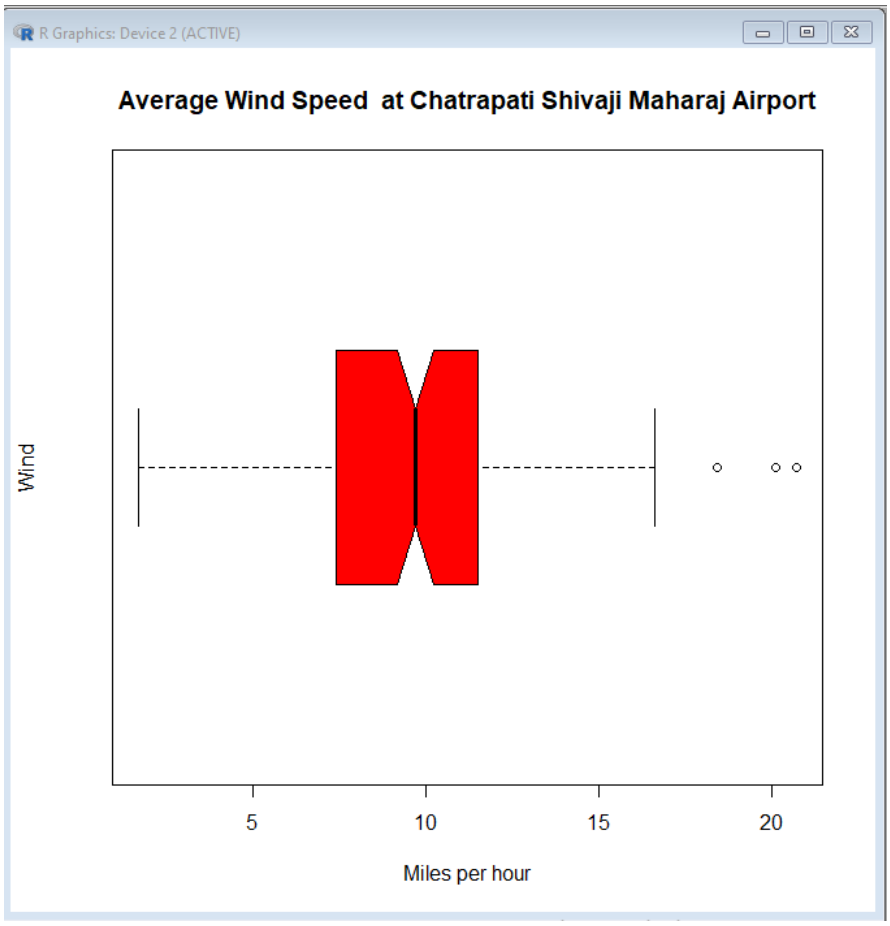
Box Plot

Example 1:

Code:

```
> # Box plot for average wind speed
> data(airquality)
> boxplot(airquality$Wind, main="Average Wind Speed \ at Chatrapati Shivaji Maharaj
Airport",
+ xlab="Miles per hour", ylab="Wind",
+ col="red", border="black",
+ horizontal=TRUE, notch = TRUE)
```

Output:

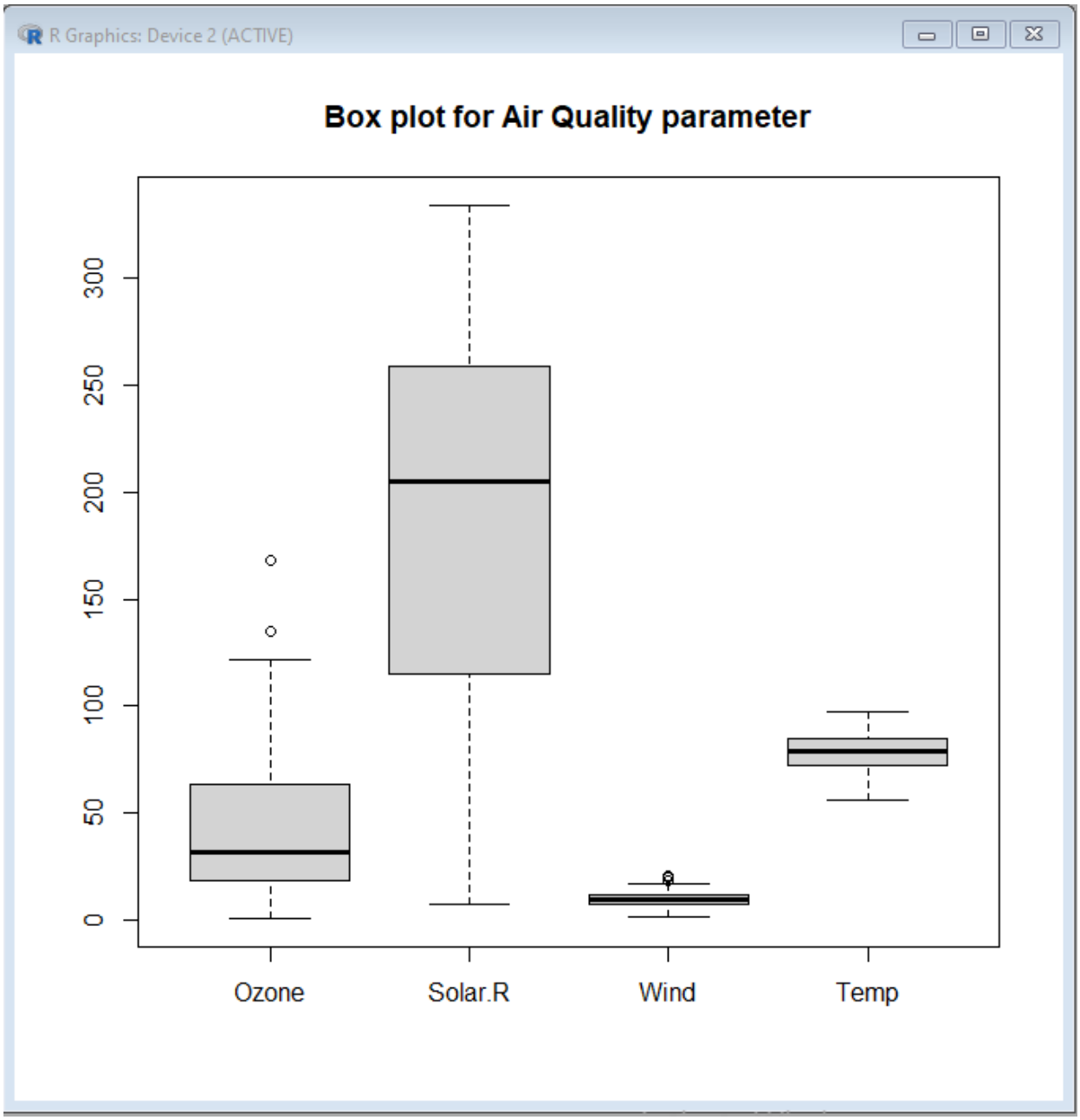


Example 2:

Code:

```
> #Multiple box plots, each representing
> # an Air Quality Parameter
> boxplot(airquality[,0:4],
+ main='Box plot for Air Quality parameter')
```

Output:


R Graphics: Device 2 (ACTIVE)
Box plot for Air Quality parameter
0
50
100
150
200
250
300
Ozone
Solar.R
Wind
Temp

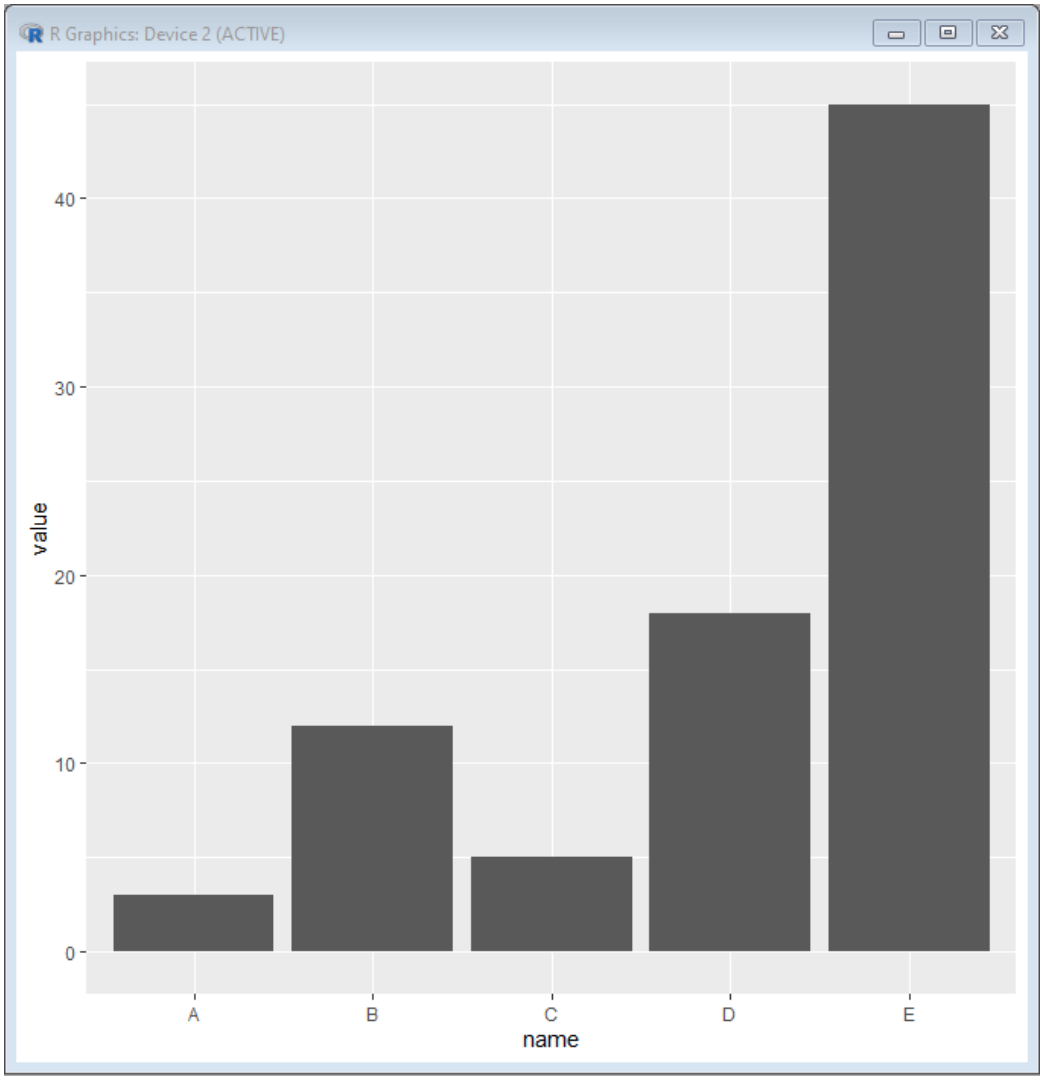
Practical 2.2: Create various visualizations such as histograms, scatter plots, line plots, and bar charts using the ggplot() function in R.

Code:

Example 1: Bar Graph using ggplot2

```
# Loading ggplot2
library(ggplot2)

> # Create data
> data <- data.frame(
+   name=c("A","B","C","D","E") ,
+   value=c(3,12,5,18,45)
+   )

> # Barplot
> ggplot(data, aes(x=name, y=value)) +
+  geom_bar(stat = "identity")
```

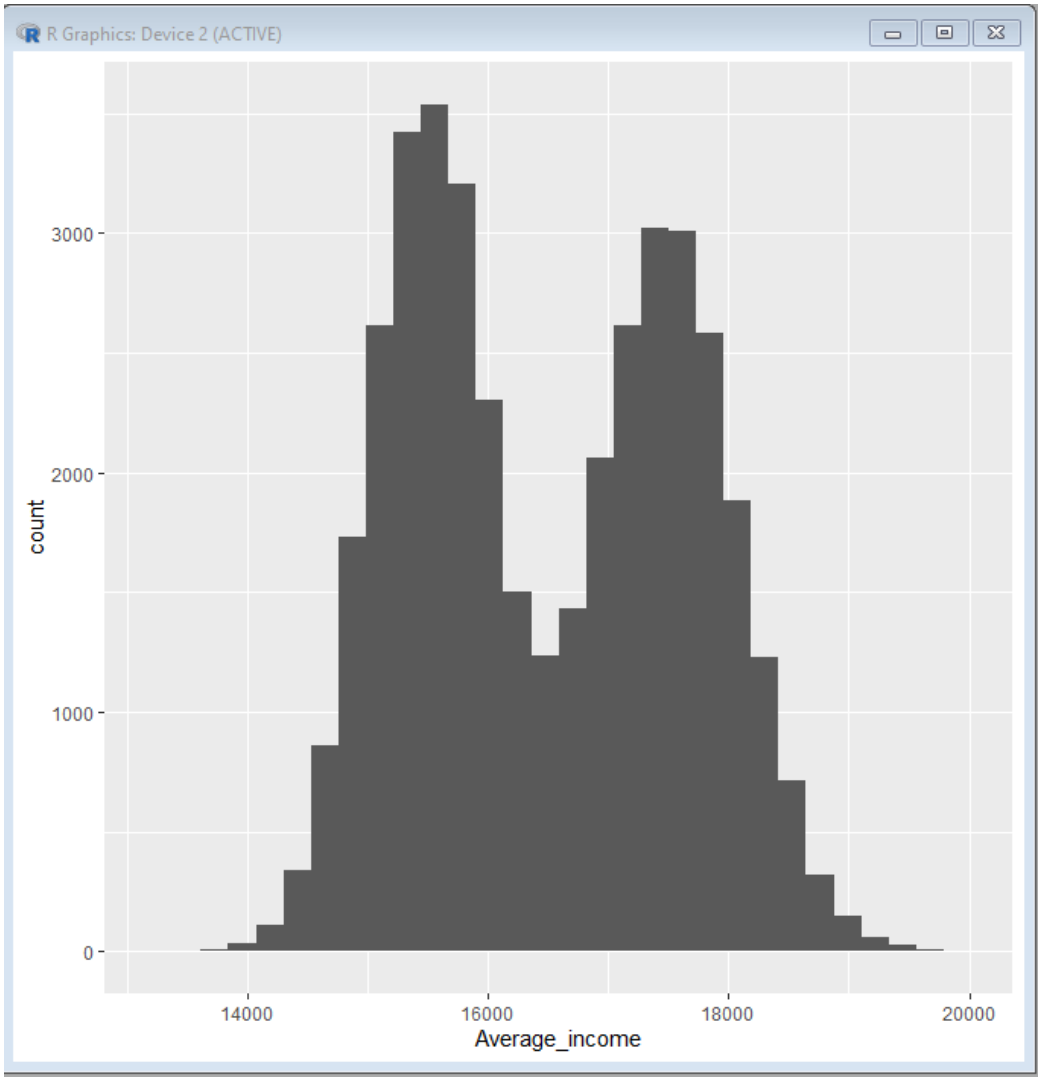
Output:



Example 2: Histogram using ggplot2

```
df <- data.frame(
  gender=factor(rep(c(
    "Average Female income ", "Average Male incmome"), each=20000)),
  Average_income=round(c(rnorm(20000, mean=15500, sd=500),
                         rnorm(20000, mean=17500, sd=600)))
)
head(df)

# if already installed ggplot2 then use library(ggplot2)
library(ggplot2)

# Basic histogram
ggplot(df, aes(x=Average_income)) + geom_histogram()
```

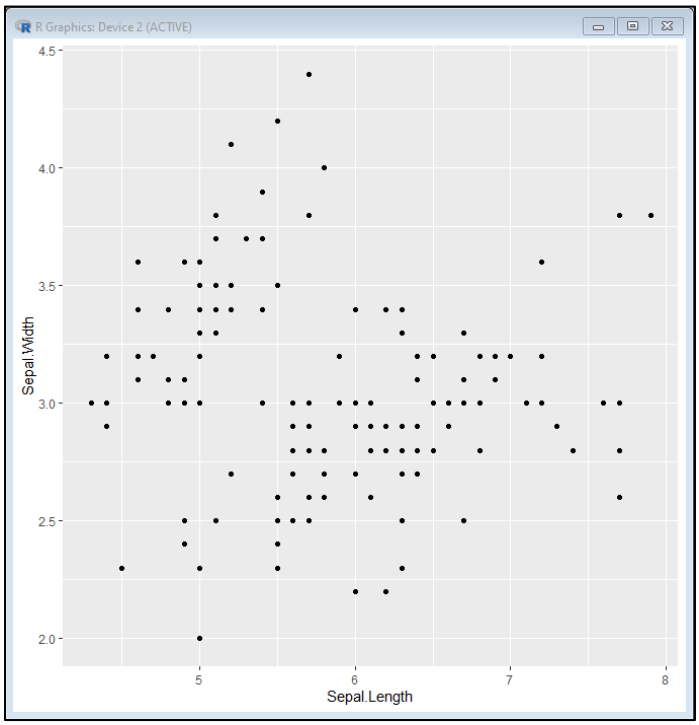
Output:



Example 3: Scater diagram using ggplot2

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_point()
```

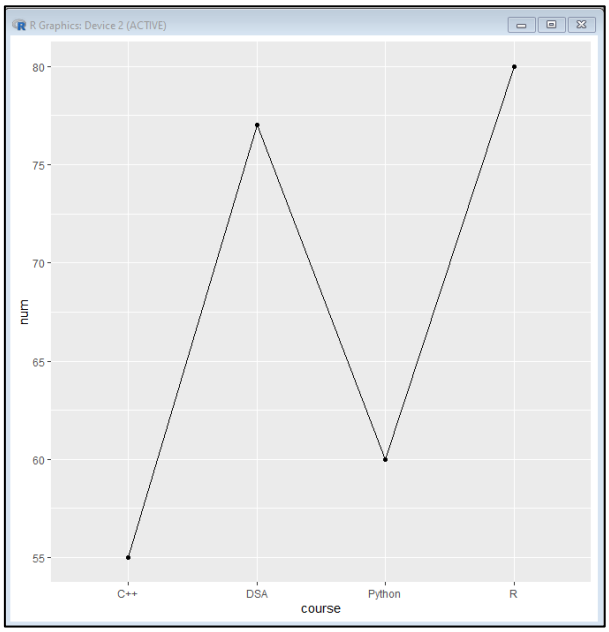
Output:



Example 4: Line plot using ggplot2

```
# Create data for chart
val <-data.frame(course=c('DSA','C++','R','Python'),
                 num=c(77,55,80,60))

# Basic Line
ggplot(data=val, aes(x=course, y=num, group=1)) +
  geom_line()+
  geom_point()
```

Output:



Practical 2.3: Apply appropriate visualization techniques to effectively communicate marketing insights.

Visualizing your marketing data can help you gain insights, communicate results, and optimize your strategy. But how can you create effective and engaging data visualizations that suit your goals and audience? Here are some tips to help you choose the right tools, formats, and techniques for your marketing data.

Effective visualization plays a crucial role in communicating marketing insights. Here are some appropriate visualization techniques you can apply:

1. Bar Charts and Column Charts:
- Use these to compare quantities across different categories.
- Helpful for visualizing sales figures, market share, and other categorical data.
2. Line Charts:
- Effective for showing trends over time.
- Useful for illustrating changes in metrics such as website traffic, social media engagement, or sales performance.
3. Pie Charts:
- Display parts of a whole.
- Useful for representing market segmentation or distribution of resources.
4. Heat Maps:
- Ideal for visualizing patterns and trends in large datasets.
- Useful for analyzing customer behavior, website clicks, or geographical distribution.
5. Scatter Plots:
- Illustrate the relationship between two variables.
- Useful for showing correlations and identifying outliers.
6. Bubble Charts:

- Similar to scatter plots but with an additional dimension represented by the size of the bubble.
- Useful for displaying three dimensions of data in a two-dimensional space.

7. Word Clouds:
- Represent the frequency of words in a dataset.
- Useful for analyzing customer feedback, social media comments, or keyword emphasis.

8. Infographics:
- Combine text, images, and data visualizations for a comprehensive overview.
- Useful for presenting complex marketing strategies or campaigns in a visually appealing manner.

9. Dashboard:
- Bring together multiple visualizations in one comprehensive display.
- Useful for presenting an overview of various marketing metrics and KPIs.

10. Gantt Charts:
- Display project timelines and progress.
- Useful for illustrating the duration and dependencies of marketing campaigns.

11. Treemaps:
- Visualize hierarchical data structures.
- Useful for illustrating the breakdown of marketing budgets or website content structure.

12. Radar Charts:
- Display multivariate data in the form of a two-dimensional chart.
- Useful for comparing marketing performance across multiple dimensions.

When selecting visualization techniques, consider your audience, the message you want to convey, and the specific insights you want to highlight. Additionally, ensure that your visualizations are clear, concise, and aligned with your overall marketing objectives.

Practical 3:

Aim:

- **Learn about experimental design and its application in marketing.**
- **Design experiments using examples from marketing scenarios.**
- **Implement randomization and sample splitting techniques.**
- **Conduct the experiments and collect relevant data for analysis**

Implement randomization and sample splitting techniques.

Method 1: Using base R

The sampling method has the following documentation in R :

Syntax: sample(vec, size, replace = FALSE, prob = NULL)

Code:

