RFM, K-Means and Hierarchical Clustering

Recency, Frequency, and Monetary (RFM) Model

Objective

To review the gym usage on campus.

Questions: 1. What are the average frequency and duration of gym visit across gym visitors? 2. How can RFM model help to segment and profile our gym visitors? 3. How has the profile of the gym users change over time? 4. Explore other attributes associated to different gym visitor segments.

Load the required libraries.

```
# Load libraries
pacman::p_load(readxl, lubridate, tidyverse, ggplot2, treemapify, rfm, caret, cluster, factoextr
a, plotly)
```

Load and Inspect the data.

```
# Read in the whole gym dataset
df <- read_excel("...gym_staff.xlsx") # To paste file path here

# Check the dataset structure
str(df)
```

```
## tibble [7,312 × 11] (S3: tbl_df/tbl/data.frame)
##  $ ID      : chr [1:7312] "kGO6" "3eqG" "We51" "GLqr" ...
##  $ Venue   : chr [1:7312] "Wellness Outreach Gym" "University Town - Fitness gym" "University Town - Fitness gym" "Wellness Outreach Gym" ...
##  $ Passtype: chr [1:7312] "Per-Entry Ticket (Fitness Gym)" "Fitness Gym Membership" "Fitness Gym Membership" "Per-Entry Ticket (Fitness Gym)" ...
##  $ Date    : POSIXct[1:7312], format: "2022-01-02" "2022-01-02" ...
##  $ Checkin : POSIXct[1:7312], format: "1899-12-31 17:39:35" "1899-12-31 07:17:57" ...
##  $ Checkout: POSIXct[1:7312], format: "1899-12-31 18:52:57" "1899-12-31 08:31:24" ...
##  $ Domain  : chr [1:7312] "Academic" "Academic" "Academic" "Academic" ...
##  $ ULU     : chr [1:7312] "Others" "School of Computing" "College of Design and Engineering" "Yong Loo Lin Sch of Medicine" ...
##  $ FDLU    : chr [1:7312] "Dean's Office (Law)" "Department of Computer Science" "Mechanical Engineering" "Microbiology And Immunology" ...
##  $ Gender  : chr [1:7312] "M" "M" "M" "M" ...
##  $ Age     : chr [1:7312] "40-49" "40-49" "<30" "40-49" ...
```

```
# Check summary
summary(df)
```

```
##       ID               Venue             Passtype        
##  Length:7312        Length:7312        Length:7312       
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
##       Date                           Checkin                     
##  Min.   :2022-01-02 00:00:00.00   Min.   :1899-12-31 06:56:21.00  
##  1st Qu.:2022-02-23 00:00:00.00   1st Qu.:1899-12-31 12:06:51.50  
##  Median :2022-04-06 00:00:00.00   Median :1899-12-31 15:58:48.00  
##  Mean   :2022-04-06 05:06:49.63   Mean   :1899-12-31 15:08:30.61  
##  3rd Qu.:2022-05-20 00:00:00.00   3rd Qu.:1899-12-31 17:59:26.50  
##  Max.   :2022-06-30 00:00:00.00   Max.   :1899-12-31 21:45:51.00  
##     Checkout                         Domain               ULU           
##  Min.   :1899-12-31 07:44:52.00   Length:7312        Length:7312       
##  1st Qu.:1899-12-31 13:08:31.25   Class :character   Class :character  
##  Median :1899-12-31 17:04:44.50   Mode  :character   Mode  :character  
##  Mean   :1899-12-31 16:18:21.82                                        
##  3rd Qu.:1899-12-31 19:03:05.50                                        
##  Max.   :1899-12-31 22:12:00.00                                        
##      FDLU              Gender              Age           
##  Length:7312        Length:7312        Length:7312       
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
```

```
#change variables in character format to factor
df <- df %>% mutate_if(is.character,as.factor)

#transform POSIXct object to date format (ymd means year, month, day).
df$Date <- ymd(df$Date)
```

Data Cleaning

```
## Check for missing values
sum(is.na(df))
```

```
## [1] 0
```

```
## Return rows with missing values in any variable
df[rowSums(is.na(df)) > 0, ]
```

```
## # A tibble: 0 × 11
## # ℹ 11 variables: ID <fct>, Venue <fct>, Passtype <fct>, Date <date>,
## #   Checkin <dttm>, Checkout <dttm>, Domain <fct>, ULU <fct>, FDLU <fct>,
## #   Gender <fct>, Age <fct>
```

```
## Check for duplicated rows
sum(duplicated(df))
```

```
## [1] 0
```

(Latency) What is the typical time span between gym visits?