```
> # creating the data set
> mat = matrix(
+ # values from 1 to 21
+ c(1:21),
+ # No of rows
+ nrow = 7,
+ # No of columns
+ ncol = 3,
+ byrow = TRUE
+ )
> print ("Dataset")
> print (mat)
```

Output:

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
[6,]   16   17   18
[7,]   19   20   21
```

> # divide the matrix into training set 70% and

```
> # testing 30% respectively with replacement
> sample <- sample(c(TRUE,FALSE), nrow(mat), replace=TRUE, prob=c(0.7,0.3))
> # creating training dataset
> train_dataset <- mat[sample, ]
> # creating testing dataset
> test_dataset <- mat[!sample, ]
> print("Training Dataset")
> print (train_dataset)
> print (test_dataset)
```

Output:

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    7    8    9
[3,]   16   17   18
[4,]   19   20   21
```

```
     [,1] [,2] [,3]
[1,]    4    5    6
[2,]   10   11   12
[3,]   13   14   15
```

Method 2: Using dplyr package in R

Syntax: anti_join(dataset, dataframe, by = col_name)

Code:

```
> # installing the reqd library
> library("dplyr")
> # creating a data frame
> data_frame = data.frame(col1 = c(1:15), col2 = letters[1:15], col3 = c(0,1,1,1,0,0,0,0,
0,1,1,0,1,1,0))
> print(data_frame)
```

Output:

	col1	col2	col3
1	1	a	0
2	2	b	1
3	3	c	1
4	4	d	1
5	5	e	0
6	6	f	0
7	7	g	0
8	8	h	0
9	9	i	0
10	10	j	1
11	11	k	1
12	12	l	0
13	13	m	1
14	14	n	1
15	15	o	0

```
> training_dataset <- data_frame %>% dplyr::sample_frac(0.7)
> print (training_dataset)
```

Output:

	col1	col2	col3
1	8	h	0
2	2	b	1
3	14	n	1
4	3	c	1
5	10	j	1
6	4	d	1
7	15	o	0
8	5	e	0
9	7	g	0
10	6	f	0

```
> testing_dataset <- dplyr::anti_join(data_frame, training_dataset, by = 'col1')
> print (testing_dataset)
```

Output:

	col1	col2	col3
1	1	a	0
2	9	i	0
3	11	k	1
4	12	l	0
5	13	m	1

Method 3: Using catools package in R

Syntax: sample.split(vec , SplitRatio = x)

```
> # installing the reqd library
> library("caTools")
> # creating a data frame
> data_frame = data.frame(col1 = c(1:15), col2 = letters[1:15],
+ col3 = c(0,1,1,1,0,0,0,0,0,1,1,0,1,1,0))
> print(data_frame)
```

Output:

	col1	col2	col3
1	1	a	0
2	2	b	1
3	3	c	1
4	4	d	1
5	5	e	0
6	6	f	0
7	7	g	0
8	8	h	0
9	9	i	0
10	10	j	1
11	11	k	1
12	12	l	0
13	13	m	1
14	14	n	1
15	15	o	0

```
> # creating a sample diving into the ratio of 60:40
> sample <- sample.split(data_frame$col2, SplitRatio = 0.6)
> print ("Training Dataset")
> # check if sample is true
> training_dataset <- subset(data_frame, sample == TRUE)
> print (training_dataset)
```

Output:

	col1	col2	col3
2	2	b	1
5	5	e	0
6	6	f	0
7	7	g	0
9	9	i	0
11	11	k	1
12	12	l	0
13	13	m	1
14	14	n	1

```
> # check if sample holds false
> testing_dataset <- subset(data_frame, sample == FALSE)
> print (testing_dataset)
```

Output:

	col1	col2	col3
1	1	a	0
3	3	c	1
4	4	d	1
8	8	h	0
10	10	j	1
15	15	o	0

Practical no:4

Aim: Understand the concept of hypothesis testing and its role in assessing experiment outcomes

Require("pwr")

```
> require("pwr")
Loading required package: pwr
Warning message:
package 'pwr' was built under R version 4.3.2
```

pwr.anova.test(k=4,f=.25,sig.level=.05,power=.8)

```
> pwr.anova.test(k=4,f=.25,sig.level=.05,power=.8)

     Balanced one-way analysis of variance power calculation

              k = 4
              n = 44.59927
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

pwr.t.test(n=12,d=0.75,sig.level=.05,alternative="greater")

```
> pwr.t.test(n=12,d=0.75,sig.level=.05,alternative="greater")

     Two-sample t test power calculation

              n = 12
              d = 0.75
      sig.level = 0.05
          power = 0.5538378
    alternative = greater

NOTE: n is number in *each* group
```

pwr.2p.test(n=20,sig.level=0.05,power=0.75)

```
    Difference of proportion power calculation for binomial distribution

            h = 0.8331021
            n = 20
    sig.level = 0.05
        power = 0.75
  alternative = two.sided

NOTE: same sample sizes
```

if(!require(lsr)){install.packages("lsr")}

```
> if(!require(lsr)){install.packages("lsr")}
Loading required package: lsr
Installing package into 'C:/Users/LAB PC/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.3/lsr_0.5.2.zip'
Content type 'application/zip' length 209466 bytes (204 KB)
downloaded 204 KB

package 'lsr' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\LAB PC\AppData\Local\Temp\RtmpANWWDs\downloaded_packages
Warning message:
In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return =$
  there is no package called 'lsr'
>
```

binom.test(5, 100, 0.5)

```
> binom.test(5, 100, 0.5)

        Exact binomial test

data:  5 and 100
number of successes = 5, number of trials = 100, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.01643188 0.11283491
sample estimates:
probability of success
                  0.05
```

```
Input =("
+ Classroom  Passed  Failed
+ A     8   2
+ B     3   7
+ ")
>
> Matrix = as.matrix(read.table(textConnection(Input),
+             header=TRUE,
+             row.names=1))
>
> Matrix
```

```
> Input =("
+  Classroom  Passed  Failed
+  A       8   2
+  B       3   7
+ ")
>
> Matrix = as.matrix(read.table(textConnection(Input),
+                header=TRUE,
+                row.names=1))
>
> Matrix
  Passed Failed
A      8      2
B      3      7
```

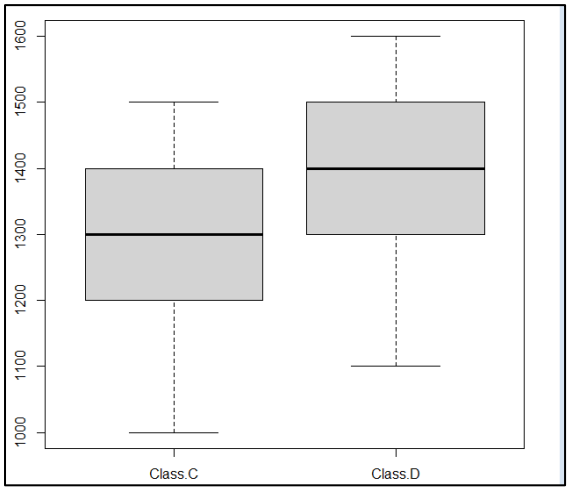
```
> fisher.test(Matrix)

        Fisher's Exact Test for Count Data

data:  Matrix
p-value = 0.06978
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.8821175 127.0558418
sample estimates:
odds ratio
  8.153063
```

```
> Class.A = c(1500, 1505, 1505, 1510, 1510, 1510, 1515, 1515, 1520, 1520)
>
> Class.B = c(1510, 1515, 1515, 1520, 1520, 1520, 1525, 1525, 1530, 1530)
>
> t.test(Class.A, Class.B)

        Welch Two Sample t-test

data:  Class.A and Class.B
t = -3.3968, df = 18, p-value = 0.003214
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -16.184947  -3.815053
sample estimates:
mean of x mean of y
     1511      1521
```

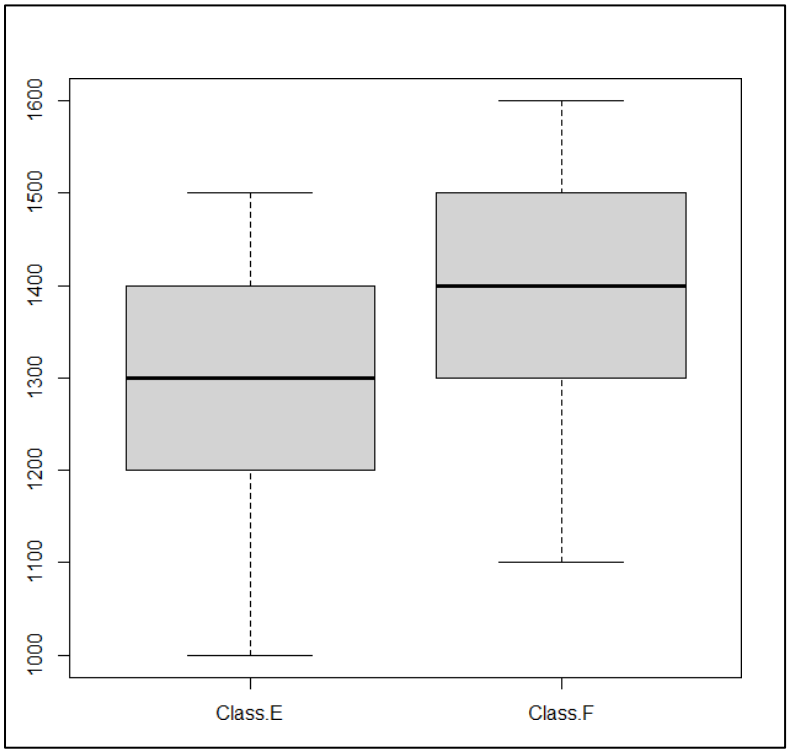
```
> Class.C = c(1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500)
> Class.D = c(1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600)
>
> t.test(Class.C, Class.D)

        Welch Two Sample t-test

data:  Class.C and Class.D
t = -1.4174, df = 18, p-value = 0.1735
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -248.22713   48.22713
sample estimates:
mean of x mean of y
     1290      1390
```


```
> Class.C = c(1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500)
> Class.D = c(1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600)
>
> t.test(Class.C, Class.D)

        Welch Two Sample t-test

data:  Class.C and Class.D
t = -1.4174, df = 18, p-value = 0.1735
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -248.22713   48.22713
sample estimates:
mean of x mean of y
     1290      1390
```


boxplot(cbind(Class.C, Class.D))



```
> Class.E = c(1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500,
+          1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500)
> Class.F = c(1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600,
+          1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600)
>
> t.test(Class.E, Class.F)

        Welch Two Sample t-test

data:  Class.E and Class.F
t = -2.0594, df = 38, p-value = 0.04636
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -198.300852   -1.699148
sample estimates:
mean of x mean of y
     1290      1390
```



```
> install.packages('lsr')
Installing package into 'C:/Users/LAB PC/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.3/lsr_0.5.2.zip'
Content type 'application/zip' length 209466 bytes (204 KB)
downloaded 204 KB

package 'lsr' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\LAB PC\AppData\Local\Temp\RtmpANWWDs\downloaded_packages
>
```

```
> cohensD(Class.C, Class.D,
+     method = "raw")
[1] 0.6681531
>
```

```
>
> Class.G = Class.E / 1600
> Class.H = Class.F / 1600
>
> Class.G
 [1] 0.62500 0.68750 0.75000 0.78125 0.81250 0.81250 0.87500 0.87500 0.90625
[10] 0.93750 0.62500 0.68750 0.75000 0.78125 0.81250 0.81250 0.87500 0.87500
[19] 0.90625 0.93750
> Class.H
 [1] 0.68750 0.75000 0.81250 0.84375 0.87500 0.87500 0.93750 0.93750 0.96875
[10] 1.00000 0.68750 0.75000 0.81250 0.84375 0.87500 0.87500 0.93750 0.93750
[19] 0.96875 1.00000
>
> cohensD(Class.G, Class.H,
+     method="raw")
[1] 0.6681531
>
```

Practical 5

Aim:

Calculate and predict Customer Lifetime Value (CLV).

- **Calculate CLV using different approaches and frameworks.**
- **Explore predictive modeling techniques such as linear regression and logistic regression for CLV prediction.**
- **Assess the accuracy and reliability of CLV predictions.**

1 Prerequisites: Setup the R environment

- **Install the stable version from CRAN:**
  install.packages("CLVTools")

- **Install the development version from GitHub (using the devtools package (Wickham, Hester, and Chang 2019)):**
  install.packages("devtools")
  devtools::install_github("bachmannpatrick/CLVTools", ref = "development")

- **Load the package**
  library("CLVTools")

2 Apply the CLVTools Package

2.1 General workflow

Independent of the latent attrition model applied in CLVTools, the general workflow consists of three main steps:

1. Create a clv.data object containing the dataset and required meta-information such as date formats and column names in the dataset. After initializing the object, there is the option to add additional information on covariates in a separate step.

2. Fit the model on the data provided.

3. Use the estimated model parameters to predict future customer purchase behavior.

2.2 Load sample data provided in the package

Code:

```
data("apparelTrans")
apparelTrans
```

Output:

```
         Id       Date  Price
   1:     1 2005-01-03 230.30
   2:    10 2005-01-03  84.39
   3:    10 2005-02-25 131.07
   4:    10 2005-04-05  86.43
   5:   100 2005-01-03  11.49
  ---
2349:  1221 2006-01-23  26.57
2350:  1221 2006-03-09 129.82
2351:  1221 2006-05-14  14.37
2352:  1222 2005-01-03  44.77
2353:  1222 2005-03-03  99.21
```

2.3 Initialize the CLV-Object

```
clv.apparel <- clvdata(apparelTrans,
                       date.format="ymd",
                       time.unit = "week",
                       estimation.split = 40,
                       name.id = "Id",
                       name.date = "Date",
                       name.price = "Price")
```

2.4 Check the clvdata Object

To get details on the clvdata object, print it to the console.

Code:

```
clv.apparel
```

Output:

```
CLV Transaction Data

Call:
clvdata(data.transactions = apparelTrans, date.format = "ymd",
    time.unit = "week", estimation.split = 40, name.id = "Id",
    name.date = "Date", name.price = "Price")

Total # customers    250
Total # transactions 2257
Spending information TRUE


Time unit         Weeks

Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Holdout start     2005-10-11
Holdout end       2006-07-16
Holdout length    39.71429 Weeks
```

Alternatively the summary() command provides full detailed summary statistics for the provided transactional detail. summary() is available at any step in the process of estimating a probabilistic customer attrition model with CLVTools

Code:

summary(clv.apparel)

Output:

```
CLV Transaction Data

Time unit         Weeks
Estimation length 40.0000 Weeks
Holdout length    39.71429 Weeks

Transaction Data Summary
                                Estimation      Holdout         Total
Number of customers             -               -               250
First Transaction in period     2005-01-03      2005-10-11      2005-01-03
Last Transaction in period      2005-10-10      2006-07-16      2006-07-16
Total # Transactions            1311            946             2257
Mean # Transactions per cust    5.244           8.226           9.028
(SD)                            6.082           8.934           12.603
Mean Spending per Transaction   39.436          38.519          39.051
(SD)                            42.649          59.723          50.503
Total Spending                  51700.490       36438.640       88139.130
Total # zero repeaters          77              -               -
Percentage of zero repeaters    30.800          -               -
Mean Interpurchase time         7.361           5.756           9.462
(SD)                            6.791           6.394           12.266
```

2.5 Estimate Model Parameters

2.5.1 Estimating the model using formula interface:

Code:

```
est.pnbd <- latentAttrition(~pnbd(),
                            data=clv.apparel)
```

Output:

```
Starting estimation...
Estimation finished!
```

est.pnbd

Output:

```
Pareto/NBD Standard Model

Call:
latentAttrition(formula = ~pnbd(), data = clv.apparel)

Coefficients:
      r     alpha         s      beta
 0.7866    5.3349    0.3570   11.6152
KKT1: TRUE
KKT2: TRUE

Used Options:
Correlation:      FALSE
```

Using start parameters and other additional arguments for the optimzier:

```
est.pnbd <- latentAttrition(~pnbd(start.params.model=c(r=1, alpha=10, s=2, beta=8)),
                            optimx.args = list(control=list(trace=5),
                                               method="Nelder-Mead"),
                            data=clv.apparel)
```

Output:

```
Starting estimation...
fn is  fn1
Looking for method =  Nelder-Mead
Methods to be used:[1] "Nelder-Mead"
optcfg:$fname
[1] "fn1"

$npar
[1] 4

$ctrl
$ctrl$follow.on
[1] FALSE

$ctrl$save.failures
[1] TRUE

$ctrl$trace
[1] 5

$ctrl$kkt
[1] TRUE

$ctrl$all.methods
```

When using the formula interface, it is also possible to fit the model without prior specification of of a 'cl]

est.pnbd <- latentAttrition(data(split=40, format=ymd, unit=w)~pnbd(), data=apparelTrans)

Output:

```
Starting estimation...
Estimation finished!
```

2.5.2 Estimating the model using non-formula interface:

est.pnbd <- pnbd(clv.data = clv.apparel)

Starting estimation...

Estimation finished!

est.pnbd

Output:

```
Pareto/NBD Standard Model

Call:
pnbd(clv.data = clv.apparel)

Coefficients:
      r    alpha        s     beta
 0.7866   5.3349   0.3570  11.6152
KKT1: TRUE
KKT2: TRUE

Used Options:
Correlation:     FALSE
```

If we assign starting parameters and additional arguments for the optimizer we use:

Code:

```
est.pnbd <- pnbd(clv.data = clv.apparel,
                 start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                 optimx.args = list(control=list(trace=5),
                                    method="Nelder-Mead"
                                    ))
```

Output:

```
Starting estimation...
fn is  fnl
Looking for method =  Nelder-Mead
Methods to be used:[1] "Nelder-Mead"
optcfg:$fname
[1] "fnl"

$npar
[1] 4

$ctrl
$ctrl$follow.on
[1] FALSE

$ctrl$save.failures
[1] TRUE

$ctrl$trace
[1] 5

$ctrl$kkt
[1] TRUE

$ctrl$all.methods
[1] FALSE
```

#Full detailed summary of the parameter estimates

summary(est.pnbd)

Output:

```
Pareto/NBD Standard  Model

Call:
pnbd(clv.data = clv.apparel, start.params.model = c(r = 1, alpha = 2,
    s = 1, beta = 2), optimx.args = list(control = list(trace = 5),
    method = "Nelder-Mead"))

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
      Estimate Std. Error z-val Pr(>|z|)
r       0.7868     0.1325 5.940 2.85e-09 ***
alpha   5.3336     0.9029 5.907 3.48e-09 ***
s       0.3580     0.1848 1.938   0.0527 .
beta   11.6625    10.7228 1.088   0.2768
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Optimization info:
LL     -2879.4700
AIC     5766.9399
BIC     5781.0258
KKT 1  TRUE
KKT 2  TRUE
fevals 363.0000
Method Nelder-Mead

Used Options:
Correlation FALSE
```

To extract only the coefficients, we can use coef(). To access the confidence intervals for all parameters confint() is available.

#Extract the coefficients only

coef(est.pnbd)

Output:

```
         r      alpha          s       beta
 0.7868019  5.3335761  0.3580460 11.6625048
>
```

#Extract the confidence intervals

confint(est.pnbd)

Output:

```
              2.5 %     97.5 %
r      0.527195704  1.0464080
alpha  3.563873567  7.1032786
s     -0.004116411  0.7202083
beta  -9.353814281 32.6788239
>
```

In order to get the Likelihood value and the corresponding Variance-Covariance Matrix we use the following commands:

 # LogLikelihood at maximum

logLik(est.pnbd)

Output:

```
'log Lik.' -2879.47 (df=4)
>
```

 # Variance-Covariance Matrix at maximum

vcov(est.pnbd)

Output:

```
                  r       alpha             s        beta
r      0.017544210  0.10265517 -0.005674498  -0.5584541
alpha  0.102655173  0.81527545 -0.024147952  -2.4829094
s     -0.005674498 -0.02414795  0.034143690   1.8728996
beta  -0.558454074 -2.48290941  1.872899580 114.9786288
>
```

2.5.3 Estimating the model using formula interface:

```
est.ggomnbd <- latentAttrition(~ggomnbd(start.params.model=
                                        c(r=0.7, alpha=5, b=0.005, s=0.02, beta=0.001)),
                               optimx.args = list(method="Nelder-Mead"),
                               data=clv.apparel)
```

2.5.4 Estimating the model using non-formula interface:

```
est.ggomnbd <- ggomnbd(clv.data = clv.apparel,
                       start.params.model = c(r=0.7, alpha=5, b=0.005, s=0.02,
beta=0.001),
                       optimx.args = list(method="Nelder-Mead"))
```

Output:

```
Starting estimation...
Estimation finished!
```

2.6 Predict Customer Behavior

To use the parameter estimates on new data (e.g., an other customer cohort), the argument newdata optionally allows to provide a new clvdata object.

Code:

```
results <- predict(est.pnbd)
```

Output:

```
Predicting from 2005-10-11 until (incl.) 2006-07-16 (39.86 Weeks).
Estimating gg model to predict spending...
Starting estimation...
Estimation finished!
```

```
print(results)
```

Output:

	Id	period.first	period.last	period.length	actual.x	actual.total.spending	PAlive	CET	DERT	predicted.mean.spending	predicted.CLV
1:	1	2005-10-11	2006-07-16	39.85714	0	0.00	0.3569860	0.2211461	0.05846727	39.95483	2.336050
2:	10	2005-10-11	2006-07-16	39.85714	0	0.00	0.4221442	0.9262527	0.24488554	55.23031	13.525104
3:	100	2005-10-11	2006-07-16	39.85714	23	737.53	0.9154000	13.5402404	3.57981029	43.57390	155.986294
4:	1000	2005-10-11	2006-07-16	39.85714	23	1069.91	0.9967720	13.1742654	3.48305270	41.60921	144.927061
5:	1001	2005-10-11	2006-07-16	39.85714	11	364.00	0.5093342	3.5236788	0.93160102	45.58153	42.463797

246:	1219	2005-10-11	2006-07-16	39.85714	14	413.76	0.9578481	3.6099793	0.95441741	33.58728	32.056283
247:	122	2005-10-11	2006-07-16	39.85714	0	0.00	0.3569860	0.2211461	0.05846727	39.95483	2.336050
248:	1220	2005-10-11	2006-07-16	39.85714	0	0.00	0.3569860	0.2211461	0.05846727	39.95483	2.336050
249:	1221	2005-10-11	2006-07-16	39.85714	9	302.65	0.9433295	4.2979825	1.13631380	34.28958	38.963722
250:	1222	2005-10-11	2006-07-16	39.85714	0	0.00	0.4132389	0.5813533	0.15369998	47.35500	7.278463

To change the duration of the prediction time, we use the predicton.end argument. We can either provide a time period (30 weeks in this example):

Code:

```
predict(est.pnbd, prediction.end = 30)
```

Output:

```
Predicting from 2005-10-11 until (incl.) 2006-05-08 (30 Weeks).
Estimating gg model to predict spending...
Starting estimation...
Estimation finished!
       Id period.first period.last period.length actual.x actual.total.spending    PAlive        CET       DERT predicted.mean.spending predicted.CLV
  1:    1   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
  2:   10   2005-10-11  2006-05-08            30        0                 0.00 0.4221442  0.7135842 0.24488554                55.23031     13.525104
  3:  100   2005-10-11  2006-05-08            30       18               493.47 0.9154000 10.4313879 3.57981029                43.57390    155.986294
  4: 1000   2005-10-11  2006-05-08            30       20              1019.59 0.9967720 10.1494412 3.48305270                41.60921    144.927061
  5: 1001   2005-10-11  2006-05-08            30        8               267.82 0.5093342  2.7146387 0.93160102                45.58153     42.463797
 ---
246: 1219   2005-10-11  2006-05-08            30       14               413.76 0.9578481  2.7811245 0.95441741                33.58728     32.056283
247:  122   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
248: 1220   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
249: 1221   2005-10-11  2006-05-08            30        8               288.28 0.9433295  3.3111615 1.13631380                34.28958     38.963722
250: 1222   2005-10-11  2006-05-08            30        0                 0.00 0.4132389  0.4478740 0.15369998                47.35500      7.278463
```

or provide a date indication the end of the prediction period:

```
predict(est.pnbd, prediction.end = "2006-05-08")
```

```
Predicting from 2005-10-11 until (incl.) 2006-05-08 (30.00 Weeks).
Estimating gg model to predict spending...
Starting estimation...
Estimation finished!
       Id period.first period.last period.length actual.x actual.total.spending    PAlive        CET       DERT predicted.mean.spending predicted.CLV
  1:    1   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
  2:   10   2005-10-11  2006-05-08            30        0                 0.00 0.4221442  0.7135842 0.24488554                55.23031     13.525104
  3:  100   2005-10-11  2006-05-08            30       18               493.47 0.9154000 10.4313879 3.57981029                43.57390    155.986294
  4: 1000   2005-10-11  2006-05-08            30       20              1019.59 0.9967720 10.1494412 3.48305270                41.60921    144.927061
  5: 1001   2005-10-11  2006-05-08            30        8               267.82 0.5093342  2.7146387 0.93160102                45.58153     42.463797
 ---
246: 1219   2005-10-11  2006-05-08            30       14               413.76 0.9578481  2.7811245 0.95441741                33.58728     32.056283
247:  122   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
248: 1220   2005-10-11  2006-05-08            30        0                 0.00 0.3569860  0.1703707 0.05846727                39.95483      2.336050
249: 1221   2005-10-11  2006-05-08            30        8               288.28 0.9433295  3.3111615 1.13631380                34.28958     38.963722
250: 1222   2005-10-11  2006-05-08            30        0                 0.00 0.4132389  0.4478740 0.15369998                47.35500      7.278463
```

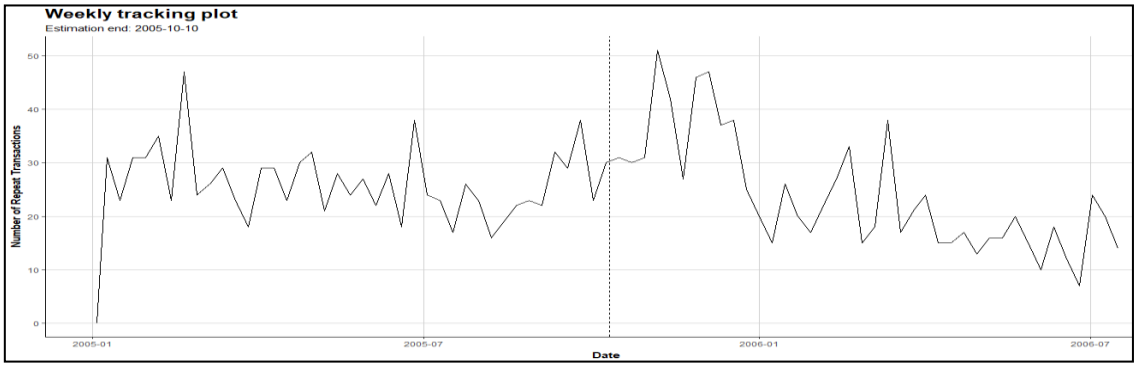
2.7 Plotting

In the following, we have a basic tracking-plot for the aggregated repeat transactions

Code:

```
plot(clv.apparel)
```

Output:

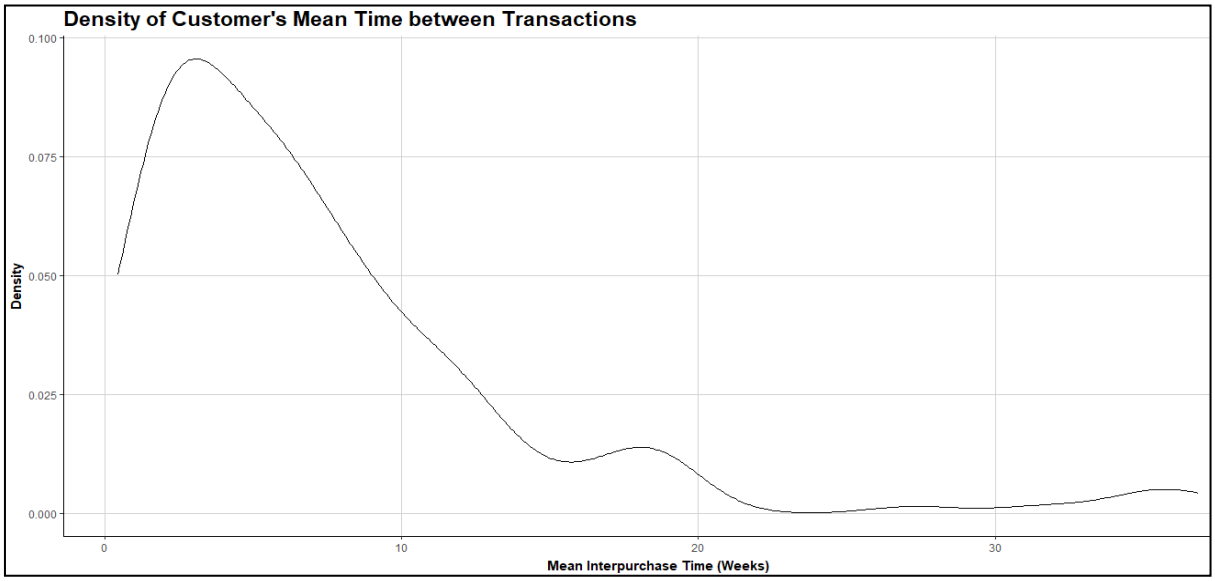


To plot customers mean interpurchase time, we use:

Code:

```
plot(clv.apparel, which="interpurchasetime")
```

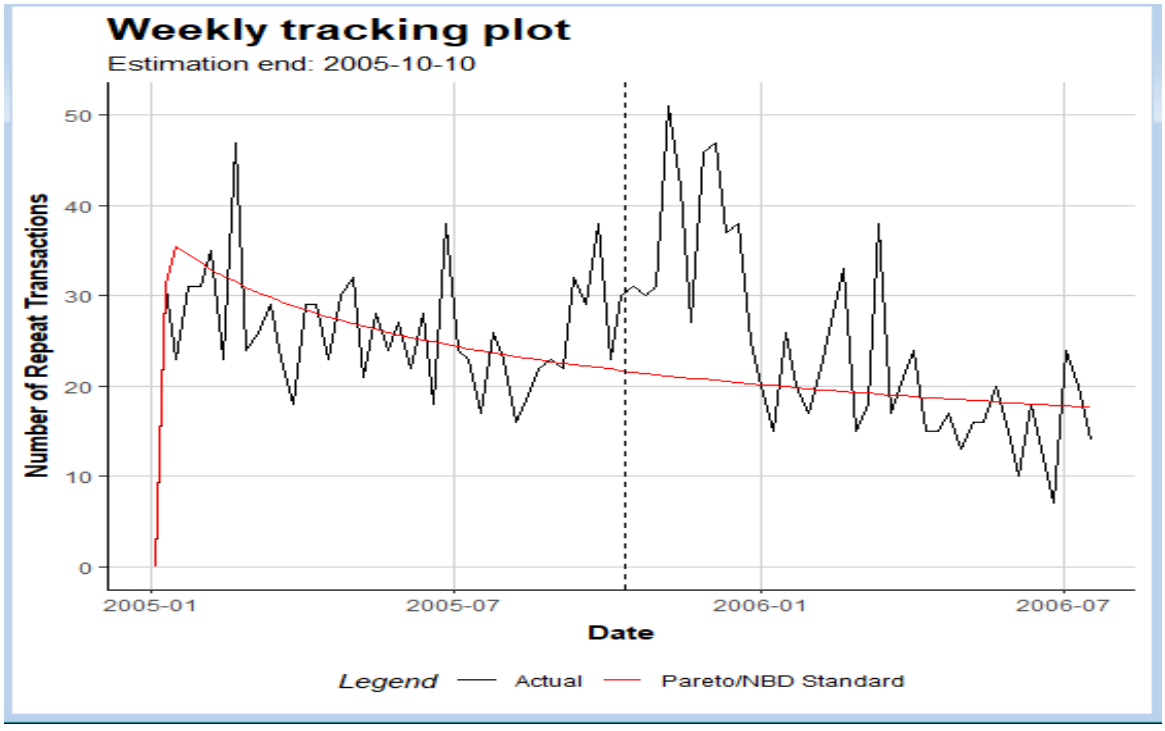
Output:



For a standard tracking plot including the model, we use:

```
plot(est.pnbd)
```

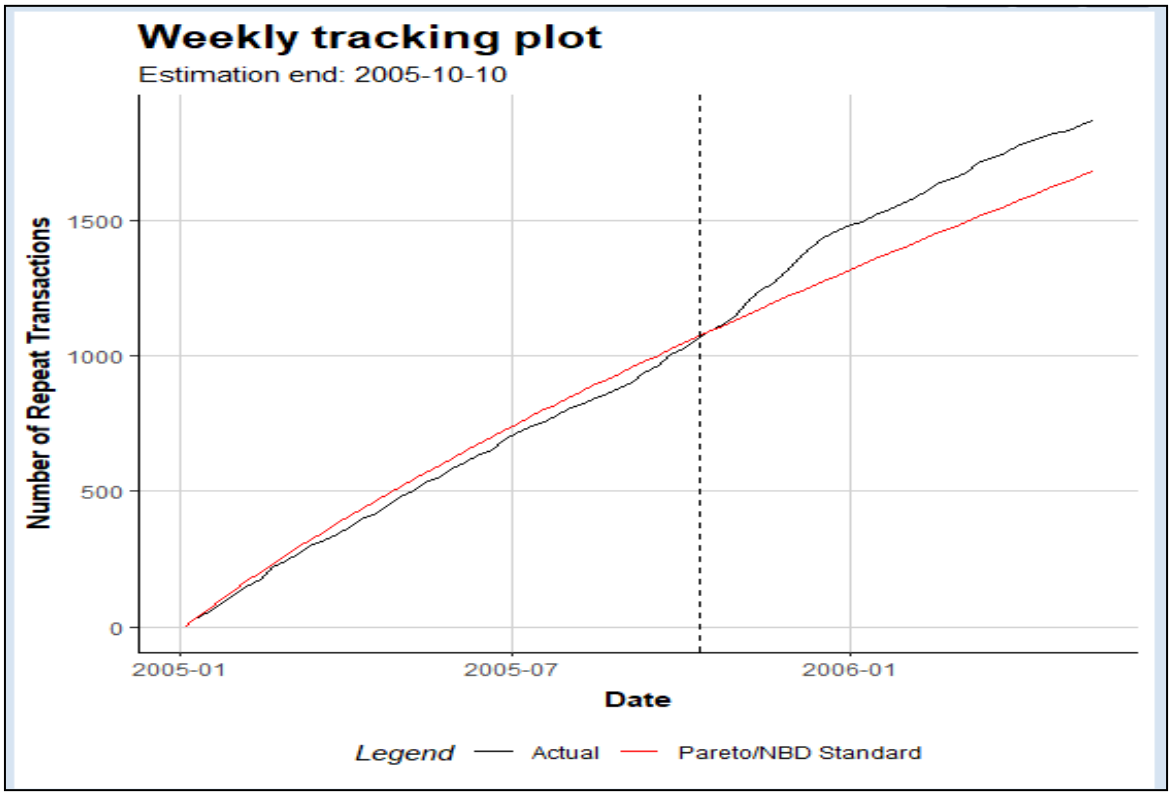
Output:



To plot the cumulative expected transactions 30 time units (30 weeks in this example) ahead of the end of the estimation plot, we use:

plot(est.pnbd, prediction.end = 30, cumulative = TRUE)

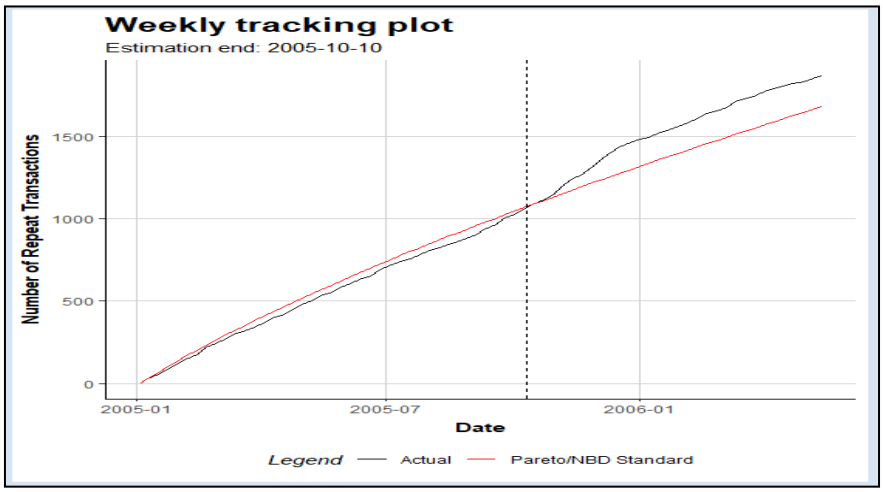
Output:



Alternatively, it is possible to specify a date for the prediction.endargument. Note that dates are rounded to the next full time unit (i.e. week):

plot(est.pnbd, prediction.end = "2006-05-08", cumulative = TRUE)

Output:

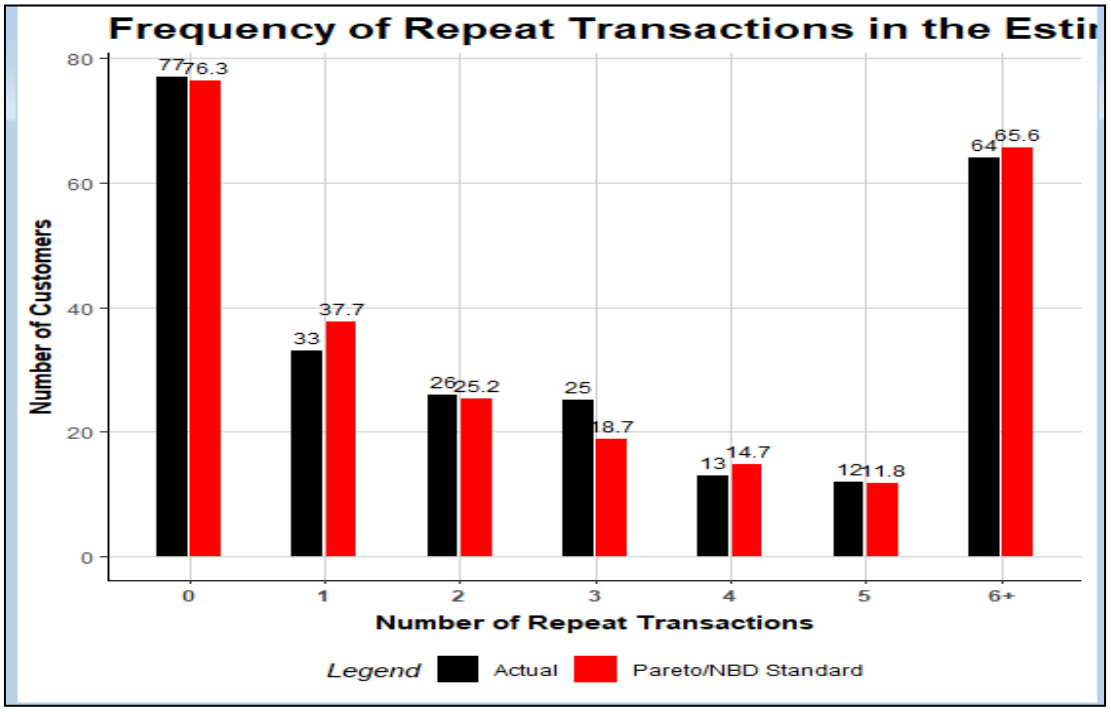


For a plot of the probability mass function (pmf), with 7 bins, we use:

Code:

```
plot(est.pnbd, which="pmf", trans.bins=0:5, label.remaining="6+")
```

Output:



2.8 Covariates

- **Use the data("apparelStaticCov") command to load the time-invariant covariates.**

Code:

```
data("apparelStaticCov")

apparelStaticCov
```

Output:

```
        Id Gender Channel
  1:     1      0       0
  2:    10      0       0
  3:   100      1       0
  4:  1000      1       1
  5:  1001      1       0
 ---
246:  1219      0       1
247:   122      0       0
248:  1220      0       0
249:  1221      1       1
250:  1222      1       0
```

- **Use data("apparelDynCov") command to load**

Code:

```
data("apparelDynCov")

apparelDynCov
```

Output:

```
          Id   Cov.Date Marketing Gender Channel
    1:     1 2004-12-26         1      0       0
    2:     1 2005-01-02         1      0       0
    3:     1 2005-01-09         0      0       0
    4:     1 2005-01-16         1      0       0
    5:     1 2005-01-23         2      0       0
   ---
20496:  1222 2006-06-18         0      1       0
20497:  1222 2006-06-25         0      1       0
20498:  1222 2006-07-02         0      1       0
20499:  1222 2006-07-09         0      1       0
20500:  1222 2006-07-16         0      1       0
```