Number of visits per customer

First, we will find how frequent our customers visit the gym. To do so, we will need to re-arrange our dataset to count the number of visits per customer.

```
# Group the rows by customers and count them.
df_visits <- df %>%
  group_by(ID) %>%
  mutate(Visit=n()) %>%
  distinct(ID, .keep_all = TRUE)

#Show the descriptive statistics for the number of visits
summary(df_visits$Visit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    3.25   10.00   18.19   27.00  141.00
```

```
summary(df_visits)
```

```
##        ID                                  Venue    
##  05Pj   :  1   Kent Ridge - Fitness gym @MPSH3: 15  
##  0d2j   :  1   University Sports Centre - Gym : 79  
##  0lXl   :  1   University Town - Fitness gym  : 73  
##  0n6e   :  1   Wellness Outreach Gym          :235  
##  0p2JO  :  1                                        
##  0p2QN  :  1                                        
##  (Other):396                                        
##                           Passtype        Date           
##  Fitness Gym Membership        : 57   Min.   :2022-01-02  
##  Per-Entry Ticket (Fitness Gym):345   1st Qu.:2022-01-07  
##                                       Median :2022-02-08  
##                                       Mean   :2022-02-27  
##                                       3rd Qu.:2022-04-11  
##                                       Max.   :2022-06-30  
##                                                           
##     Checkin                          Checkout                     
##  Min.   :1899-12-31 07:06:50.00   Min.   :1899-12-31 07:44:52.00  
##  1st Qu.:1899-12-31 12:11:53.00   1st Qu.:1899-12-31 13:07:06.75  
##  Median :1899-12-31 15:57:02.50   Median :1899-12-31 17:01:32.50  
##  Mean   :1899-12-31 15:10:48.44   Mean   :1899-12-31 16:13:40.76  
##  3rd Qu.:1899-12-31 17:48:17.25   3rd Qu.:1899-12-31 18:58:41.75  
##  Max.   :1899-12-31 21:45:01.00   Max.   :1899-12-31 21:55:00.00  
##                                                                   
##                      Domain                                   ULU     
##  Academic               :305   Arts & Social Sciences          : 17  
##  Administration         : 25   College of Design and Engineering:120  
##  Corporate              : 13   Engineering                      :  1  
##  Innovation & Enterprise:  1   Others                           :161  
##  Research               : 58   School of Computing              : 16  
##                                Science                          : 46  
##                                Yong Loo Lin Sch of Medicine     : 41  
##                                   FDLU     Gender     Age          Visit       
##  Electrical And Computer Engineering: 32   F:100   <30  :148   Min.   :  1.00  
##  Civil And Environmental Engineering: 23   M:302   >=60 :  8   1st Qu.:  3.25  
##  Research                           : 19           30-39:170   Median : 10.00  
##  Mechanical Engineering             : 17           40-49: 53   Mean   : 18.19  
##  Chemistry                          : 16           50-59: 23   3rd Qu.: 27.00  
##  Biomedical Engineering             : 15                       Max.   :141.00  
##  (Other)                            :280                                        
```

What are the number of days between visits to the

gym?

```
# Similarly, we will arrange the ID, group the ID, calculate the date difference between the row
s and add the results into a new column "counter".
df_days<- df %>%
  group_by(ID) %>%
  mutate(Date_Diff = as.numeric(difftime(Date, lag(Date), units = "days"))) %>%
  mutate(counter=seq_along(ID))%>%
  arrange(ID)

view(df_days)

# Select the required variables
df_days <- df_days %>%
  select(ID, counter, Date_Diff)
```

(Latency) What is the average number of days between gym visits?

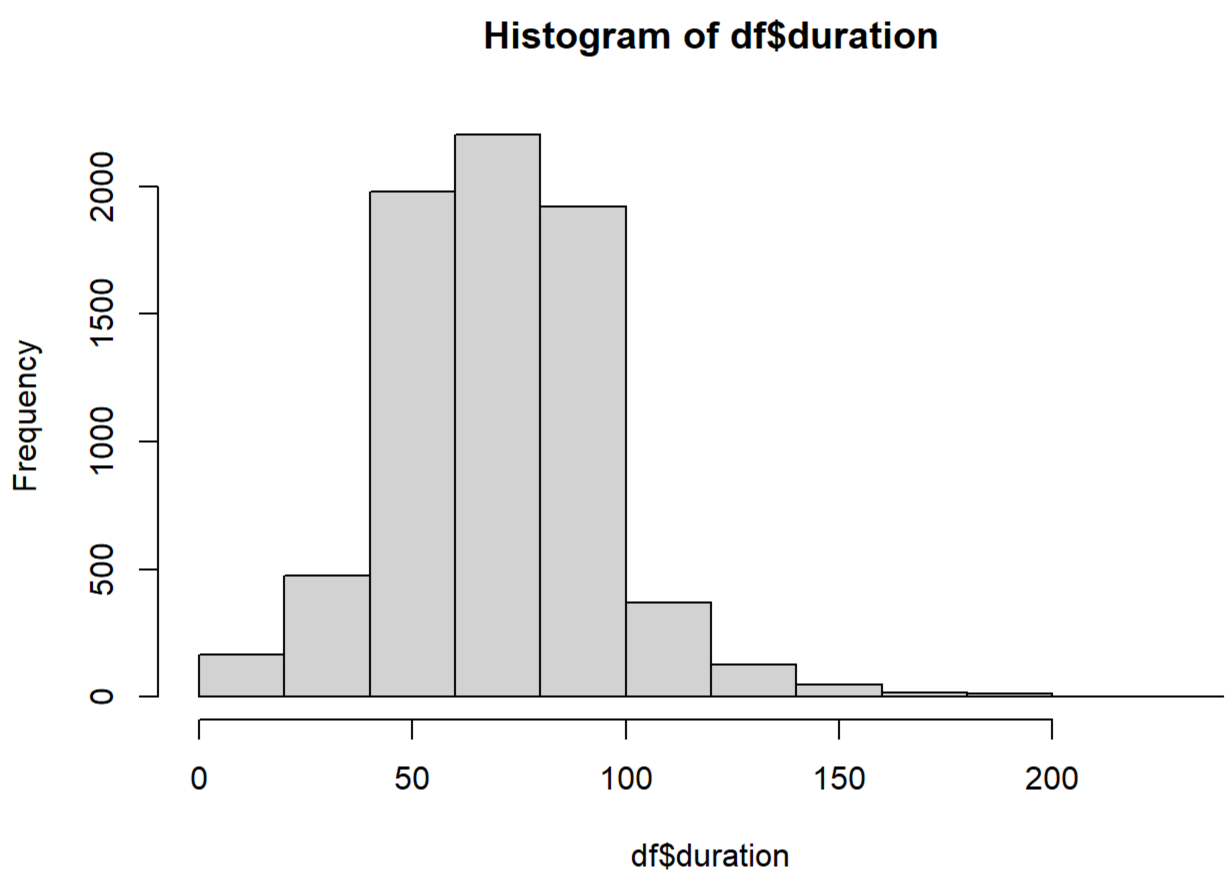
```
# Pivot the dataframe into a table showing the days between each visits
# Vertical to Horizontal
latency <- pivot_wider(df_days, names_from =counter, values_from = Date_Diff)

# Remove ID to rownames
latency <- as.data.frame(latency)
rownames(latency) <- latency[,"ID"]
latency$`ID` <- NULL

# We can choose to remove column "1", since it represents the first visit
latency$`1` <- NULL

# Calculate the means, median and number of unique gym visitors.'na.rm = TRUE' means remove NA.
lmean <-round(colMeans(latency, na.rm = TRUE), digit=2)
# apply(array, 1 -> row, 2 -> column)
lmedian <- apply(latency,2,median, na.rm = TRUE)
lcount <- apply(latency, 2, function(x) sum(!is.na(x)))

# (rbind/rowbind) Bind the lmean and lmedian into the latency dataset
lsummary <- rbind(lcount, lmean, lmedian)
rownames(lsummary) <- c("Count", "Mean","Median")
```