Accordingly, we specify the time-invariant covariates "Gender" and "Channel" as follows:

```
clv.static<- SetStaticCovariates(clv.data = clv.apparel,
                                 data.cov.life = apparelStaticCov,
                                 data.cov.trans = apparelStaticCov,
                                 names.cov.life = c("Gender", "Channel"),
                                 names.cov.trans =c("Gender", "Channel"),
                                 name.id = "Id")
```

To specify the time-varying contextual factors for seasonal patterns and direct marketing, we use the following:

```
clv.dyn <- SetDynamicCovariates(clv.data = clv.apparel,
                                data.cov.life = apparelDynCov,
                                data.cov.trans = apparelDynCov,
                                names.cov.life = c("Marketing", "Gender", "Channel"),
                                names.cov.trans = c("Marketing", "Gender", "Channel"),
                                name.id = "Id",
                                name.date = "Cov.Date")
```

Output:

```
The Lifetime covariate data before 2005-01-02 (period of estimation start) is cut off.
The Transaction covariate data before 2005-01-02 (period of estimation start) is cut off.
>
```

2.8.1 Estimating the model using formula interface:

We use all present covariates:

est.pnbd.static <- latentAttrition(~pnbd()|.|., clv.static)

Output:

```
Starting estimation...
Estimation finished!
```

Using the formula interface, we can use only selected covariates (only Gender for the lifetime process and both, Channel and Gender for the transaction process):

est.pnbd.static <- latentAttrition(~pnbd()|Gender|Channel+Gender, clv.static)

Or we can transform covariates:

est.pnbd.static <- latentAttrition(~pnbd()|Channel+Gender|I(log(Channel+2)), clv.static)

It is also possible to not initialize a clvdata object for the covariates but instead specify the covariate data directly in the model:

est.pnbd.static <- latentAttrition(data()~pnbd()|.|.,

data=apparelTrans, cov=apparelStaticCov)

Analogously, we can estimate the model containing time-varying covariates. In this example we also activate output of the optimizer in order to observe the progress.

est.pnbd.dyn <- latentAttrition(~pnbd()|.|., optimx.args = list(control=list(trace=5)),

clv.dyn)

```
Creating walks...
Walks created.
Generating model start parameters by fitting a no covariate pnbd model...
Optimization of no covariate model found model start parameters (r= 0.7866, alpha= 5.3349, s= 0.3570, beta=11.6152)
Starting estimation...
fn is  fn1
Looking for method =  Nelder-Mead
Methods to be used:[1] "Nelder-Mead"
optcfg:$fname
[1] "fn1"

$npar
[1] 10

$ctrl
$ctrl$follow.on
[1] FALSE

$ctrl$save.failures
[1] TRUE

$ctrl$trace
[1] 5

$ctrl$kkt
[1] TRUE

$ctrl$all.methods
[1] FALSE

$ctrl$starttests
[1] FALSE

$ctrl$maximize
[1] FALSE

$ctrl$dowarn
[1] TRUE
```

2.8.2 Estimating the model using non-formula interface:

Code:

```
est.pnbd.static <- pnbd(clv.static,
                        start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                        start.params.life = c(Gender=0.6, Channel=0.4),
                        start.params.trans = c(Gender=0.6, Channel=0.4))
```

Output:

```
Starting estimation...
Estimation finished!
```

It is not possible to alter or select covariates in the non-formula interface, but, we can also estimate a model containing time-varying covariates:

Code:

```
est.pnbd.dyn <- pnbd(clv.dyn,
                     start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                     start.params.life = c(Marketing=0.5, Gender=0.6, Channel=0.4),
                     start.params.trans = c(Marketing=0.5, Gender=0.6, Channel=0.4),
                     optimx.args = list(control=list(trace=5)))
```

Output:

```
Creating walks...
Walks created.
Starting estimation...
fn is  fn1
Looking for method =  Nelder-Mead
Methods to be used:[1] "Nelder-Mead"
optcfg:$fname
[1] "fn1"

$npar
[1] 10

$ctrl
$ctrl$follow.on
[1] FALSE

$ctrl$save.failures
[1] TRUE

$ctrl$trace
[1] 5

$ctrl$kkt
[1] TRUE
```

If this is the case it is usually a good idea to rerun the estimation using alternative starting values.

Code:

summary(est.pnbd.static)

Output:

```
Pareto/NBD with Static Covariates  Model

Call:
pnbd(clv.data = clv.static, start.params.model = c(r = 1, alpha = 2,
    s = 1, beta = 2), start.params.life = c(Gender = 0.6, Channel = 0.4),
    start.params.trans = c(Gender = 0.6, Channel = 0.4))

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
              Estimate Std. Error  z-val Pr(>|z|)
r              1.41800    0.27733  5.113 3.17e-07 ***
alpha         35.62069    8.58072  4.151 3.31e-05 ***
s              0.27258    0.09512  2.866  0.00416 **
beta           8.63265   11.26299  0.766  0.44340
life.Gender    1.53314    1.09655  1.398  0.16207
life.Channel  -1.70528    0.66153 -2.578  0.00994 **
trans.Gender   1.42366    0.19764  7.203 5.88e-13 ***
trans.Channel  0.40225    0.15123  2.660  0.00782 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Optimization info:
LL      -2846.1677
AIC     5708.3355
BIC     5736.5071
KKT 1   TRUE
KKT 2   TRUE
fevals 52.0000
Method L-BFGS-B

Used Options:
Correlation     FALSE
Regularization  FALSE
```

2.9 Add Correlation to the model

2.9.1 Estimating the model using formula interface:

```
est.pnbd.cor <- latentAttrition(~pnbd(use.cor=TRUE),
                    data=clv.apparel)
```

Output:

```
Starting estimation...
Estimation finished!
```

2.9.2 Estimating the model using non-formula interface:

Code:

```
est.pnbd.cor <- pnbd(clv.apparel,
                     use.cor= TRUE)
summary(est.pnbd.cor)
```

Output:

```
Starting estimation...
Estimation finished!
> summary(est.pnbd.cor)
Pareto/NBD Standard  Model

Call:
pnbd(clv.data = clv.apparel, use.cor = TRUE)

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
                 Estimate Std. Error  z-val Pr(>|z|)
r                0.790394   0.212211  3.725 0.000196 ***
alpha            5.355977   1.168532  4.584 4.57e-06 ***
s                0.278047   0.183852  1.512 0.130448
beta             7.116957   6.229374  1.142 0.253253
Cor(life,trans) -0.009087   0.338442 -0.027 0.978581
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Optimization info:
LL      -2879.6615
AIC     5769.3229
BIC     5786.9302
KKT 1   TRUE
KKT 2   TRUE
fevals 746.0000
Method Nelder-Mead

Used Options:
Correlation TRUE
```

2.10 Advanced Options for Covariates

2.10.1 Estimating the model using formula interface:

Code:

```
est.pnbd.reg <- latentAttrition(~pnbd()|.|.|regularization(life=3, trans=8), clv.static)
summary(est.pnbd.reg)
```

Output:

```
Pareto/NBD with Static Covariates  Model

Call:
latentAttrition(formula = ~pnbd() | . | . | regularization(life = 3,
    trans = 8), data = clv.static)

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
                Estimate Std. Error  z-val Pr(>|z|)
r              7.914e-01  2.115e+00  0.374    0.708
alpha          5.412e+00  1.465e+01  0.369    0.712
s              3.565e-01  2.887e+00  0.124    0.902
beta           1.152e+01  1.666e+02  0.069    0.945
life.Gender   -8.207e-04  4.080e-01 -0.002    0.998
life.Channel  -3.969e-03  4.078e-01 -0.010    0.992
trans.Gender   6.670e-03  2.498e-01  0.027    0.979
trans.Channel  5.505e-03  2.490e-01  0.022    0.982

Optimization info:
LL      -11.5172
AIC     39.0345
BIC     67.2061
KKT 1   TRUE
KKT 2   TRUE
fevals 70.0000
Method L-BFGS-B

Used Options:
Correlation      FALSE
Regularization   TRUE
   lambda.life   3.0000
   lambda.trans 8.0000
Constraint covs FALSE
```

2.10.2 Estimating the model using non-formula interface:

Code:

```
est.pnbd.reg <- pnbd(clv.static,
                     start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                     reg.lambdas = c(trans=100, life=100))
summary(est.pnbd.reg)
```

Output:

```
Pareto/NBD with Static Covariates  Model

Call:
pnbd(clv.data = clv.static, start.params.model = c(r = 1, alpha = 2,
    s = 1, beta = 2), reg.lambdas = c(trans = 100, life = 100))

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
                Estimate Std. Error  z-val Pr(>|z|)
r              7.834e-01  2.078e+00  0.377    0.706
alpha          5.323e+00  1.423e+01  0.374    0.708
s              3.664e-01  3.048e+00  0.120    0.904
beta           1.223e+01  1.791e+02  0.068    0.946
life.Gender   -2.122e-05  1.791e-02 -0.001    0.999
life.Channel  -1.198e-04  1.791e-02 -0.007    0.995
trans.Gender   5.321e-04  1.791e-02  0.030    0.976
trans.Channel  4.449e-04  1.791e-02  0.025    0.980

Optimization info:
LL      -11.5178
AIC     39.0357
BIC     67.2074
KKT 1   TRUE
KKT 2   TRUE
fevals 155.0000
Method L-BFGS-B

Used Options:
Correlation      FALSE
Regularization   TRUE
   lambda.life   100.0000
   lambda.trans 100.0000
Constraint covs FALSE
```

2.10.3 Estimating the model using formula interface:

Code:

```
est.pnbd.constr <- latentAttrition(~pnbd(names.cov.constr=c("Gender"),
                                         start.params.constr = c(Gender = 0.6))|.|.,
                                   clv.static)
summary(est.pnbd.constr)
```

Output:

```
Pareto/NBD with Static Covariates  Model

Call:
latentAttrition(formula = ~pnbd(names.cov.constr = c("Gender"),
    start.params.constr = c(Gender = 0.6)) | . | ., data = clv.static)

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
               Estimate Std. Error  z-val Pr(>|z|)
r               1.42358    0.27542  5.169 2.36e-07 ***
alpha          35.46988    8.38196  4.232 2.32e-05 ***
s               0.27401    0.09524  2.877  0.00402 **
beta            7.86722    7.66575  1.026  0.30476
life.Channel   -1.68963    0.64255 -2.630  0.00855 **
trans.Channel   0.40203    0.15102  2.662  0.00776 **
constr.Gender   1.41588    0.18404  7.693 1.42e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Optimization info:
LL      -2846.1729
AIC     5706.3458
BIC     5730.9960
KKT 1   TRUE
KKT 2   TRUE
fevals 50.0000
Method L-BFGS-B

Used Options:
Correlation           FALSE
Regularization        FALSE
Constraint covs       TRUE
   Constraint params Gender
```

2.10.4 Estimating the model using non-formula interface:

Code:

```
est.pnbd.constr <- pnbd(clv.static,
                        start.params.model = c(r=1, alpha = 2, s = 1, beta = 2),
                        start.params.constr = c(Gender=0.6),
                        names.cov.constr=c("Gender"))
summary(est.pnbd.constr)
```

Output:

```
Pareto/NBD with Static Covariates  Model

Call:
pnbd(clv.data = clv.static, start.params.model = c(r = 1, alpha = 2,
    s = 1, beta = 2), names.cov.constr = c("Gender"), start.params.constr = c(Gender = 0.6))

Fitting period:
Estimation start  2005-01-03
Estimation end    2005-10-10
Estimation length 40.0000 Weeks

Coefficients:
              Estimate Std. Error  z-val Pr(>|z|)
r              1.42491    0.27552  5.172 2.32e-07 ***
alpha         35.43796    8.37521  4.231 2.32e-05 ***
s              0.27249    0.09451  2.883  0.00393 **
beta           7.72441    7.53594  1.025  0.30536
life.Channel  -1.69866    0.64451 -2.636  0.00840 **
trans.Channel  0.40107    0.15098  2.656  0.00790 **
constr.Gender  1.41494    0.18404  7.688 1.49e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Optimization info:
LL      -2846.1731
AIC     5706.3462
BIC     5730.9964
KKT 1   TRUE
KKT 2   TRUE
fevals 28.0000
Method L-BFGS-B

Used Options:
Correlation          FALSE
Regularization       FALSE
Constraint covs      TRUE
   Constraint params Gender
```

Practical 6

Aim:

Apply CLV analysis and cohort analysis in marketing analytics.

- **Analyze CLV data and identify patterns and trends.**
- **Perform cohort analysis to segment customers based on their behavior or characteristics.**
- **Interpret the results of CLV analysis and cohort analysis to derive actionable insights for marketing strategies.**

3 Customer Spending

3.1 Load sample data provided in the package

Code:

```
data("apparelTrans")

apparelTrans
```

Output:

```
         Id       Date  Price
   1:     1 2005-01-03 230.30
   2:    10 2005-01-03  84.39
   3:    10 2005-02-25 131.07
   4:    10 2005-04-05  86.43
   5:   100 2005-01-03  11.49
  ---
2349:  1221 2006-01-23  26.57
2350:  1221 2006-03-09 129.82
2351:  1221 2006-05-14  14.37
2352:  1222 2005-01-03  44.77
2353:  1222 2005-03-03  99.21
```

```
clv.apparel <- clvdata(apparelTrans,
                       date.format="ymd",
                       time.unit = "week",
                       estimation.split = 40,
                       name.id = "Id",
                       name.date = "Date",
                       name.price = "Price")
```

3.2 Estimate Model Parameters

3.2.1 Estimating the model using formula interface:

Code:

```
est.gg <- spending(~gg(),
                   data=clv.apparel)
est.gg
```

Output:

```
Gamma-Gamma Model

Call:
spending(formula = ~gg(), data = clv.apparel)

Coefficients:
      p         q     gamma
  2.305    17.148   279.974
KKT1: TRUE
KKT2: TRUE
```

Using start parameters or other additional arguments for the optimzier:

Code:

```
est.gg <- spending(~gg(start.params.model=c(p=0.5, q=15, gamma=2)),
                                optimx.args = list(control=list(trace=5)),
                   data=clv.apparel)
```

Output:

```
Starting estimation...
fn is  fn1
Looking for method =  L-BFGS-B
Methods to be used:[1] "L-BFGS-B"
optcfg:$fname
[1] "fn1"

$npar
[1] 3

$ctrl
$ctrl$follow.on
[1] FALSE

$ctrl$save.failures
[1] TRUE

$ctrl$trace
[1] 5

$ctrl$kkt
[1] TRUE

$ctrl$all.methods
[1] FALSE

$ctrl$starttests
[1] FALSE

$ctrl$maximize
[1] FALSE

$ctrl$dowarn
[1] TRUE

$ctrl$usenumDeriv
[1] FALSE
```

Specify the option to NOT remove the first transaction:

```
est.gg <- spending(~gg(remove.first.transaction=FALSE),
                   data=clv.apparel)
```

When using the formula interface, it is also possible to fit the model without prior specification of of a clvdata object:

```
est.gg <- spending(data(split=40, format=ymd, unit=w)~gg(),
                   data=apparelTrans)
```

3.2.2 Estimating the model using non-formula interface:

Code:

```
est.gg<- gg(clv.data = clv.apparel)
est.gg
```

Output:

```
Gamma-Gamma Model

Call:
gg(clv.data = clv.apparel)

Coefficients:
      p         q     gamma
  2.305    17.148   279.974
KKT1: TRUE
KKT2: TRUE
```

Using start parameters and other additional arguments for the optimzier:

Code:

```
est.gg<- gg(start.params.model=c(p=0.5, q=15, gamma=2), clv.data = clv.apparel)
est.gg
```

Output:

```
Gamma-Gamma Model

Call:
gg(clv.data = clv.apparel, start.params.model = c(p = 0.5, q = 15,
    gamma = 2))

Coefficients:
      p         q     gamma
  2.305    17.148   279.976
KKT1: TRUE
KKT2: TRUE
```

Specify the option to NOT remove the first transaction:

Code:

```
est.gg<- gg(clv.data = clv.apparel, remove.first.transaction=FALSE)

est.gg
```

Output:

```
Gamma-Gamma Model

Call:
gg(clv.data = clv.apparel, remove.first.transaction = FALSE)

Coefficients:
     p        q    gamma
  2.47    15.45   227.09
KKT1: TRUE
KKT2: TRUE
```

3.3 Predict Customer Spending

Code:

```
results.spending <- predict(est.gg)

print(results.spending)
```

Output:

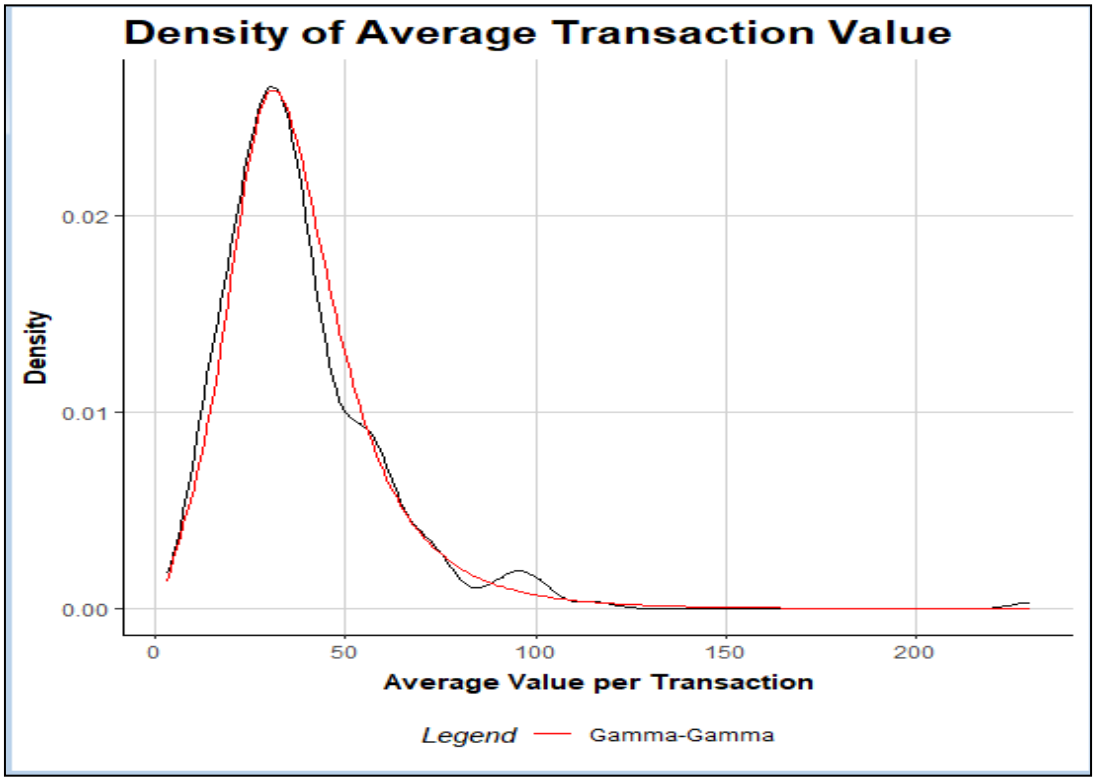
```
        Id actual.mean.spending predicted.mean.spending
  1:     1               0.00000                66.78361
  2:    10               0.00000                59.77959
  3:   100              32.06652                42.18584
  4:  1000              46.51783                40.32674
  5:  1001              33.09091                44.89572
 ---
246:  1219              29.55429                30.40038
247:   122               0.00000                35.19998
248:  1220               0.00000                37.77123
249:  1221              33.62778                31.27807
250:  1222               0.00000                47.27699
```

3.4 Plot Spendings

Code:

```
plot(est.gg)
```

Output:



Practical 7:

Aim: Extract data from social media platforms and perform analysis to gain insights into customer behavior and preferences.

1. Utilize Python libraries like Beautiful Soup and requests to scrape data from social media platforms.

What is beautiful soap library?

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It provides Pythonic idioms for

iterating, searching, and modifying the parse tree.

Here's a brief overview of what Beautiful Soup can do:

- Parsing HTML or XML Documents:

  Beautiful Soup allows you to parse HTML or XML documents and create a parse tree from the page's source code.

- Search and Navigation:

  It provides methods for searching and navigating the parse tree. You can search for tags, attributes, and text content.

- Filtering:

  You can filter content based on various criteria, making it easier to extract the specific data you're interested in.

- Modifying Content:

  Beautiful Soup supports modifying the parse tree, allowing you to add, remove, or modify tags and attributes.

  However, if you're working with data for educational purposes or on a platform that explicitly allows scraping, you can use the Beautiful Soup library along with the requests library in Python. Here's a general outline of the process:

  1. **Install the required libraries:** Make sure you have both beautifulsoup4 and requests installed. You can install them using pip:

     Code:

     pip install beautifulsoup4 requests

  2. **Import the libraries:** In your Python script, import the necessary libraries.

     Code:

     import requests

     from bs4 import BeautifulSoup

3. **Send a request to the social media platform:** Use the requests library to send a GET request to the URL you want to scrape.

   Code:

   url = "https://www.example.com"

   response = requests.get(url)

4. **Parse the HTML content:** Use Beautiful Soup to parse the HTML content of the page.

   Code:

   soup = BeautifulSoup(response.content, 'html.parser')

5. **Extract data:** Use Beautiful Soup methods to find and extract the data you need from the parsed HTML.

   Code:

   # Example: Extract all links on the page

   links = soup.find_all('a')

   for link in links:

   print(link['href'])

Code:

```
import requests
from bs4 import BeautifulSoup
url = "https://www.amazon.com"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
# Example: Extract all links on the page
links = soup.find_all('a')
for link in links:
    print(link['href'])
```

Output:

```
/ref=cs_503_logo
/ref=cs_503_link
/dogsofamazon/ref=cs_503_d
```

Add these output links at the end of website

EG: https://www.amazon.com/ref=cs_503_link

2. Clean and preprocess the scraped data.

   Cleaning and preprocessing scraped data is an essential step to ensure that the data is in a usable format for analysis or further processing. The specific steps you take will depend on the nature of the data you've scraped. Below are some common tasks you might perform using Python and its libraries like pandas, numpy, and regular expressions:

   1. Remove HTML tags:

      If your scraped data contains HTML tags, you may want to remove them to extract only the text content.

      Beautiful Soup is handy for this purpose:

      Code:

      ```
      from bs4 import BeautifulSoup
      # Assuming 'html_data' is your scraped HTML content
      soup = BeautifulSoup(html_data, 'html.parser')
      cleaned_text = soup.get_text()
      ```

   2. Remove extra whitespaces:
      Clean up extra whitespaces, leading and trailing spaces:

      Code:

      ```
      cleaned_text = ' '.join(cleaned_text.split())
      ```

   3. Handle missing values:
      If your scraped data contains missing values, you may want to handle them using pandas:

      Code:

      ```
      import pandas as pd
      # Assuming 'data' is a list of dictionaries where each dictionary represents a data point
      df = pd.DataFrame(data)
      df = df.dropna()  # Drop rows with missing values
      ```

   4. Convert data types:
      Ensure that the data types are appropriate for analysis. For example, convert strings to numbers if needed:

      Code:

      ```
      df['numeric_column'] = pd.to_numeric(df['numeric_column'])
      ```

   5. Remove duplicates:
      Remove duplicate entries from your dataset:

      Code:

      ```
      df = df.drop_duplicates()
      ```

6. Text cleaning:
   Perform text-specific cleaning tasks, such as converting text to lowercase, removing punctuation, and handling special characters:
   Code:

```
import string
def clean_text(text):
text = text.lower()
text = text.translate(str.maketrans("", "", string.punctuation))
# Add more cleaning steps as needed
return text
df['text_column'] = df['text_column'].apply(clean_text)
```

7. Tokenization:
   If you're working with text data, tokenize the text into words or tokens:
   Code:

```
from nltk.tokenize import word_tokenize
df['tokenized_text'] = df['text_column'].apply(word_tokenize)
```

8. Date parsing:
   If your data contains dates, parse them into a standardized format:
   Code:

```
df['date_column'] = pd.to_datetime(df['date_column'], format='%Y-%m-%d')
```

9. Save the cleaned data:
   Finally, save the cleaned data to a new file or overwrite the existing one:
   Code:

```
df.to_csv('cleaned_data.csv', index=False)
```

Practical 8

Aim: Analyze customer purchasing patterns and build a recommender system based on market basket analysis.

- Use transactional data to identify frequently occurring item sets using association rule mining algorithms.
- Calculate support, confidence, and lift for the identified item sets.
- Build a recommendation engine using collaborative filtering techniques.
- Evaluate the performance of the recommender system and make recommendations based on customer preferences.

1.1. DATA MINING

According to David Hand, "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner". (David Hand, Heikki Mannila, and Padhraic Smyth, 2001)

Application of data mining techniques in various fields have been very effective till now. For example, in the field of healthcare, it can help healthcare insurers detect fraud and abuse, healthcare organizations make customer relationship management decisions, physicians identify effective treatments and best practices, and patients receive better and more affordable healthcare services (Hian Chye Koh and Gerald Tan). In the field of marketing, customer segmentation involves the subdivision of an entire customer base into smaller customer groups, consisting of customers who are similar within each specific segment (Woo, Bae, & Park, 2005). This segmentation technique is useful to identify the customers and cluster them based on their characteristics and features.

One major application domain of data mining is the analysis of transactional data. In a recorded transactional database each transaction is a collection of items. The best technique to analyze and find the relationships and patterns between items is market basket analysis. It is one of the most interesting research areas of the data mining that have received more attention by researchers nowadays.

1.2. MARKET BASKET ANALYSIS

Market basket is defined as an itemset bought together by a customer on a single visit to a store. In our visit to the super market we tend to buy a lot of products from different categories and put them all together in one single basket. Which is considered to a be a single transaction. Market basket analysis is the analysis of those baskets all together.

Market basket analysis encompasses a broad set of analytic techniques aimed at uncovering the associations and connections between specific objects, discovering customer behaviors and relation between items. In retail, it is used based in the following idea, if a customer buys a certain group of items, is more (or less) likely to buy another group of items. For example, it is known that when a customer buy beer, in most of cases, buys chips as well. These behaviors produced in purchases is something that the companies selling their products are interested in. The sellers/ supermarkets are interested in analyzing which items are purchased together in order to create new marketing/sales strategies that can be helpful in improving the benefits of the company as well as customer experiences.

Most of the retail markets are more focused on the what their customer's buy. But they ignore the fact about when they buy it. Which is also considered to be a huge factor in their behavior of purchase. This thesis is focused on not just "what" the customer buys but also "when" they buy it. According to Forbes magazine marketers are constantly looking into future, trying to predict next big trend and data driven marketing is the top most trend right now in which time plays a highly significant role. So, data driven marketing with time as a crucial factor will help us predict a better future for the retail company.

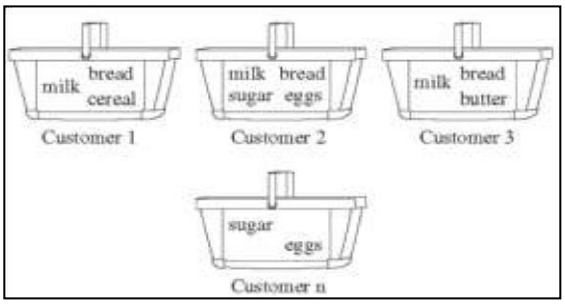


figure 1: Market basket

The market basket analysis is a powerful tool for the implementation of up-selling, cross-selling, inventory management strategies (Chen, Tang, Shen, & Hu, 2005). Market Basket Analysis is also known as association rule mining or affinity analysis, which have been used to understand consumer behavior regarding the types of the purchases they make. It is a Data Mining technique that originated in the field of marketing and was initially used to understand purchase patterns of the customers by extracting associations and co-occurrence from a transactional database (i.e. market basket data). For example, when shopping in a supermarket, consumer rarely buy one product, they far more likely to purchase an entire basket of products, mostly from different product categories. This allows us to uncover nonobvious, usually hidden and counterintuitive associations between items, products, or categories. We are also able to extract products and product categories which are purchased together, and these associations can be represented in the form of association rules. These association rules enable managers to develop marketing strategies like developing interventions, promoting specific product categories, offering promotions, etc. which eventually leads to get customers spend more money based on two different principles. *Upselling,* which consists in buying a large quantity of the same product or adding new features and *Cross-selling,* which consists in adding more products from various categories. Market Basket Analysis is also very much useful in *stock management and placement of items.*

CHAPTER 5: STRATEGY FOR MARKET BASKET ANALYSIS

In this section we describe the entire research process. Before getting into the steps of the analysis.

First, we clear some of the concepts that we will be coming across in our analysis.

5.1. KEY TERMS AND CONCEPTS

5.1.1. Association rules:

Association analysis is also known as affinity analysis or association rule mining, a method commonly used for market basket analysis. ARM is currently the most suitable method for analysis of big market basket data but when there is a large volume of sales transaction with high number of products, the data matrix to be used for association rule mining usually ends up large and sparse, resulting in longer time to process data. Association rules provide information of this type in the form of "IF-THEN" statements. There are three indexes which are commonly used to understand the *presence*, *nature* and *strength* of an association rule. (Berry & Linoff, 2004; Larose, 2005; Zhang & Zhang, 2002)

Lift is obtained first because it provides information on whether an association exist or not or if the association is positive or negative. If the value for lift suggests that there is an existence of association rule, then we obtain the value for support.

$$\boldsymbol{Lift = \frac{P(A \cap B)}{P(A) * P(B)}}$$

Support of an item or itemset is the fraction of transactions in our dataset that contain that item or itemset. It is an important measure because a rule that have low support may occur simply by chance. A low support rule may also be uninteresting from a business perspective because it may not be profitable to promote items that are seldom bought together. For these reasons, support is often used to eliminate uninteresting rules.

$$\boldsymbol{Support = P(A \cap B)/N}$$

Confidence is defined as the conditional probability that shows that the transaction containing the

LHS will also contain RHS. Association analysis results should be interpreted with caution. The inference made by an association rules does not necessarily imply causality. Instead, it suggests a strong co-occurrence relationship between the items in the antecedent and consequent of the rule.

$$Confidence = \frac{P(A \cap B)}{P(A)}$$

Confidence and support measure the strength of an association rule. Since the transactional database is quite large, there is a higher risk of getting too many unimportant and rules which may not be of our interest. To avoid these kinds of errors we commonly define a threshold of support and confidence prior to the analysis, so that only useful and interesting rules are generated in our result.

If lift is greater than 1, it suggests that the presence of the items on the LHS has increased the probability that the items on the RHS will occur on this transaction. If the lift is below 1, it suggests that the presence of the items on the LHS make the probability that the items on the RHS will be part of the transaction lower. If the lift is 1, it suggests that the presence of items on the LHS and RHS are independent: knowing that the items on the LHS are present makes **no** difference to the probability that items will occur on the RHS.

While performing market basket analysis, we look for rules with a lift of more than one. It is also preferable to have rules which have high support as this will be applicable to a large number of transactions and rules with higher confidence are ones where the probability of an item appearing on the RHS is high, given the presence of items on the LHS.

5.1.2. Antecedent and Consequent:

In every association rule we have an antecedent and a consequent, also called rule body and rule head accordingly. The generated association rule relates the rule body with the rule head. LHS is the Antecedent and RHS is the Consequent.

5.1.3. Causality:

Ideally, we would like to know that in an association rule the presence of an item/ itemset causes another item/ itemset to be bought. However, "causality" is an elusive concept. Nevertheless, for market-basket data, the following test suggests what causality means. If we lower the price of diapers and raise the price of beer, we can lure diaper buyers, who are more likely to pick up beer while in the store, thus covering our losses on the diapers. That strategy works because diapers cause beer. However, working it the other way around, running a sale on beer and raising the price of diapers, will not result in beer buyers buying diapers in any great numbers, and we lose money.

5.1.4. Frequent Itemset:

In many (but not all) situations, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that no one buys anyway. Thus, much data mining starts with the assumption that we only care about sets of items with high support; i.e., they appear together in many baskets. We then find association rules or causalities only involving a high-support set of items. The consequent must appear in at least a certain percent of the baskets, called the support threshold.

5.1.5. Time Period:

Each transaction takes place in a single time period where $W = \{W_1, W_2, W_3, \ldots, W_t\}$ is the set of all the time periods under consideration. Only one transaction occurs per store, per user in a time period.

5.1.6. Transaction:

In general terms transaction is an agreement, contract, understanding, or transfer of cash or property that takes place between two parties and establishes a legal obligation.

In accounting terms events that initiates a change in the asset, liability, or net worth account. Transactions first entry are made in journal and then posted to ledger. From there they move to other accounting books like profit and loss account, balance sheet, etc.

In banking a transaction would be an activity performed by the account holder at his/her request which is affecting a bank account.

In the language of commerce, a transaction will be considered exchange of goods or services between a buyer and a seller. It has 3 components:

1. Transfer of goods or services and money,
2. Transfer of title which may or may not be accompanied by a transfer of possession,
3. Transfer of exchange rights.

In the field of marketing, a typical transaction consists of a set of products purchased by a customer at a retail store or on a website. These transactions contain all the information about each specific transaction which make up the data entered into the database. These can include

information on the customer, information of what products were purchased in what quantity, information on time of purchases, information on if the companies marketing strategies are attracting customers or not, etc.

Also, a transaction can take place at one point in time or over time and could involve a day, a quarter, a fiscal year, or even longer periods. Because they are not limited to an event.

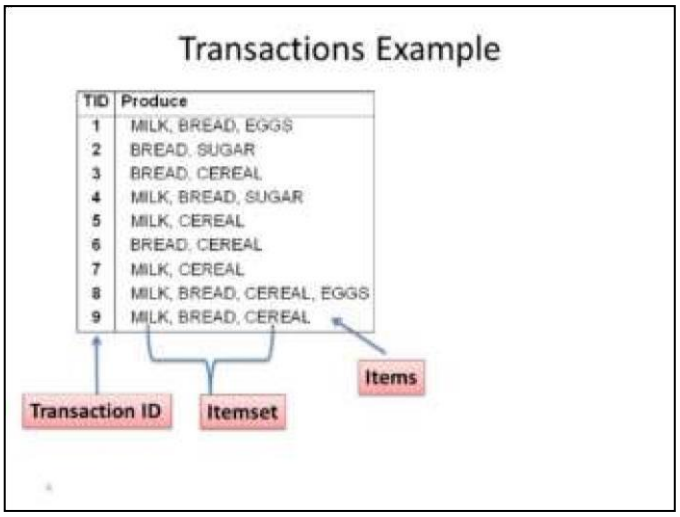


figure 2: Transaction example

5.1.7. Long Tail Effect:

This term often refers to data products purchase in supermarkets describing their distribution as a long tail in which a small number of products is purchased more frequently whereas a large one is purchased less frequently. This phenomenon creates data sparsity problem and worsens even more their elaboration.

Studying such data with MBA would be practically helpful because its transactional data and best approach to work with transactional data is to do market basket analysis.

5.2. DETERMINE SUITABILITY OF Market Basket Analysis

MBA is typically used to examine associations based on data with binary variables. However, it would not be possible to apply MBA when the data set consists of data collected from continuous or Likert-type scales only. There must be at least some categorical variables among which there is some interest or meaningfulness for deriving categorical or quantitative association rules. Apparently, the data collected by organizations as a part of their business analytics and data collected by scholars for research purposes usually includes both binary as

well as categorical variables. Therefore, MBA can be potentially used with many types of data available.

In addition, if the researcher is interested in theory building; examining multilevel, contingency, and dynamic relationships; producing results that can be easily communicated to the practitioner audience; if the data seems too messy or unstable to use, then MBA is likely to be a good methodological alternative. On the other hand, MBA is not appropriate for theory testing purposes due to its exploratory nature.

5.2.1. ADVANTAGES OF USING MARKET BASKET ANALYSIS

- It allows researchers to make use of data from stores which is in abundance to build their theory. This capability of MBA was first highlighted by Locke in 2007. He said using MBA has the potential to lead to important contributions by allowing researchers to implement an inductive approach to theory building, which, despite its advantages is currently underutilized. Indeed, it has led to insights in marketing and other fields. For example, Russell et al. (1999) pointed out multiple-category decision making i.e. theoretical models of purchasing decisions involving products in more than one category. Apart from marketing, other researches like patients with food allergies allowed Kanagawa et al. (2009) to build models regarding which allergens are related to which.

- MBA allows the researchers to use the data which appears to be messy and unusable. Given the affordability and availability of data storage systems, the organizations collect data every day on employees (their performance, training, skills, etc.), customers (frequency of visits, purchases, expenditure, etc.) and many other issues. Most of that data is collected in a very unsystematic and unorganized format without any specific study in mind. MBA is ideally suited to be used inductively with such datasets to uncover association rules that may not be readily apparent (Hafley & Lewis, 1963; Shmueli, Patel, & Bruce, 2010). Messy data often involves dirty data with lots of missing values and outliers. MBA is not immune to the problem of messy data, it just allows the interpretation of missing data as indicating that no option was selected, and association rules are less influenced by outliers compared to other traditional data analytic approaches. In the context of MBA, outliers result in infrequently occurring associations (He, Xu, Huang, & Deng, 2004)

- MBA can help in building dynamic theories, which states how important is the role of time in theory building. There are mainly two approaches of building dynamic theories via MBA i.e. multiple MBA and sequential MBA. When the available data include transactions as they have occurred overtime, multiple MBA approach is used (Tang et al. 2008). Sequential MBA is used when the available data describes individual events as they have occurred over time. It may uncover the presence of pattern in which event A occurs before event B, which occurs before event C (Han, Kim, & Sohn, 2009).

- MBA can be used to access multilevel relationships. It can be applied across all level of analysis ranging from an individual level to firm level, industry level and country level contexts.

5.2.2. CHECK MBA REQUIREMENTS

Next step would be to check two key requirements for MBA (Marakas, 2003). *Firstly*, it is necessary to have sufficiently huge number of transactions. Fortunately, it is straightforward to meet the sample size requirements because of vast storage of data being collected by firms in response to the analytics movement (Davenport et al., 2010) as well as sharp decrease in cost of data storage technology (Shmueli et al., 2010). In fact, Berry and Linoff (2004) had an argument that firms face the problem of too much data rather than too little.

It is important for the researchers to explain in detail the sources of their data so that readers can be fully aware/ informed regarding the extent to which association rules may generalize to other settings, because it may be the case even if the sample size is large, the data might be collected only in one context.

Secondly, it must be verified that the data doesn't have wrong entry or missing values which should mandatory. It is necessary to check the data for any sort of weirdness. To do so, the researchers can use exploratory data analysis techniques, such as frequency tables, unique values, date of transaction, etc. Since our data will be transactions in retail market, we can choose the important categories that we want to focus in our analysis, rather than looking in to all the categories.

Our illustrative data set meets both requirements. It has sufficiently large number of transactions and data cleaning is the part of the whole process od data exploration before starting to analyze the data.

5.2.3. APRIORI ALGORITHM

This part will explain how the algorithm that will be running behind the python libraries for Market Basket Analysis. This will help the companies to understand their clients more and analyze their data more closely and attentively. Rakesh Agrawal proposed the Apriori algorithm which was the first associative algorithm proposed and future developments in association, classification, associative classification algorithms have used it as a part of the technique.

Association rule mining is seen as a two-step approach:

1. **Frequent Itemset Generation:** Find all frequent item-sets with support >= pre-determined minimum support count. In frequent mining usually the interesting associations and correlations between item sets in transactional and relational databases are found. In short, Frequent Mining shows which items appear together in a transaction or relation. The discovery of frequent item sets is accomplished in several iterations. Counting new candidate item-sets from existing item sets requires scanning the entire training data. In short it involves only two important steps:
    a. Pruning
    b. Joining
2. **Rule Generation:** List all association rules from frequent item-sets. Calculate Support and Confidence for all the rules. Prune rules which fail minimum support and minimum confidence thresholds.

Frequent Itemset Generation scan the whole database and find the frequent itemset with a threshold on support. Since it scans the whole database, it is the most computationally expensive step. In the real-world, transaction data for retail, can exceed to Gigabytes and Terabytes of data for which an optimized algorithm is needed to exclude item-sets that will not help in later steps. For this Apriori algorithm is used.

Apriori algorithm sates “Any subset of a frequent itemset must also be frequent. In other words, no superset of an infrequent itemset must be generated or tested.”
In the image below, which is a graphical representation of the Apriori algorithm principle. It consists of k-item-set node and relation of subsets of the k-item-set. You can notice in the figure that in the bottom is all the items in the transaction data and then you start moving up creating subsets till it reaches to the null set.

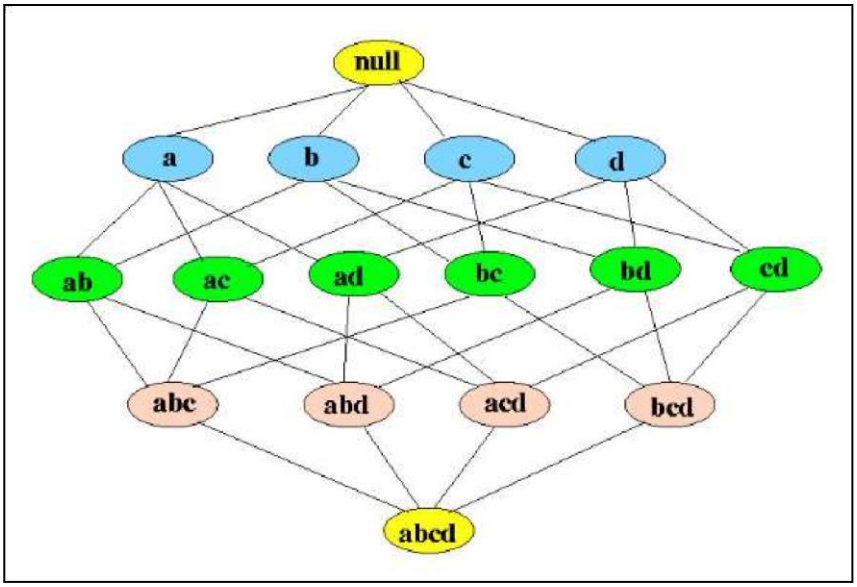


figure 3: All possible subsets

This shows that it will be difficult to generate frequent item-set by finding support for each combination. Therefore, in the figure below we can notice that Apriori algorithm helps to reduce the number of sets to be generated.