What is the duration for each visit?

```
# Calculate the gym duration
df$duration <- difftime(df$Checkout, df$Checkin, units = "mins")

# Convert the variable into numeric mode
df$duration <- round(as.numeric(df$duration),2)

# Check the descriptive statistics for duration
summary(df$duration) #variable must be in numeric mode
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.08   53.30   69.16   69.85   85.92  223.97
```

```
# Visualise the data
hist(df$duration) #variable must be in numeric mode
```



(Customer Lifetime Value) Which generates more revenue: single entry or membership?

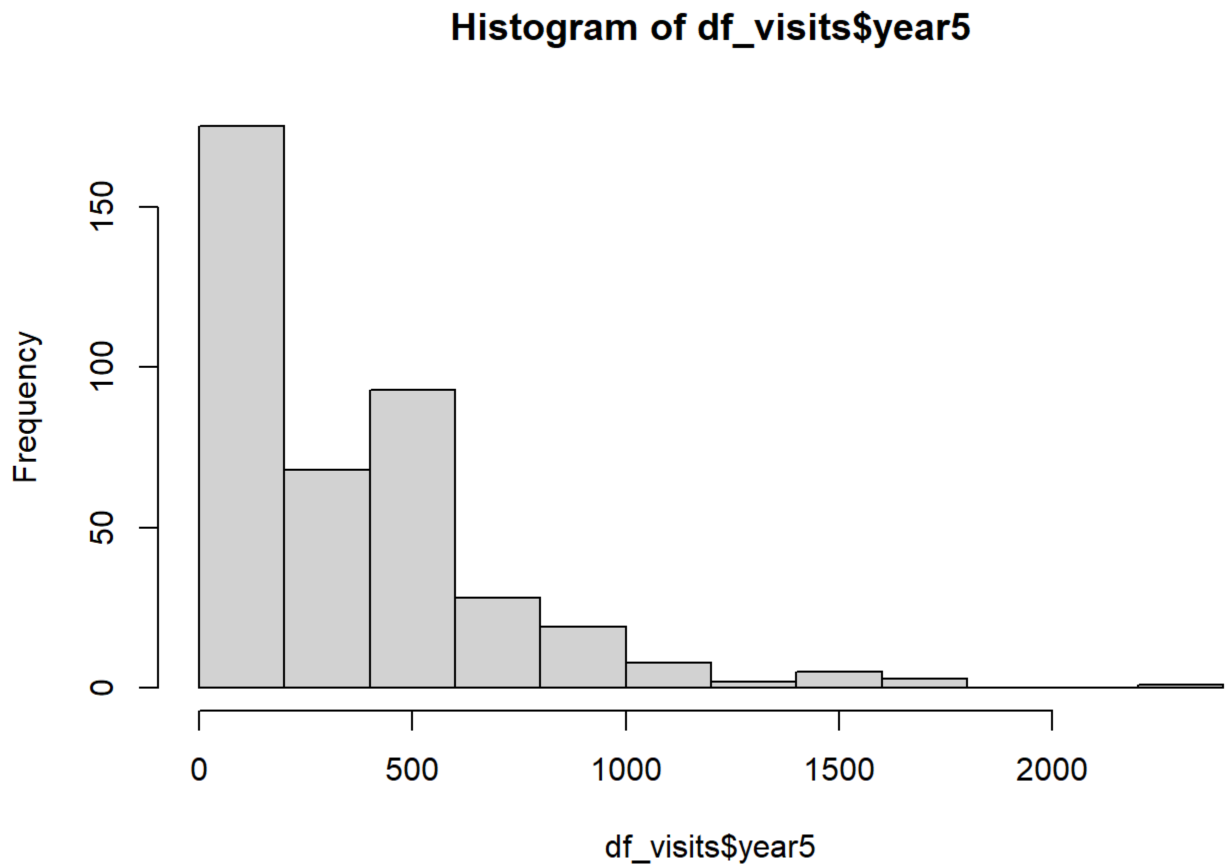
Assume that these are the yearly and per entry fees: Gym Membership (Yearly) - $100 Gym Individual (Single Entry) - $4.50

How much revenue will the gym earn?

```
# Which one is more popular? Single entry or membership?
summary(as.factor(df_visits$Passtype))
```

```
##         Fitness Gym Membership Per-Entry Ticket (Fitness Gym)
##                             57                            345
```

```
# (grepl) Look for the text and calculate the cost and add to column "Membership".
df_visits$Membership <- ifelse(grepl("Fitness Gym Membership", df_visits$Passtype)==TRUE, 100,
0)

# (grepl) Look for the text and calculate the cost and add to column "Single"..
df_visits$Single <- ifelse(grepl("Per-Entry Ticket", df_visits$Passtype)==TRUE, df_visits$Visit*
4.50, 0)

#How much revenue does the gym earn for Membership and Single?
Revenue_membership <- sum(df_visits$Membership)
Revenue_single <- sum(df_visits$Single)
Total_revenue <- Revenue_membership + Revenue_single

# Suppose the users were to visit the same number of times for the next 5 years, the expected CL
V will be.
df_visits$year5 <- ifelse(df_visits$Membership==100, df_visits$Membership* 5, df_visits$Single*
5)

# Visualise the CLV distribution
hist(df_visits$year5)
```



This is evident that the membership will provide the gym with a stable income as compared to the single entry users.