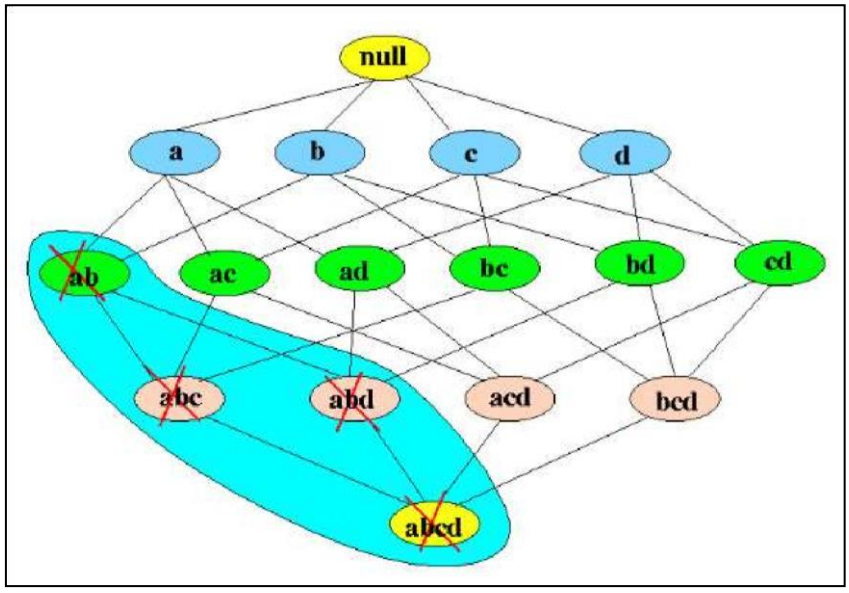


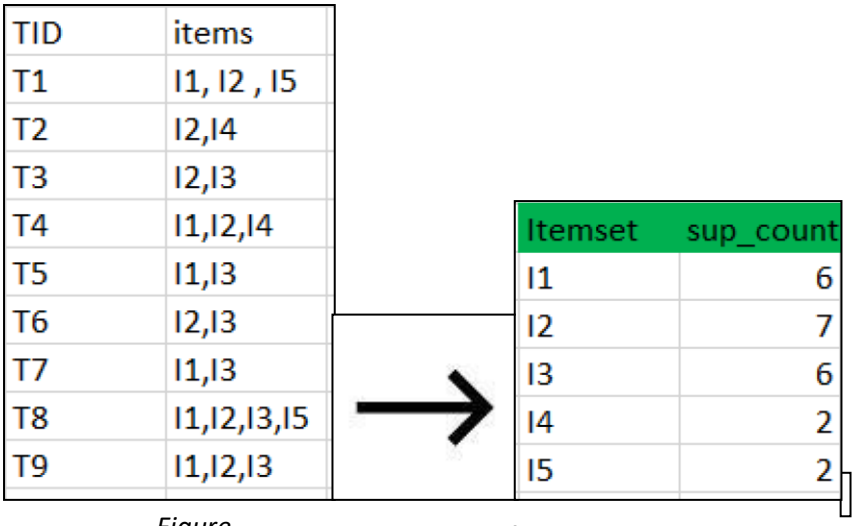
figure 4: if an item set is infrequent, we do not consider its super sets

If an item-set {a, b} is infrequent then we do not need to consider all its super sets.

We can also look at it in the form of a transactional data-set. In the following example you can notice why Apriori algorithm is much more effective and it generates stronger association rules step by step.

Step: 1

- Create a table containing support count of each item present in the dataset.
- Compare support count with minimum support count (in this case we have minimum support count = 2 and if support count is less than the minimum support count then remove those items), this gives us a new set of items.



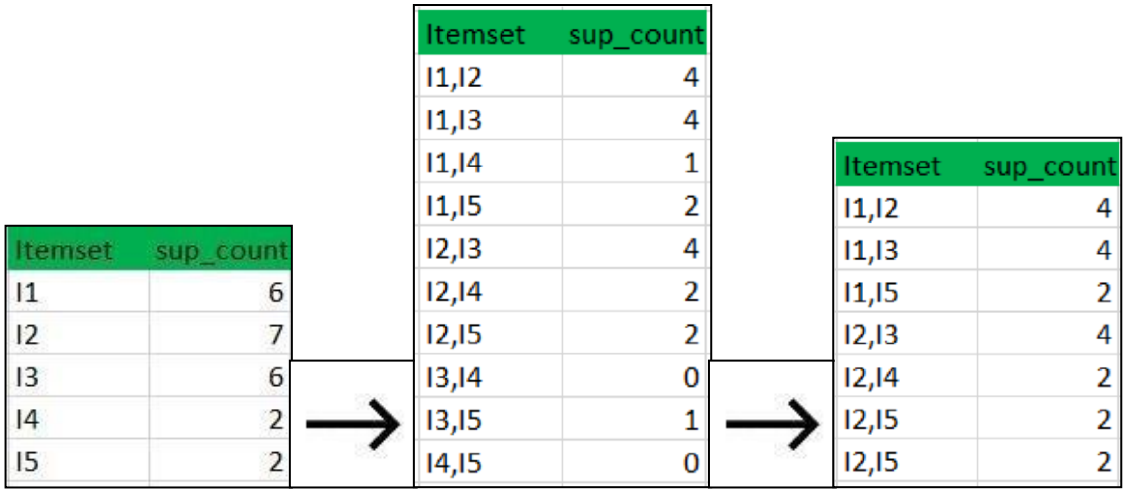
TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Figure transactional data to frequent items

Step: 2

- This step is known as join step. We generate another set by cross joining each item with one another.
- Check if the subsets of an itemset are frequent or not and if not remove that itemset. For example, in the case below we can see that the subset of {I1, I2} are {I1}, {I2} and are frequent. We must check for the each itemset in the same way.
- Now find the support count of these item-sets by searching in the dataset.
- Since we have already specified a threshold of minimum support count of 2. We compare the minimum support count and if the support count is less than the minimum support count, then remove those items. Gives us another itemset as we can see below.



Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

figure 6: pruning and joining

Step: 3

- After getting another dataset we follow the same step (I.e. join step). We cross join each itemset with each other. So, the itemset generated after this step will be:

{I1, I2, I3}

{I1, I2, I4}

{I1, I2, I5}

{I1, I3, I5}

{I2, I3, I4}

{I2, I4, I5}

{I2, I3, I5}

- Check all subsets of these item sets are frequent or not, if not then remove that item sets. For example, in this case the subset of {I1, I2, I3} are {I1, I2}, {I1, I3}, {I2, I3} which are frequent. But for {I2, I3, I4} one of the subsets is {I3, I4}, which is not frequent. So, we remove this. We do the same for every itemset.

- Once we have removed all the non-frequent item sets, find support count of the remaining itemset by searching in the dataset.
- Compare the minimum support count and if the support count is less than the minimum support count, then remove those items. It gives us another itemset as we can see below.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

→

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

figure 7: pruning and joining *again until there are no more frequent items left.*

Step: 4

- We follow the same procedure again. First, we do the join step and we cross join each itemset with one another. In out example the first two elements of the item set should match.
- After, check all subsets of these item sets are frequent or not. In our example the itemset formed after join step is {I1, I2, I3, I5}. So, one of the subsets of this itemset is {I1, I3, I5} which is not frequent. Therefore, there is no itemset left anymore.
- We stop here because no more frequent itemset are found anymore.

This was the first step of association rule mining.

The next step will be to list all frequent item-sets and generate how strong are the association rules. For that we calculate the confidence of each rule. To calculate confidence, we use the following formula:

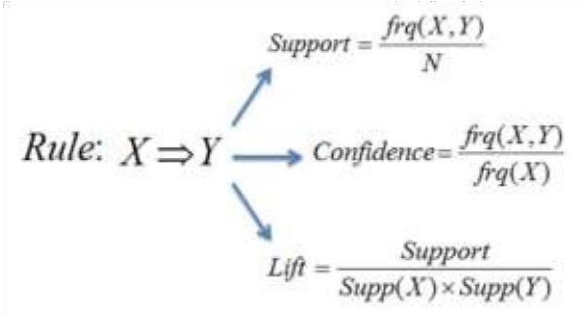


figure 8: support, confidence and lift calculation

By taking an example of any frequent Item (we took {I1, I2, I3}) we will show how rule generation is done:

Rules	Formula	Confidence
{I1, I2} => {I3}	2 / 4 * 100	50,00%
{I1, I3} => {I2}	2 / 4 * 100	50,00%
{I2, I3} => {I1}	2 / 4 * 100	50,00%
{I1} => {I2, I3}	2 / 6 * 100	33,33%
{I2} => {I1, I3}	2 / 7 * 100	28,57%
{I3} => {I1, I2}	2 / 6 * 100	33,33%

figure 9: calculation of confidence

So, in this case if the minimum confidence is 50% then the first 3 rules can be considered strong association rules. For example, take **{I1, I2}** => **{I3}** having confidence equal to 50% tells that 50% of people who bought **I1** and **I2** also bought **I3.**

5.2.4. OTHER ALGORITHMS USED IN MARKET BASKET ANALYSIS

Even though Apriori algorithm is the mostly used algorithm for market basket analysis. There have been different approaches taken by the researchers to build different algorithms since accuracy and efficiency are crucial factors in data mining.

The AIS algorithm was the first algorithm proposed for mining association rule. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contains one item, for example we only generate rules like $X \cap Y \rightarrow Z$ but not those rules as $X \rightarrow Y \cap Z$. The main drawback of the AIS algorithm is too many candidate itemset that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database. Apriori is more efficient during the candidate generation process. Apriori uses pruning techniques to avoid measuring certain item sets, while guaranteeing completeness. These are the item sets that the algorithm can prove will not turn out to be large. However, there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements.

As we know that apriori algorithm was an algorithm proposed by Rakesh Agrawal in 1993. Since then there have been future developments in association, classification, associative classification algorithms, which have used apriori as part of the technique.

In 1998, Liu el al. Proposed CBA the first associative classification algorithm. CBA implements the famous Apriori algorithm in order to discover the frequent rule items. CBA selects high confidence rules to represent the classifier. Finally, to predict a test case, CBA applies the highest confidence rule whose body matches the test case. Experimental results designated that CBA derives higher quality classifiers with regards to accuracy than rule induction and decision tree classification approaches.

Han et al., in 2000 presented a new association rule mining approach that does not use candidate rule generation called FP-growth that generates a highly condensed frequent pattern tree (FP-tree) representation of the transactional database. Each database transaction is represented in the tree by at most one path. FP-tree is smaller in size than the original database the construction of it requires two database scans:

1. Frequent item sets along with their support in each transaction are produced
2. FP-tree is constructed.

The mining process is performed by concatenating the pattern with the ones produced from the conditional FP-tree. One constraint of FP-growth method is that memory may not fit FP-tree especially in dimensionally large database.

In 2001, Li et al., recommended Classification based on Multiple Association Rules (CMAR). This method is an extension of FP-growth, constructs a class distribution-associated FP-tree, and mines large database efficiently. Moreover, it applies a CR-tree structure to store and retrieve mined association rules efficiently, and prunes rules effectively based on confidence, correlation and database coverage. The classification is performed based on a weighted x2 analysis using multiple strong association rules. Extensive experiments on 26 datasets from UCI machine learning database repository show that CMAR is consistent, highly effective at classification of various kinds of databases and has better average classification accuracy in comparison with CBA.

In 2003, Yin et al., suggested Classification based on Predictive Association Rules (CPAR). CPAR integrates the features of associative classification in predictive rule analysis. It has the following advantages:

1. CPAR generates a much smaller set of high-quality predictive rules directly from the dataset.

2. Avoids redundant rules, CPAR generates each rule by considering the set of "already generated" rules.

3. Predicting the class label of an example, CPAR uses the best k rules the example satisfies.

Moreover, CPAR employs the following features to further improve its accuracy and efficiency:

1. CPAR uses dynamic programming to avoid repeated calculation in rule generation.
2. Rule generation considers all the close-to-the-best literals are selected so that important rules will not be missed.

CPAR generates a smaller set of rules, with higher quality and lower redundancy in comparison with associative classification. As a result, CPAR is much more time efficient in both rule generation and prediction but achieves as high accuracy as associative classification.

The MCAR algorithm introduced by Fadi et al., in 2005 uses an intersection technique for discovering frequent rule items. MCAR takes advantage of vertical format representation and uses an efficient technique for discovering frequent items based on recursively intersecting the frequent items of size *n* to find potential frequent items of size *n+1*. MCAR consists of two main phases:

1. Rule generation and
2. A classifier builder.

In the first phase, the training data set is scanned once to discover frequent one rule items, and then MCAR recursively combines rule items generated to produce potential frequent rule items (candidate rule items) involving more attributes. The supports and confidences for candidate rule items are calculated simultaneously, where any rule item with support and confidence larger than minimum support and minimum confidence, respectively, is created as a potential rule.

In the second phase, the classifier builder ensures that each training instance is covered by at most one rule, which has the highest precedence among all rules applicable to it.

Furthermore, there are no useless rules in the MCAR classifier since every rule correctly covers at least one training instance. This approach is similar to the CBA classifier builder as each rule in CBA also covers at least one training instance. However, the way MCAR builds the classifier by locating training instances is more efficient than that of CBA due to abounding going through the training data set multiple times.

CHAPTER 6: PRACTICAL IMPLEMENTATION OF MARKET BASKET ANALYSIS ON THE DATASET

6.1. DATA COLLECTION

MBA was originally designed for the use with large datasets that are usually collected by others (i.e., not the scholars or the research team). Thus, the researchers can seek out partnerships with organizations that will provide data in exchange for conducting analysis and presenting results to those who provided the data. It is also possible for the research team to collect the data but given the time involved, effort and issues to access the data sources, most MBA studies are done on the data collected by other parties. In the chapter of literature review we were able to see that researchers have been able to create data-sharing partnerships in many different fields.

In addition to using data collected by organizations, researchers can implement a third-party data collection strategy consisting of reliance on publicly available information. For example, O'Boyle and Aguinis (2012) investigated issues regarding individual performance by using data on academics, entertainers, politicians, and amateur and professional athletes.

The data for this research was collected from a Belgian known super market chain and are related to customer transactions for the period from April 2018 to June 2018. Data is extracted from the information system of the company and come exclusively from customer transactions through loyalty cards. Every transaction is a record of the purchase carried out by the customer whoever used the loyalty card at the time of the purchase.

6.2. DATASET DESCRIPTION

Number of Attributes: 11

Time period of the data: 3 months (April, May, June)

Number of transactions: 78839

Number of customers: 42468

6.2.1. ATTRIBUTES

S. No	Attribute	Description
1	random_cust_no	This is a customer ID which was replaced by random number by the company itself.
2	orig_invoice_id	This is the most important attribute of the data. It can be called invoice ID or Transaction ID. The whole analysis is based on transactions. It is explained in detail under define the transactions.
3	date_of_day	This is the day when the transaction took place.
4	minute_desc	For more accuracy we do not just keep track of day of the transactions but also time of transaction. Since in case of a super market there are hundreds of transactions happening every day.
5	mikg_art_no	It is the code of the article / item.
6	art_name	Name of the article / item
7	catman_buy_domain_desc	The categories are in hierarchy level. This is the top level of the category under which the article falls. Category_1
8	pcg_main_cat_desc	This is the second level of category under which the article comes. Category_2
9	pcg_cat_desc	This is the third level of category under which the article comes. This is the important level of category since we used this category in our analysis. The association rules are based on this level of category. Category_3
10	pcg_sub_cat_desc	This is the last and final level of category under which the article comes. Category_4
11	quantity	This is the quantity of product bought by the customer. We used this attribute to change the transactional data to binary representation.

TABLE 2: ATTRIBUTES DESCRIPTION

6.3. TOOLS USED FOR ANALYSIS

- Jupyter notebook

- Python Libraries ○ Pandas ○ Datetime ○ Matplotlib ○ Mlxtend ○ Random
  ○ xlsxwriter
- MS Excel

6.4. DATA EXPLORING & TRANSFORMATION

Before doing anything with the data as soon as we read in the data in jupyter notebook, we start exploring it. Data exploring is the most important step for any analysis. While, exploring the data the issues that needed fixing were as follows:

1. The attribute *date_of_day* was read as string variable but in this research, time is a very crucial factor, so its data type was changed to date type.

2. For comparison of data of each month we needed to separate the data based on each month. Since date_of_day attribute was changed to date type. It was very easy to get the month of each transaction.

3. As explained earlier in the long tail effect the data sparsity problem tackled up to some extent by considering only the transactions in which total items purchased were more than

8 items. While exploring the data it's important to look at the data distribution. For market basket analysis it has always been observed to have the long tail effect. The long tail effect has assumed its present connotation and dynamics by Chris Anderson [21]. This term often refers to data products purchase in supermarkets describing their distribution as a long tail in which a small number of products is purchased more frequently whereas a large one is purchased less frequently. This phenomenon creates data sparsity problem and worsens even more their elaboration. For this research from the total number of transactions there were selected, those that included at least the purchase of 8 products. That is how, to some extent, it was tried to resolve the data sparsity problem and prevent the removal of any market basket product, which may present a risk of information loss. Figure below shows the curve of long tail effect.

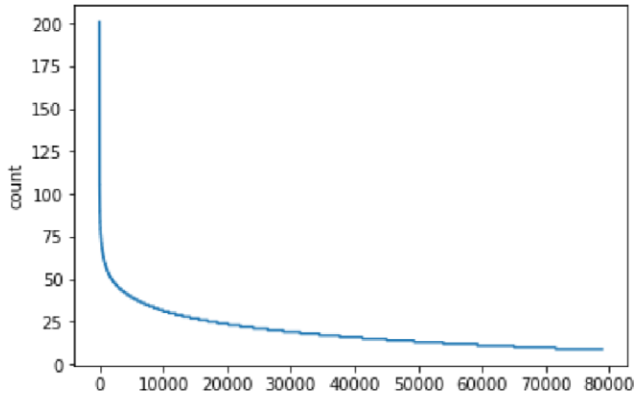


figure 10: long tail effect in our dataset

Some general exploring of data was done after all the fixes. In the table we can see the number of unique transactions, number of unique customers, number of unique articles sold, and number of articles sold in the period of 3 months as well as in each month (i.e. April, May, June).

Time	3 months	April	May	June
Number of Transactions	78839	24803	27710	26326
Number of Customers	42468	18743	20436	19347
Number of Unique Articles Sold	42694	31286	31035	31462
Number of Total Articles Sold	1541130	482776	544023	514331

TABLE 3: DATA EXPLORING

Since there are several categories and sub categories in the super market. We can easily identify the hierarchy of the categories based on the number of unique categories in the table below.

Category/Sub category	Total
catman_buy_domain_desc (Category_1)	87
pcg_main_cat_desc (Category_2)	348
pcg_cat_desc (Category_3)	1334
pcg_sub_cat_desc (Category_4)	3783

TABLE 4: CATEGORY AND SUBCATEGORIES DISTRIBUTION

6.5. BINARY REPRESENTATION

Binary data are used in information systems because data are categorized for attributes value by 0 and 1. It represents either the attribute category is present or absent in the data.

For super markets, the databases are very large and usually represented in binary format. The binary format is a matrix in which Transactions forms the rows and the items/ attributes forms the columns. For a specific transaction, if an item is purchased then the matrix position

is made as 1. If the item is not purchased in the transaction, then the matrix position will be made 0. Binary data sets are interesting and useful due to its computational efficiency and minimal storage capacity.

6.6. ANALYSIS & DISCUSSION

6.6.1. Association rules from dataset

Since we have the data for three months, we divide the data into subset of each month to compare the analysis output of each month.

Firstly, we look at the comparison of number of rules when the minimum support count is less than 0.01 and minimum lift is 1.

Secondly, for all the data and for each of the subsets we have applied market basket analysis and recorded the results of the three indicators, i.e. confidence, support and lift. For every application of a market basket analysis process, the minimum level of confidence, support and lift established in the program, is 0.5, 0.01 and 1 respectively. The results of the number of rules that came out, are registered in table below.