Customer Lifetime Value) Identify potential members.

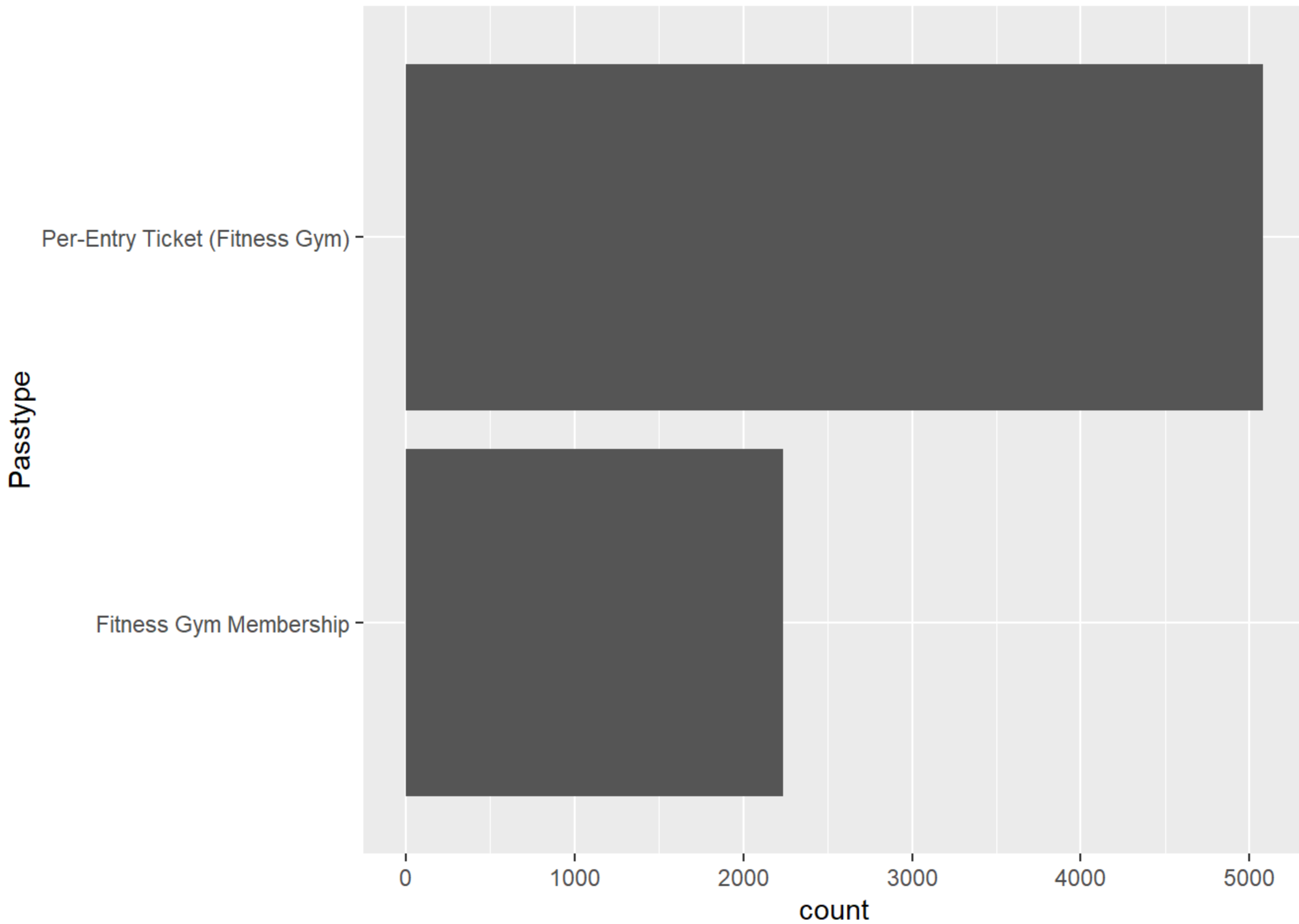
Customer analytics do not end here. The questions now is can we entice more users to sign up for membership? Who are paying more and how much can they save if they were to sign up for membership?

```
# Filter potential members based on spending more than $100 (membership fees)
Potential_member <- df_visits %>%
  filter(Single>100)

View(Potential_member)

Potential_member$Savings <- Potential_member$Single - 100
```

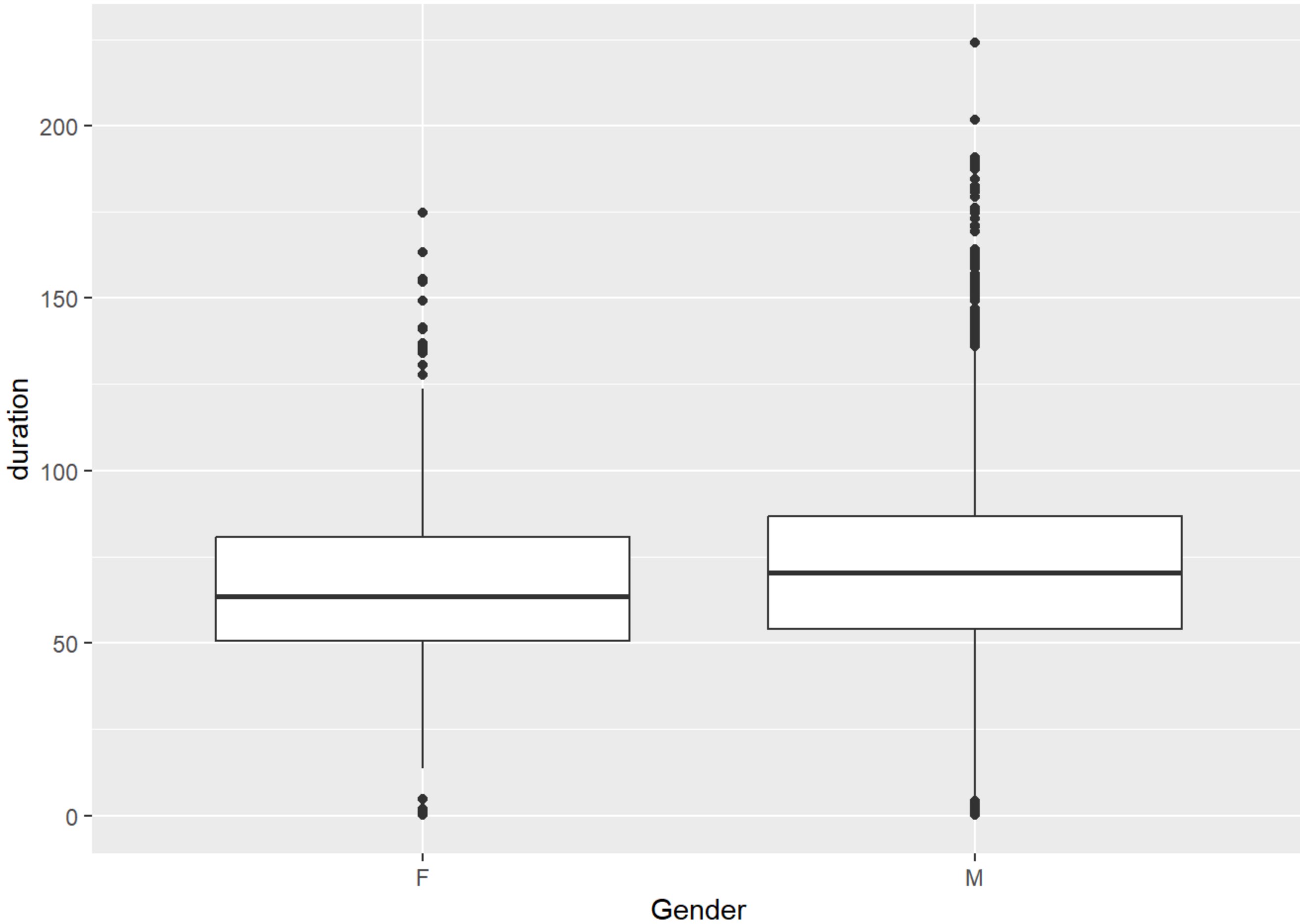
Exploratory Data Analysis - Data Visualisation

EDA is a process that can be used to summarise main characteristics, identify interesting patterns, and identify outliers.

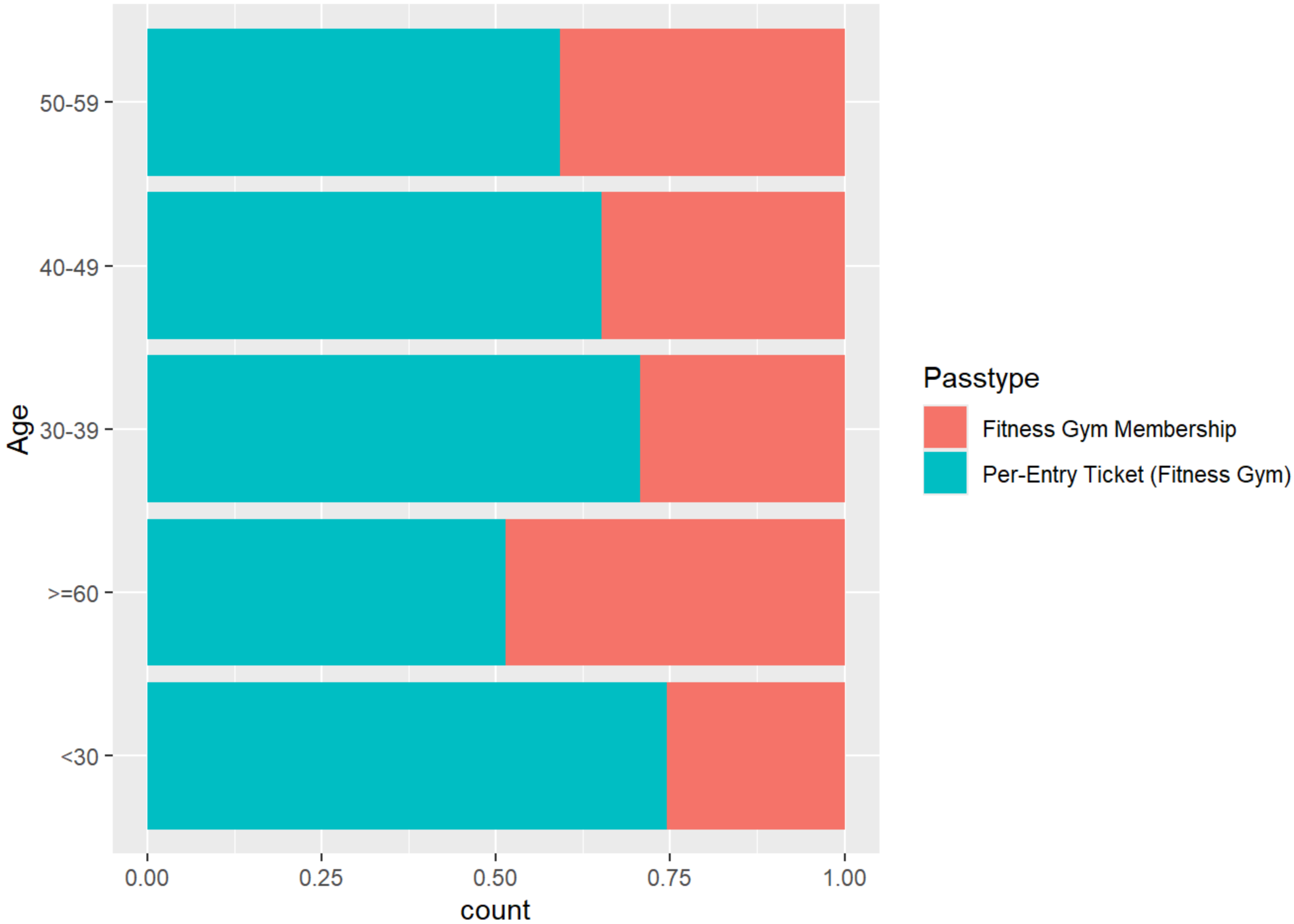
```
# EDA - Type of passes
ggplot(df) + geom_bar(aes(y=Passtype))
```