Time	Minimum support = 0.01 Lift >= 1	Minimum support = 0.01 Confidence >= 0.5 Lift >= 1
3 months	17282	1534
April	16564	1371
May	19380	1760
June	19272	1900

TABLE 5: ASSOCIATION RULES WITH DIFFERENT THRESHOLDS

In the first section we looked at the number of association rules with minimum support 0.01 and lift greater than 1. This criterion was applied to overall data as well as to monthly data. Our results were:

May > June > 3 months > April

We also calculated the number of association rules by adding degree of confidence greater than 0.5, the result changed to:

June > May > 3 months > April

6.6.2. Time development of association rules

The 128 rules, at the level of 1334 subcategories of products, resulting from the extraction process of overall 3 months data were used as a base for additional rules analysis at monthly

level. For each of the 128 rules, there have been recorded the values for the three indicators, lift, support and confidence during each month (i.e. April, May, June). Thus, the table was created with the monthly development of association rules measurement indicators. The table can be found at the end of the document in ANNEX section in the end of the document.

In the end of the ANNEX we can notice that there are 2 association rules **{'COLA KZH'} => {'GROENTEN'}** and **{'GROENTEN'} => {'COLA KZH'}** which do not exist in the month of

May, because they didn't pass the threshold of minimum support of 0.01 and lift greater than equal to 1. But these two rules only do not exist in the month of May They exist in the month of April and June.

Similarly, there are 2 association rules **{'GROENTEN'} => {'KOEKJES'}** and **{'KOEKJES'} => {'GROENTEN'}** which do not exist in the month of April and June. But they do exist in the month of May.

Whereas, the 128 association rules where chosen based on the threshold of minimum support as 0.05 and lift greater than and is equal to 1 from overall dataset. Whereas, to match with these 128 association rules the threshold set for monthly data was minimum support 0.01 and lift >= 1

Through these examples we can also say that there is a difference in the strength of the association rules in different time periods Its not just minor fluctuations in their strengths but sometimes it can be greater.

6.6.3. Top 30 association rules comparison

To show the fluctuation of degree of confidence of top 30 rule in each month. We have selected the rules with highest degree of confidence. In the table below, we can see the top 30 rules and their degree of confidence in 3 months all together as well as individually.

We can notice some major differences in rule number 4, 9, 18 and 19 in which we can see that those rules had lower degree of confidence in the month of April compared to the month of May and June. Also, in rule number 30 we can see that the degree of confidence for this rule is increasing with time.

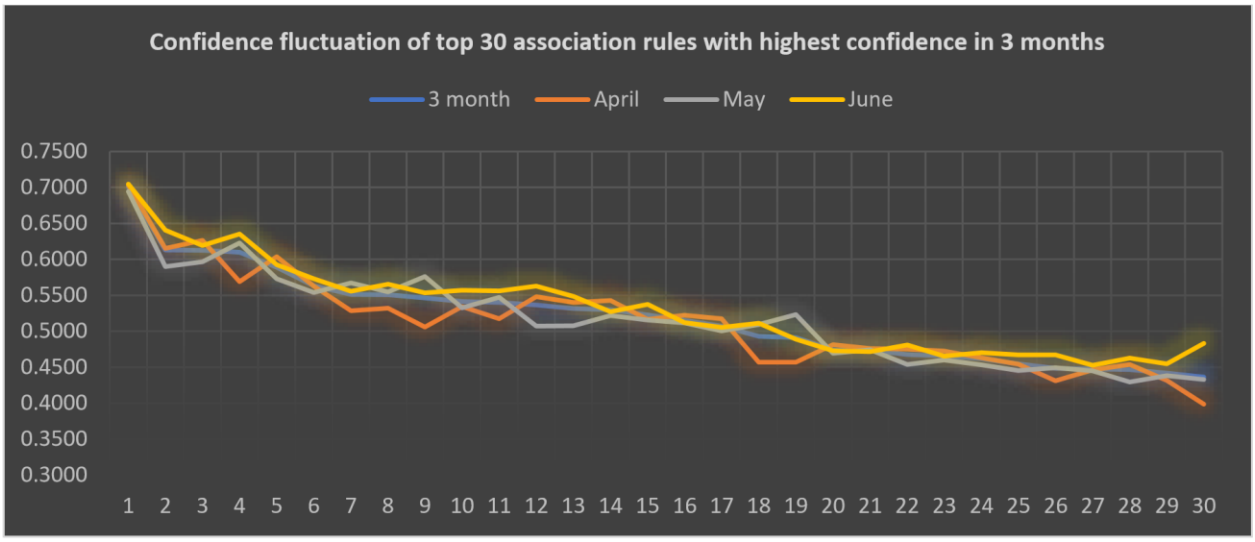


figure 11: Degree of confidence of top 30 rules

1	{'GROENTEN', 'CUCURBITACEAE'} => {'TOMATEN'}	16	{'GEBAK'} => {'BROOD'}
2	{'CUCURBITACEAE'} => {'TOMATEN'}	17	{'TOMATEN', 'BROOD'} => {'GROENTEN'}
3	{'CUCURBITACEAE', 'TOMATEN'} => {'GROENTEN'}	18	{'KRUIDEN + BLOEMEN'} => {'TOMATEN'}
4	{'PAPRIKA'} => {'TOMATEN'}	19	{'GROENTEN', 'TOMATEN'} => {'CUCURBITACEAE'}
5	{'SALADES', 'TOMATEN'} => {'GROENTEN'}	20	{'SCHARRELKIP'} => {'GROENTEN'}
6	{'SALADES', 'GROENTEN'} => {'TOMATEN'}	21	{'BANANEN'} => {'BROOD'}
7	{'BOLLEN'} => {'TOMATEN'}	22	{'STENGELGROENTEN'} => {'GROENTEN'}
8	{'KRUIDEN + BLOEMEN'} => {'GROENTEN'}	23	{'KIP'} => {'GROENTEN'}
9	{'PAPRIKA'} => {'CUCURBITACEAE'}	24	{'SALADES'} => {'GROENTEN'}
10	{'PAPRIKA'} => {'GROENTEN'}	25	{'EXOTISCH/TROPISCH'} => {'GROENTEN'}
11	{'WORTELGROENTE'} => {'TOMATEN'}	26	{'EXOTISCH/TROPISCH'} => {'TOMATEN'}
12	{'CUCURBITACEAE'} => {'GROENTEN'}	27	{'SALADES'} => {'BROOD'}
13	{'BOLLEN'} => {'GROENTEN'}	28	{'EUROPA'} => {'GROENTEN'}
14	{'TOMATEN'} => {'GROENTEN'}	29	{'SALADES'} => {'TOMATEN'}
15	{'WORTELGROENTE'} => {'GROENTEN'}	30	{'EXOTISCH/TROPISCH'} => {'BESSEN'}

TABLE 6: 30 ASSOCIATION RULES WITH HIGHEST DEGREE OF CONFIDENCE

Degree of support of top 30 rule in each month. In figure 12 and table 7 we can see the top 30 associations rules with highest support in 3 months all together as well as individually.

When we look at the figure, we can notice that there is a difference in the support level of most of the association rules. But for most of them the support is increasing with the increase in time. Most of the association rules had lower support in the month of April but it eventually increased in the month of June.

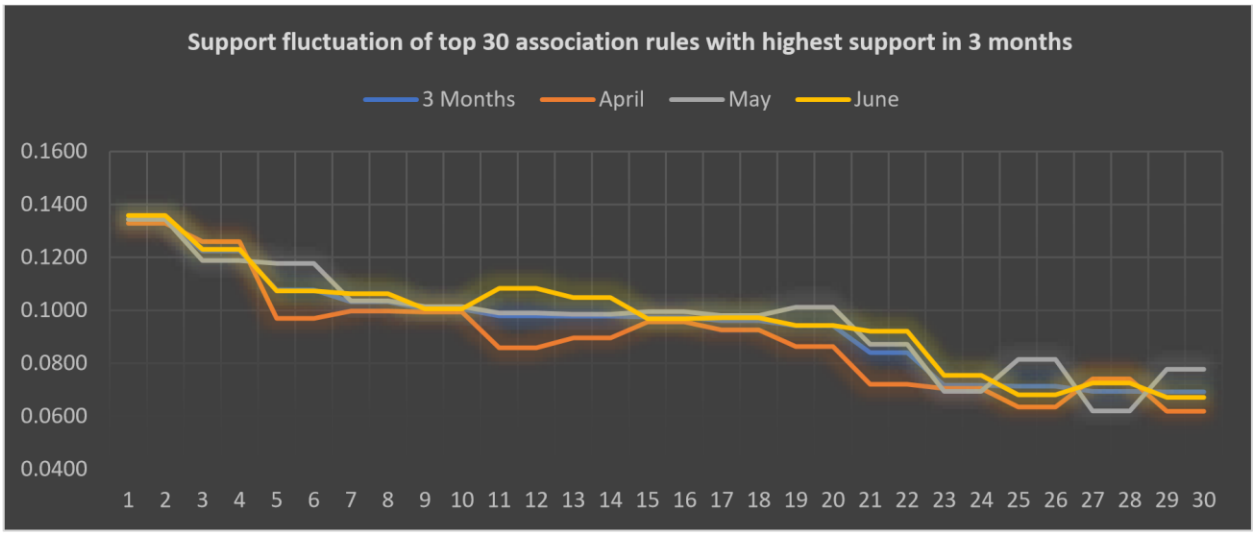


figure 12: Degree of support of top 30 rules

1	{'GROENTEN'} => {'TOMATEN'}	16	{'SALADES'} => {'BROOD'}
2	{'TOMATEN'} => {'GROENTEN'}	17	{'SALADES'} => {'TOMATEN'}
3	{'GROENTEN'} => {'BROOD'}	18	{'TOMATEN'} => {'SALADES'}
4	{'BROOD'} => {'GROENTEN'}	19	{'GROENTEN'} => {'CUCURBITACEAE'}
5	{'CUCURBITACEAE'} => {'TOMATEN'}	20	{'CUCURBITACEAE'} => {'GROENTEN'}
6	{'TOMATEN'} => {'CUCURBITACEAE'}	21	{'TOMATEN'} => {'BESSEN'}
7	{'TOMATEN'} => {'BROOD'}	22	{'BESSEN'} => {'TOMATEN'}
8	{'BROOD'} => {'TOMATEN'}	23	{'KIP'} => {'GROENTEN'}
9	{'SALADES'} => {'GROENTEN'}	24	{'GROENTEN'} => {'KIP'}
10	{'GROENTEN'} => {'SALADES'}	25	{'CUCURBITACEAE'} => {'BROOD'}
11	{'BESSEN'} => {'BROOD'}	26	{'BROOD'} => {'CUCURBITACEAE'}
12	{'BROOD'} => {'BESSEN'}	27	{'GEBAK'} => {'BROOD'}
13	{'BESSEN'} => {'GROENTEN'}	28	{'BROOD'} => {'GEBAK'}
14	{'GROENTEN'} => {'BESSEN'}	29	{'CUCURBITACEAE'} => {'SALADES'}
15	{'BROOD'} => {'SALADES'}	30	{'SALADES'} => {'CUCURBITACEAE'}

TABLE 7: 30 ASSOCIATION RULES WITH HIGHEST DEGREE OF SUPPORT

In the table 8 we can see what the top 30 association rules with the highest Lift are and in figure 13 we can see what the lift values of those 30 association rules in 3 months are all together and each month periodically.

In the comparison of lift we can notice that there is a clear difference in the lift of association rules number 1 and 2 in all the 3 months. But as the lift gets smaller and smaller the difference in different time periods decreases. Rule 5 and 6 also had higher lift in the month of April but in the month of May and June it dropped.

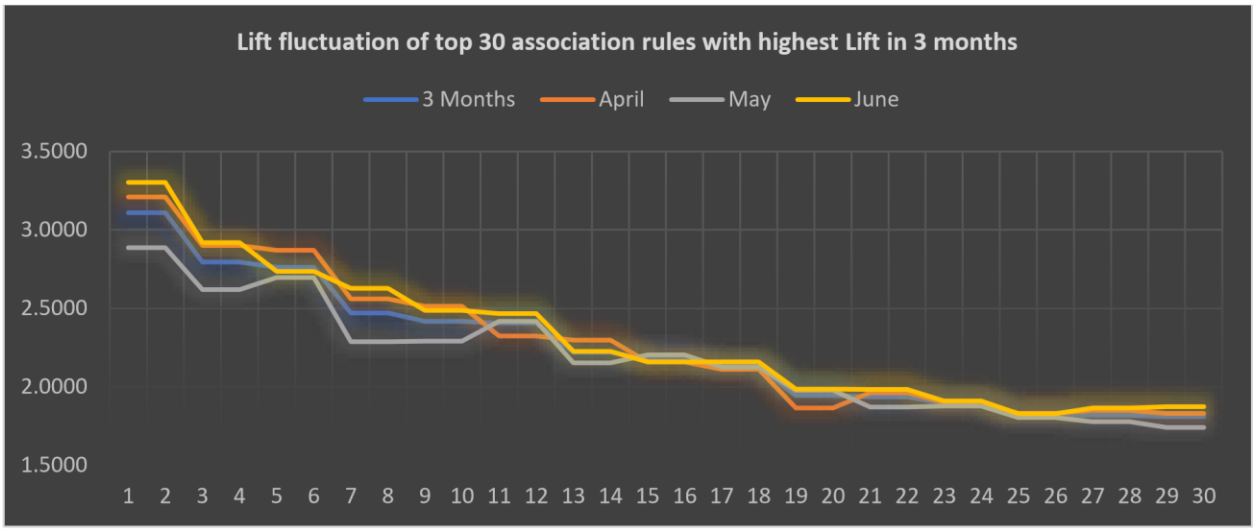


figure 13: Lift of top 30 association rules

1	{'CUCURBITACEAE'} => {'PAPRIKA'}	16	{'TOMATEN'} => {'BOLLEN'}
2	{'PAPRIKA'} => {'CUCURBITACEAE'}	17	{'TOMATEN'} => {'WORTELGROENTE'}
3	{'GROENTEN', 'TOMATEN'} => {'CUCURBITACEAE'}	18	{'WORTELGROENTE'} => {'TOMATEN'}
4	{'CUCURBITACEAE'} => {'GROENTEN', 'TOMATEN'}	19	{'TOMATEN'} => {'KRUIDEN + BLOEMEN'}
5	{'TOMATEN'} => {'GROENTEN', 'CUCURBITACEAE'}	20	{'KRUIDEN + BLOEMEN'} => {'TOMATEN'}
6	{'GROENTEN', 'CUCURBITACEAE'} => {'TOMATEN'}	21	{'GROENTEN', 'TOMATEN'} => {'SALADES'}
7	{'CUCURBITACEAE'} => {'WORTELGROENTE'}	22	{'SALADES'} => {'GROENTEN', 'TOMATEN'}
8	{'WORTELGROENTE'} => {'CUCURBITACEAE'}	23	{'GROENTEN'} => {'CUCURBITACEAE', 'TOMATEN'}
9	{'CUCURBITACEAE'} => {'TOMATEN'}	24	{'CUCURBITACEAE', 'TOMATEN'} => {'GROENTEN'}
10	{'TOMATEN'} => {'CUCURBITACEAE'}	25	{'SALADES', 'TOMATEN'} => {'GROENTEN'}
11	{'PAPRIKA'} => {'TOMATEN'}	26	{'GROENTEN'} => {'SALADES', 'TOMATEN'}
12	{'TOMATEN'} => {'PAPRIKA'}	27	{'EXOTISCH/TROPISCH'} => {'BESSEN'}
13	{'TOMATEN'} => {'SALADES', 'GROENTEN'}	28	{'BESSEN'} => {'EXOTISCH/TROPISCH'}
14	{'SALADES', 'GROENTEN'} => {'TOMATEN'}	29	{'CUCURBITACEAE'} => {'SALADES'}
15	{'BOLLEN'} => {'TOMATEN'}	30	{'SALADES'} => {'CUCURBITACEAE'}

TABLE 8: 30 ASSOCIATION RULES WITH HIGHEST LIFT

Practical 9 :

Aim:

- **Analyze customer transaction data to calculate RFM scores.**
- **Segment customers into different groups using clustering algorithms such as k-means or hierarchical clustering.**
- **Perform descriptive analysis on each customer segment to understand their characteristics.**
- **Develop targeted marketing strategies for each segment based on their RFM profiles**

9.1. Analyze customer transaction data to calculate RFM scores.

Here's an example of how you can create sample transaction data and save it to a CSV file:

```
> transaction_data <- data.frame(
+  CustomerID = rep(1:100, each = 5),  # 100 customers with 5 transactions each
+ TransactionDate = sample(seq(as.Date('2022-01-01'), as.Date('2023-01-01'), by = "day"), 500, replace = TRUE),
+  PurchaseAmount = round(runif(500, min = 10, max = 200), 2)  # random purchase amounts
+ )
> write.csv(transaction_data, "transaction_data.csv", row.names = FALSE)
>
> head(transaction_data)
```

Output:

	CustomerID	TransactionDate	PurchaseAmount
1	1	2022-01-07	134.93
2	1	2022-06-17	22.12
3	1	2022-02-10	121.22
4	1	2022-01-09	184.45
5	1	2022-02-10	15.40
6	2	2022-01-16	149.78

➔ **Load and Prepare the Data**:

```
> # Load necessary packages
> install.packages("dplyr") # if not installed
> library(dplyr)
>
> # Read the transaction data from CSV file
> transaction_data <- read.csv("transaction_data.csv")
>
```

```
> # Convert TransactionDate to a proper date format
> transaction_data$TransactionDate <- as.Date(transaction_data$TransactionDate)
>
> # Check the structure of the data
> str(transaction_data)
```

Output:

```
'data.frame':   500 obs. of  3 variables:
 $ CustomerID     : int  1 1 1 1 1 2 2 2 2 2 ...
 $ TransactionDate: Date, format: "2022-01-07" "2022-06-17" ...
 $ PurchaseAmount : num  134.9 22.1 121.2 184.4 15.4 ...
```

➔ **Calculate RFM Metrics**:

```
> # Calculate Recency (R)
> # Assuming the current date is 'today', adjust this as needed
> current_date <- as.Date("2023-12-09")
>
> recency_data <- transaction_data %>%
+  group_by(CustomerID) %>%
+  summarize(Recency = as.numeric(difftime(max(TransactionDate), current_date,
units = "days")))
>
> # Calculate Frequency (F)
> frequency_data <- transaction_data %>%
+  group_by(CustomerID) %>%
+  summarize(Frequency = n_distinct(TransactionDate))
>
> # Calculate Monetary (M)
> monetary_data <- transaction_data %>%
+  group_by(CustomerID) %>%
+  summarize(Monetary = sum(PurchaseAmount))
>
> # Merge the RFM metrics into one dataframe
> rfm_data <- merge(merge(recency_data, frequency_data, by = "CustomerID", all =
TRUE), monetary_data, by = "CustomerID", all = TRUE)
>
> # Check the RFM data
> head(rfm_data)
```

Output:

	CustomerID	Recency	Frequency	Monetary
1	1	-540	4	478.12
2	2	-359	5	614.64
3	3	-462	5	608.95
4	4	-399	5	520.61
5	5	-389	5	441.33
6	6	-344	5	387.49

➔ **Normalize and Score RFM Metrics**:

```
> # Normalize the RFM metrics (if needed)
> rfm_data$Recency <- scale(rfm_data$Recency)
> rfm_data$Frequency <- scale(rfm_data$Frequency)
> rfm_data$Monetary <- scale(rfm_data$Monetary)
>
> # Calculate RFM scores (from 1 to 5)
> rfm_data$R_Score <- as.integer(cut(rfm_data$Recency, breaks = 5, labels =
FALSE))
> rfm_data$F_Score <- as.integer(cut(rfm_data$Frequency, breaks = 5, labels =
FALSE))
> rfm_data$M_Score <- as.integer(cut(rfm_data$Monetary, breaks = 5, labels =
FALSE))
>
> # Calculate combined RFM score
> rfm_data$RFM_Score <- rfm_data$R_Score + rfm_data$F_Score +
rfm_data$M_Score
>
> # Check the final RFM scores
> head(rfm_data)
```

Output:

	CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score
1	1	-2.442528975	-4.874423	-0.3548341072	1	1	3	5
2	2	0.688031459	0.203101	0.7847110886	5	5	4	14
3	3	-1.093447683	0.203101	0.7372161269	3	5	4	12
4	4	-0.003805101	0.203101	-0.0001661072	4	5	3	12
5	5	0.169154039	0.203101	-0.6619236747	5	5	2	12
6	6	0.947470169	0.203101	-1.1113311853	5	5	2	12