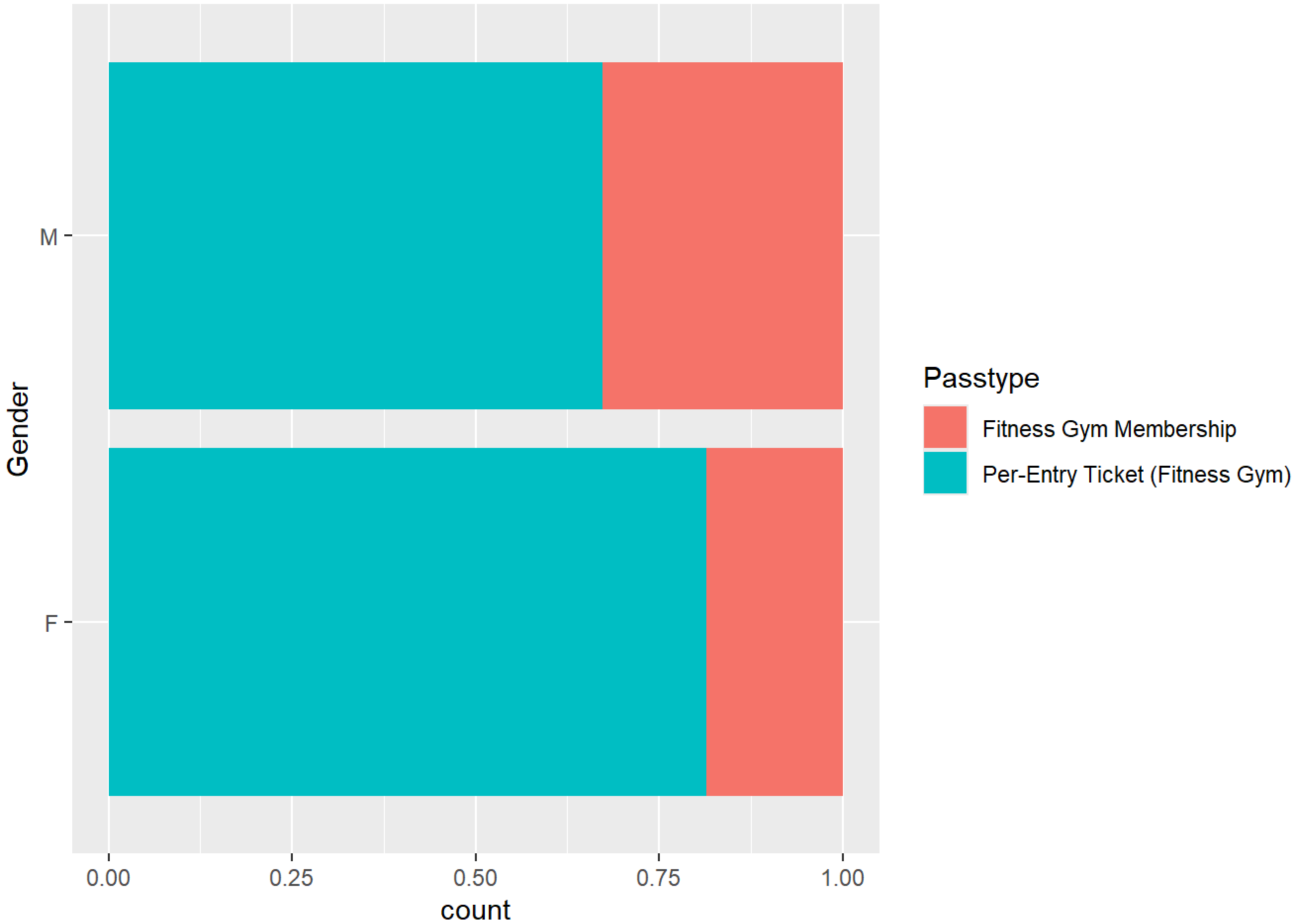

```
# EDA - Gender vs duration
ggplot(df) + geom_boxplot(aes(x=Gender, y=duration))
```



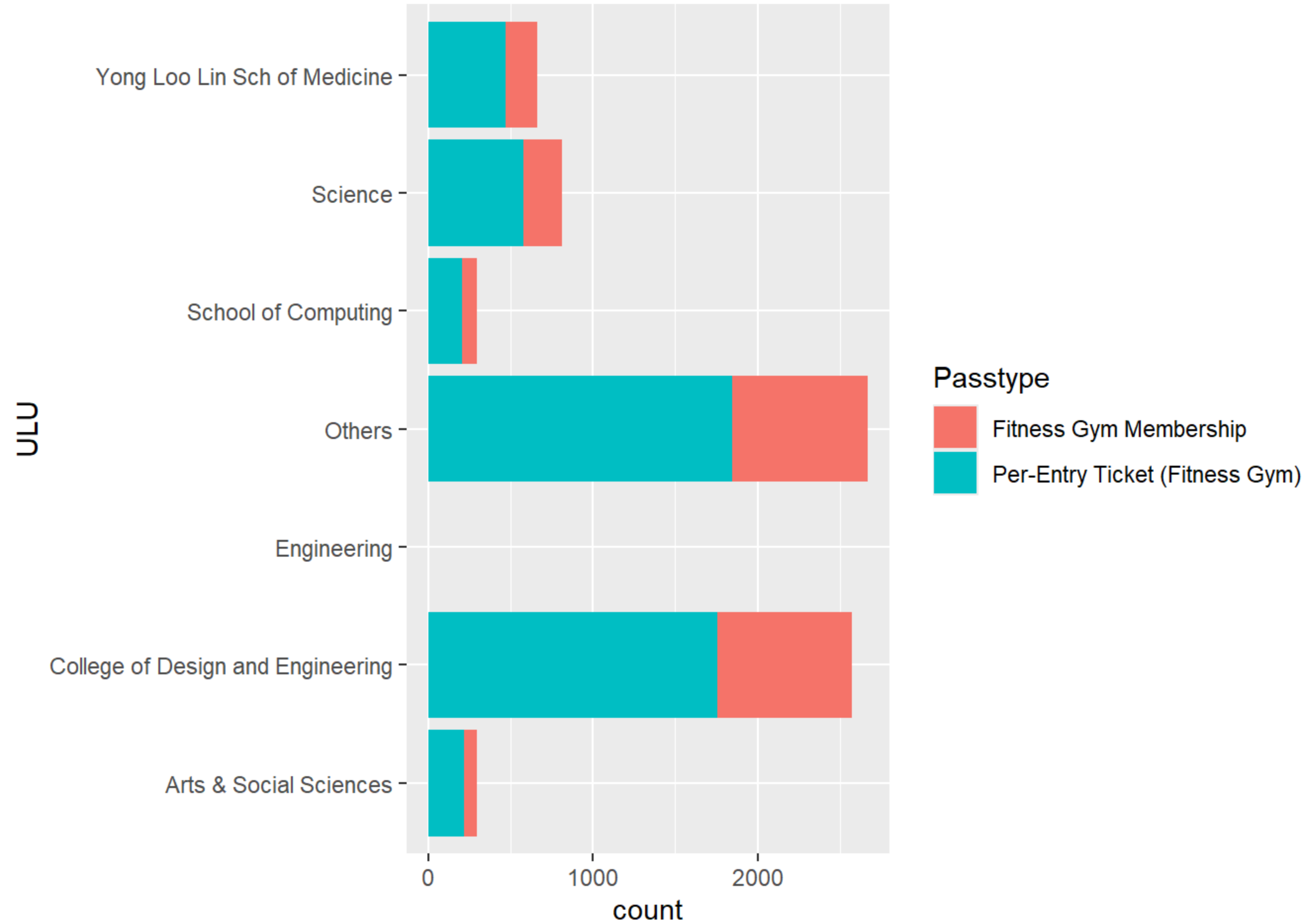

```
# EDA - Age vs Passtype
ggplot(df) + geom_bar(aes(y=Age, fill=Passtype), position = "fill")
```




```
## EDA - Gender vs Passtype
ggplot(df) + geom_bar(aes(y=Gender, fill=Passtype), position = "fill")
```




```
## EDA - ULU vs Passtype
ggplot(df, aes(y=ULU, fill=Passtype)) + geom_bar(aes(y=ULU, fill=Passtype))
```



What are the average frequency and duration of gym visit across gym visitors?

Descriptive Statistic

```
# Calculate the gym duration which is the difference between Checkin and Checkout time
df$Duration <- as.numeric(difftime(df$Checkout, df$Checkin, units = "mins"))

# Create a dataframe for the number of visits and average duration
df_visits <- df %>%
  select(ID, Duration) %>%  #select only these two variables, ID & Duration
  group_by(ID) %>%
  summarise(Visit=n(), Average_Duration = mean(Duration)) # calculate the number of visit and av
erage duration per visit for each gym visitor

# Descriptive statistic for the number of visits and average duration.
summary(df_visits$Average_Duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   9.317  53.742  63.844  65.711  77.824 135.917
```

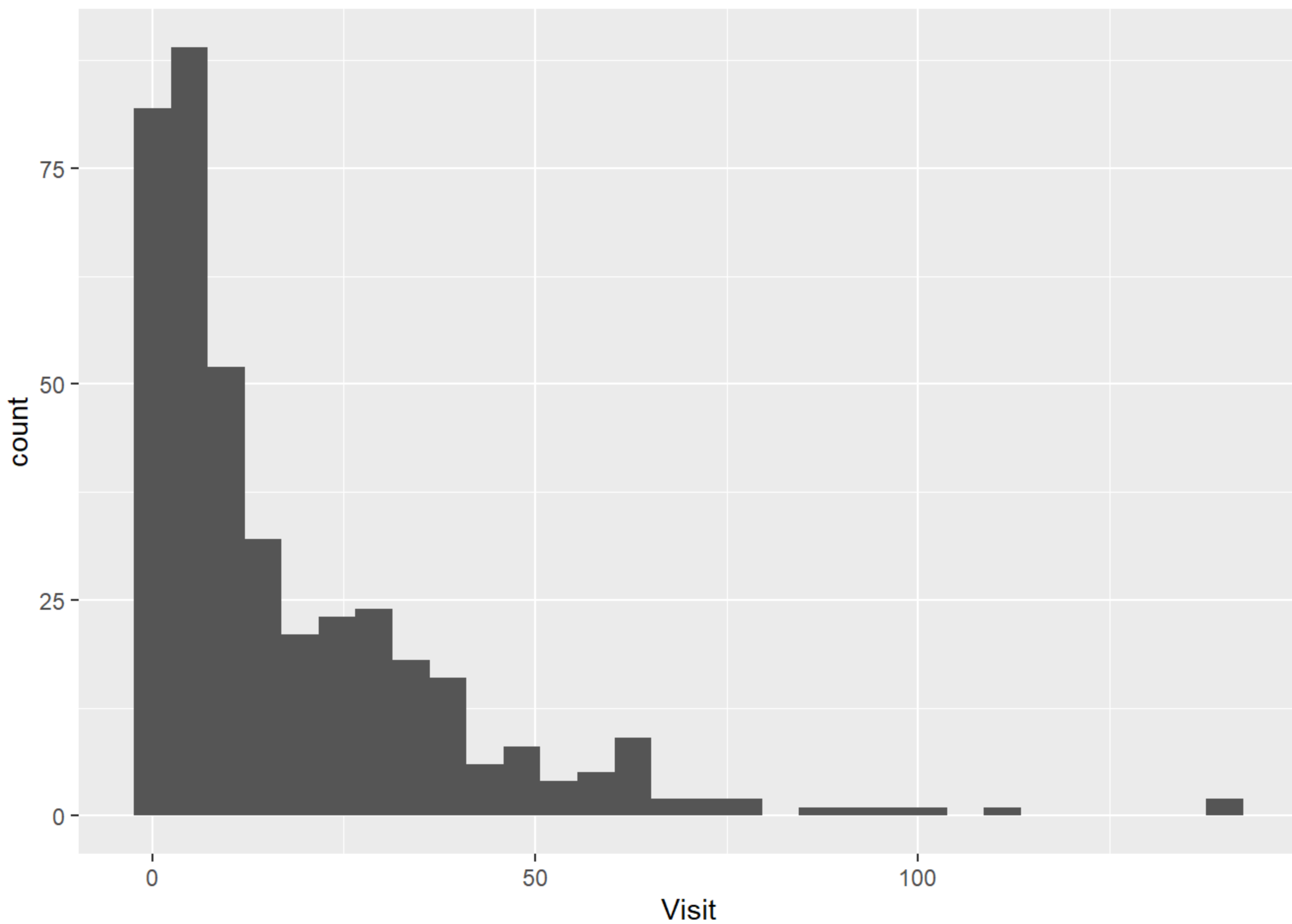
```
summary(df_visits$Visit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.25   10.00   18.19   27.00  141.00
```

Data Visualistion

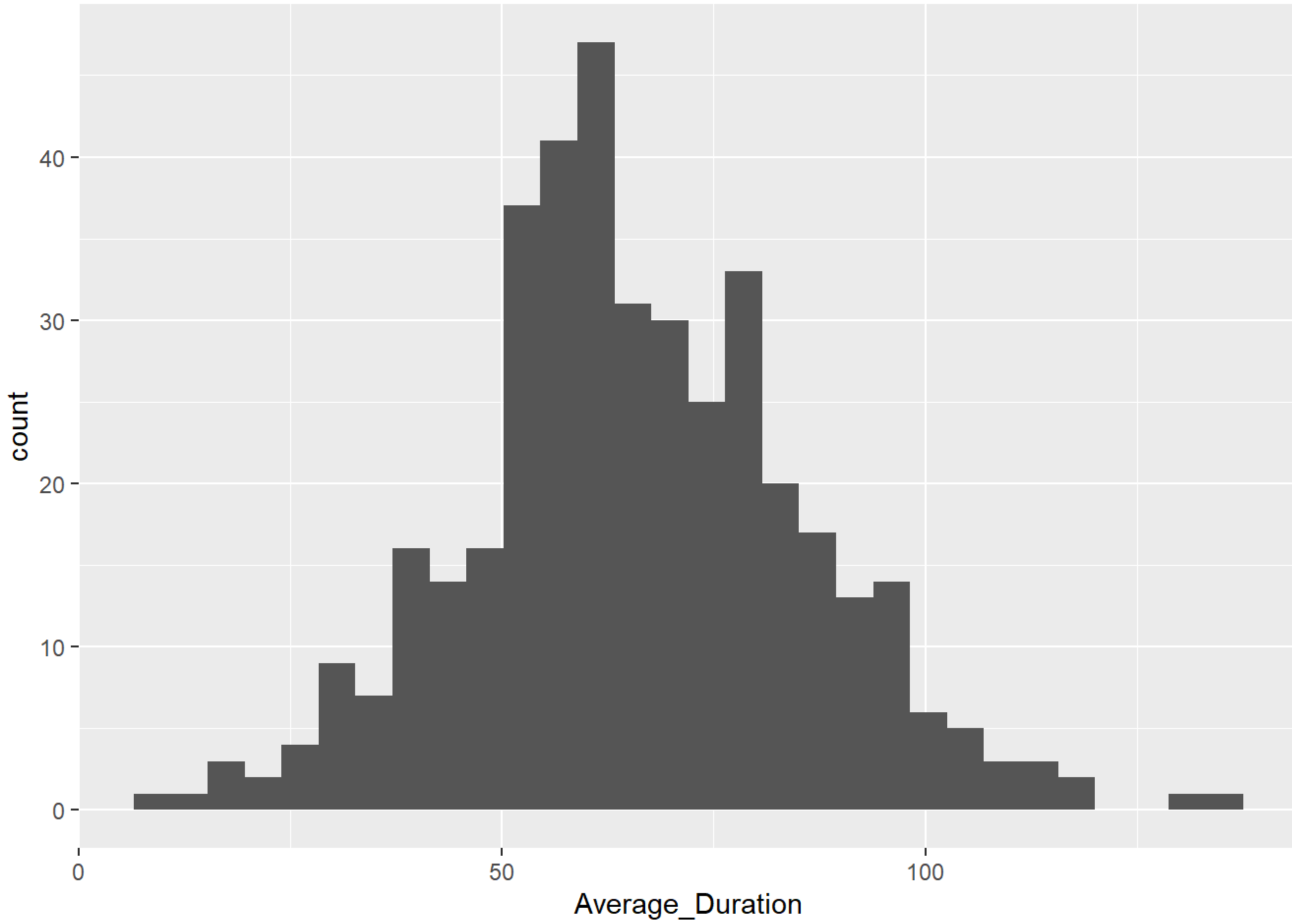
```
# Create histogram plots
## ggplot() initialises the ggplot object and it is used to declare the input data frame, geom_h
istogram returns a layer that contains a histogram
ggplot(df_visits) + geom_histogram(aes(x=Visit))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