9.2. Segment customers into different groups using clustering algorithms such as k-means or hierarchical clustering.

```
# Assuming you've already calculated and prepared the RFM data (rfm_data) as mentioned
earlier


# Select RFM scores for clustering

rfm_for_clustering <- rfm_data[, c("R_Score", "F_Score", "M_Score")]


# Standardize the data (if needed)

# rfm_for_clustering <- scale(rfm_for_clustering)
```

```
# Find optimal number of clusters using elbow method
wss <- sapply(1:10, function(k) {
kmeans(rfm_for_clustering, centers = k)$tot.withinss
})

# Plot the elbow graph to find the optimal number of clusters
plot(1:10, wss, type = "b", xlab = "Number of Clusters",
     ylab = "Within-cluster Sum of Squares", main = "Elbow Method")

# Based on the elbow method, select the optimal number of clusters
# Let's say, for example, the elbow is at k = 4

# Perform k-means clustering with the chosen number of clusters
k <- 4 # Change this based on the elbow method or your choice
set.seed(123)
kmeans_model <- kmeans(rfm_for_clustering, centers = k)

# Add cluster labels to the RFM data
rfm_data$Cluster <- as.factor(kmeans_model$cluster)

# Check the cluster centers
print(kmeans_model$centers)
```

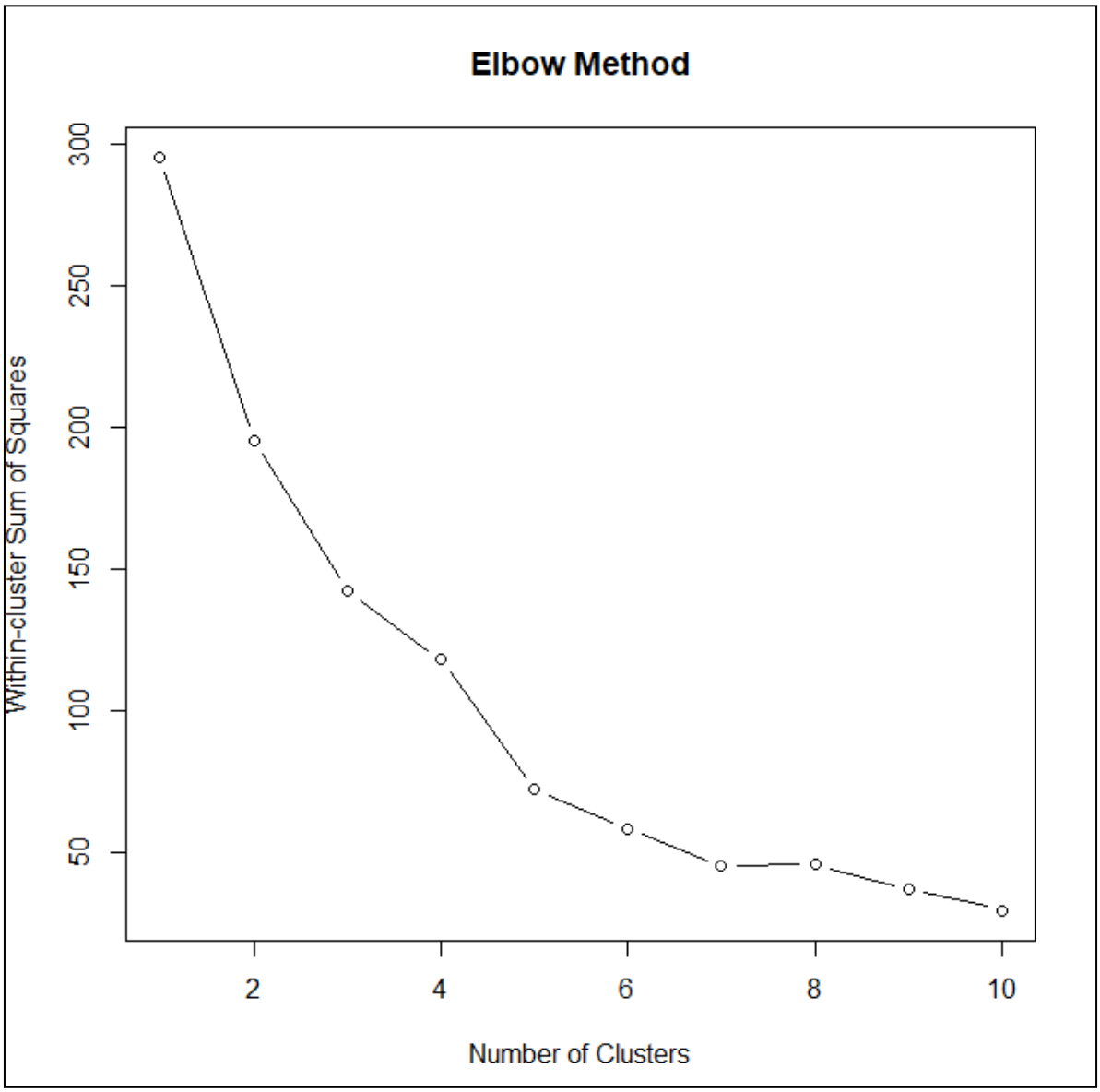
Output:

	R_Score	F_Score	M_Score
1	4.680000	5	3.6200
2	2.187500	5	3.3125
3	3.000000	1	2.5000
4	4.633333	5	1.8000

```
# View the segmented customers
head(rfm_data)
```

Output:

	CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score	Cluster
1	1	-2.442528975	-4.874423	-0.3548341072	1	1	3	5	3
2	2	0.688031459	0.203101	0.7847110886	5	5	4	14	1
3	3	-1.093447683	0.203101	0.7372161269	3	5	4	12	2
4	4	-0.003805101	0.203101	-0.0001661072	4	5	3	12	1
5	5	0.169154039	0.203101	-0.6619236747	5	5	2	12	4
6	6	0.947470169	0.203101	-1.1113311853	5	5	2	12	4



9.3. Perform descriptive analysis on each customer segment to understand their characteristics.

> # Assuming rfm_data contains RFM scores and Cluster labels as generated from the previous steps

>

```
> # Descriptive analysis by cluster
> cluster_summary <- rfm_data %>%
+  group_by(Cluster) %>%
+  summarise(
+    Avg_Recency = mean(R_Score),
+    Avg_Frequency = mean(F_Score),
+    Avg_Monetary = mean(M_Score),
+    Total_Customers = n()
+    # Add other relevant metrics as needed
+  )
>
> # View the summary statistics for each cluster
> print(cluster_summary)
```

Output:

```
# A tibble: 4 × 5
  Cluster Avg_Recency Avg_Frequency Avg_Monetary Total_Customers
  <fct>         <dbl>         <dbl>        <dbl>           <int>
1 1              4.68             5         3.62              50
2 2              2.19             5         3.31              16
3 3              3                1         2.5                4
4 4              4.63             5         1.8               30
```

9.4. Develop targeted marketing strategies for each segment based on their RFM profiles.

Applying Strategies in R:

You can further use R to target these strategies to the respective segments:

```
> # Create a function to assign marketing strategies based on clusters
> assign_strategy <- function(cluster) {
+  if (cluster == "Cluster 1") {
+    return("Re-Engagement Campaign")
+  } else if (cluster == "Cluster 2") {
+    return("Loyalty Rewards Program")
+  } else if (cluster == "Cluster 3") {
+    return("VIP or Exclusive Membership")
+  } else {
+    return("Other")
+  }
```

```
+ }
>
> # Apply the function to assign strategies to each customer
> rfm_data$Marketing_Strategy <- sapply(rfm_data$Cluster, assign_strategy)
>
> # View the marketing strategies assigned to each customer
> head(rfm_data[, c("CustomerID", "Cluster", "Marketing_Strategy")])
```

Output:

	CustomerID	Cluster	Marketing_Strategy
1	1	3	Other
2	2	1	Other
3	3	2	Other
4	4	1	Other
5	5	4	Other
6	6	4	Other

Practical 10

Aim: Conduct A/B testing to evaluate the impact of different marketing strategies and make data-driven decisions.

- **Design and implement A/B tests for marketing campaigns using randomized assignment.**
- **Collect relevant data and perform statistical analysis tocompare the performance of different strategies.**
- **Calculate key metrics such as conversion rates, click-through rates, or revenue.**
- **Interpret the results and provide recommendations for optimizing marketing campaigns based on the findings.**

10.1 Design and implement A/B tests for marketing campaigns using randomized assignment.

 # Install and load necessary packages

```
install.packages(c("dplyr", "ggplot2"))

library(dplyr)

library(ggplot2)

# Set seed for reproducibility

set.seed(123)
```

 # Simulate data with conversion rates for control and test groups

```
control_data <- data.frame(user_id = 1:500, group = "Control")

test_data <- data.frame(user_id = 501:1000, group = "Test")
```

 # Introduce a difference in conversion rates (e.g., 5% for control and 8% for test)

```
control_data$conversion <- rbinom(500, 1, 0.05)

test_data$conversion <- rbinom(500, 1, 0.08)
```

 # Combine data

```
ab_test_data <- bind_rows(control_data, test_data)
```

 # Randomly shuffle the rows to assign users to control and test groups

```
ab_test_data <- ab_test_data[sample(nrow(ab_test_data)), ]
```

 # Calculate conversion rates for control and test groups

```
conversion_rates <- ab_test_data %>%
  group_by(group) %>%
  summarise(conversion_rate = mean(conversion))
```

 # Print conversion rates

```
print(conversion_rates)
```

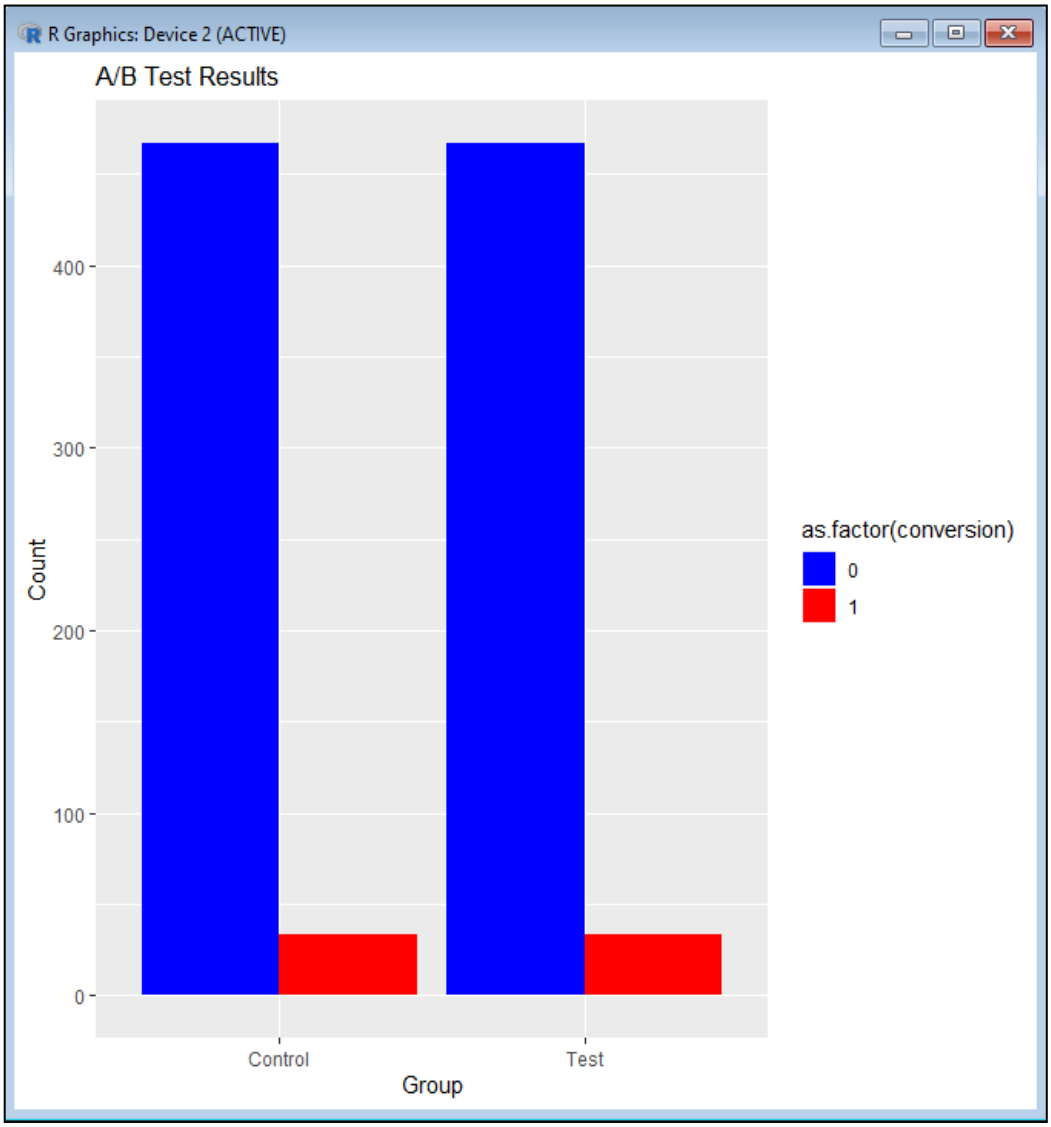
Output:

```
# A tibble: 2 × 2
  group   conversion_rate
  <chr>             <dbl>
1 Control           0.066
2 Test              0.066
```

 # Visualize results

```
ggplot(ab_test_data, aes(x = group, fill = as.factor(conversion))) +
  geom_bar(position = "dodge") +
  labs(title = "A/B Test Results", x = "Group", y = "Count") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
```

Output:



10.2 Collect relevant data and perform statistical analysis to compare the performance of different strategies in R

 # Set seed for reproducibility

```
set.seed(123)
```

 # Simulate data

```
strategy_data <- data.frame(
  strategy = rep(c("A", "B"), each = 500),
  engagement = rbinom(1000, 1, c(0.05, 0.08))
)
```

summary(strategy_data)

Output:

```
  strategy          engagement
Length:1000        Min.   :0.000
Class :character   1st Qu.:0.000
Mode  :character   Median :0.000
                   Mean   :0.059
                   3rd Qu.:0.000
                   Max.   :1.000
```

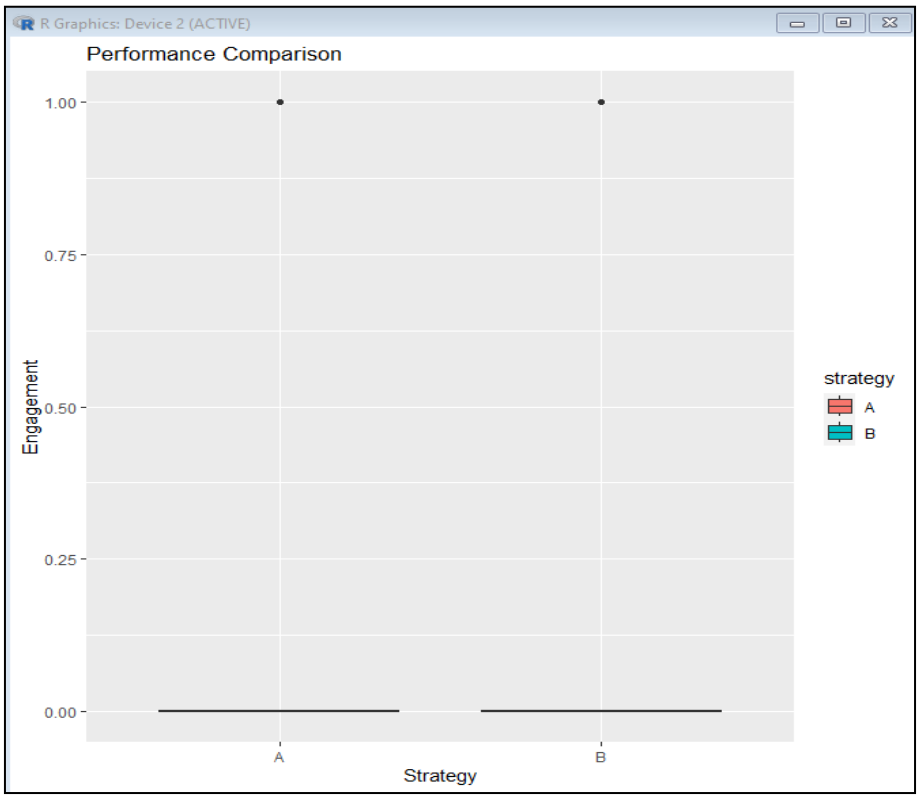
library(ggplot2)

ggplot(strategy_data, aes(x = strategy, y = engagement, fill = strategy)) +

geom_boxplot() +

labs(title = "Performance Comparison", x = "Strategy", y = "Engagement")

Output:



 # Independent samples t-test

t_test_result <- t.test(engagement ~ strategy, data = strategy_data)

```
# Print t-test result

print(t_test_result)
```

Output:

```
        Welch Two Sample t-test

data:  engagement by strategy
t = 1.2075, df = 977.95, p-value = 0.2275
alternative hypothesis: true difference in means between group A and group B is not equal to 0
95 percent confidence interval:
 -0.01125192  0.04725192
sample estimates:
mean in group A mean in group B
          0.068           0.050
```

```
# Check if the p-value is less than 0.05

if (t_test_result$p.value < 0.05) {

  print("The difference in engagement is statistically significant.")

} else {

  print("There is no statistically significant difference in engagement.")

}
```

Output:

```
[1] "There is no statistically significant difference in engagement."
>
```

10.3 Calculate key metrics such as conversion rates, click- through rates, or revenue.

Code:

```
# Create a hypothetical dataset
```

```
set.seed(123)
data <- data.frame(
  user_id = 1:1000,
  strategy = rep(c("A", "B"), each = 500),
  clicks = rbinom(1000, 1, c(0.2, 0.3)),  # Simulated click data
  conversions = rbinom(1000, 1, c(0.05, 0.08))  # Simulated conversion data
)

# Calculate click-through rate (CTR)
data$ctr <- data$clicks / sum(data$clicks)

# Calculate conversion rate (CR)
data$cr <- data$conversions / sum(data$conversions)

# Calculate revenue (assuming some hypothetical revenue value)
revenue_per_conversion <- 50  # Set your revenue per conversion value
data$revenue <- data$conversions * revenue_per_conversion

# Print summary statistics
summary(data)
```

Output:

```
    user_id         strategy             clicks      conversions
 Min.   :   1.0   Length:1000        Min.   :0.00   Min.   :0.000
 1st Qu.: 250.8   Class :character   1st Qu.:0.00   1st Qu.:0.000
 Median : 500.5   Mode  :character   Median :0.00   Median :0.000
 Mean   : 500.5                      Mean   :0.25   Mean   :0.064
 3rd Qu.: 750.2                      3rd Qu.:0.25   3rd Qu.:0.000
 Max.   :1000.0                      Max.   :1.00   Max.   :1.000
      ctr               cr              revenue
 Min.   :0.000   Min.   :0.00000   Min.   : 0.0
```

10.4. Interpret the results and provide recommendations for optimizing marketing campaigns based on the findings.

First, let's create sample data for campaign performance metrics for illustration purposes:

Code:

```
> # Simulated campaign performance data for each strategy
> set.seed(123)
> campaign_performance <- data.frame(
+ Marketing_Strategy = c("Re-Engagement Campaign", "Loyalty Rewards Program", "VIP or Exclusive Membership"),
+  Customers = c(500, 700, 300),
+  Average_Purchase = c(3.5, 5.2, 8.9),
+  Repeat_Purchase_Rate = c(25, 40, 60)  # Percentage values
+ )
>
> # View the simulated campaign performance data
> print(campaign_performance)
```

```
           Marketing_Strategy Customers Average_Purchase Repeat_Purchase_Rate
1      Re-Engagement Campaign       500              3.5                   25
2     Loyalty Rewards Program       700              5.2                   40
3 VIP or Exclusive Membership       300              8.9                   60
```

```
> # Plotting the simulated campaign performance metrics
> library(ggplot2)
> ggplot(campaign_performance, aes(x = Marketing_Strategy)) +
+  geom_bar(aes(y = Customers), stat = "identity", fill = "skyblue", alpha = 0.8) +
```

```
+  geom_line(aes(y = Average_Purchase * 10, group = 1), color = "red", size = 1.5) +
+  geom_point(aes(y = Average_Purchase * 10, group = 1), color = "red", size = 3) +
+  geom_line(aes(y = Repeat_Purchase_Rate * 5, group = 1), color = "green", size = 1.5) +
+  geom_point(aes(y = Repeat_Purchase_Rate * 5, group = 1), color = "green", size = 3) +
+  labs(
+    title = "Campaign Performance Metrics by Strategy",
+    y = "Count / Metric Value",
+    x = "Marketing Strategy"
+  ) +
+ scale_y_continuous(sec.axis = sec_axis(~./10, name = "Average Purchase / 10\nRepeat Purchase Rate / 5")) +
+  theme_minimal()
```

Output: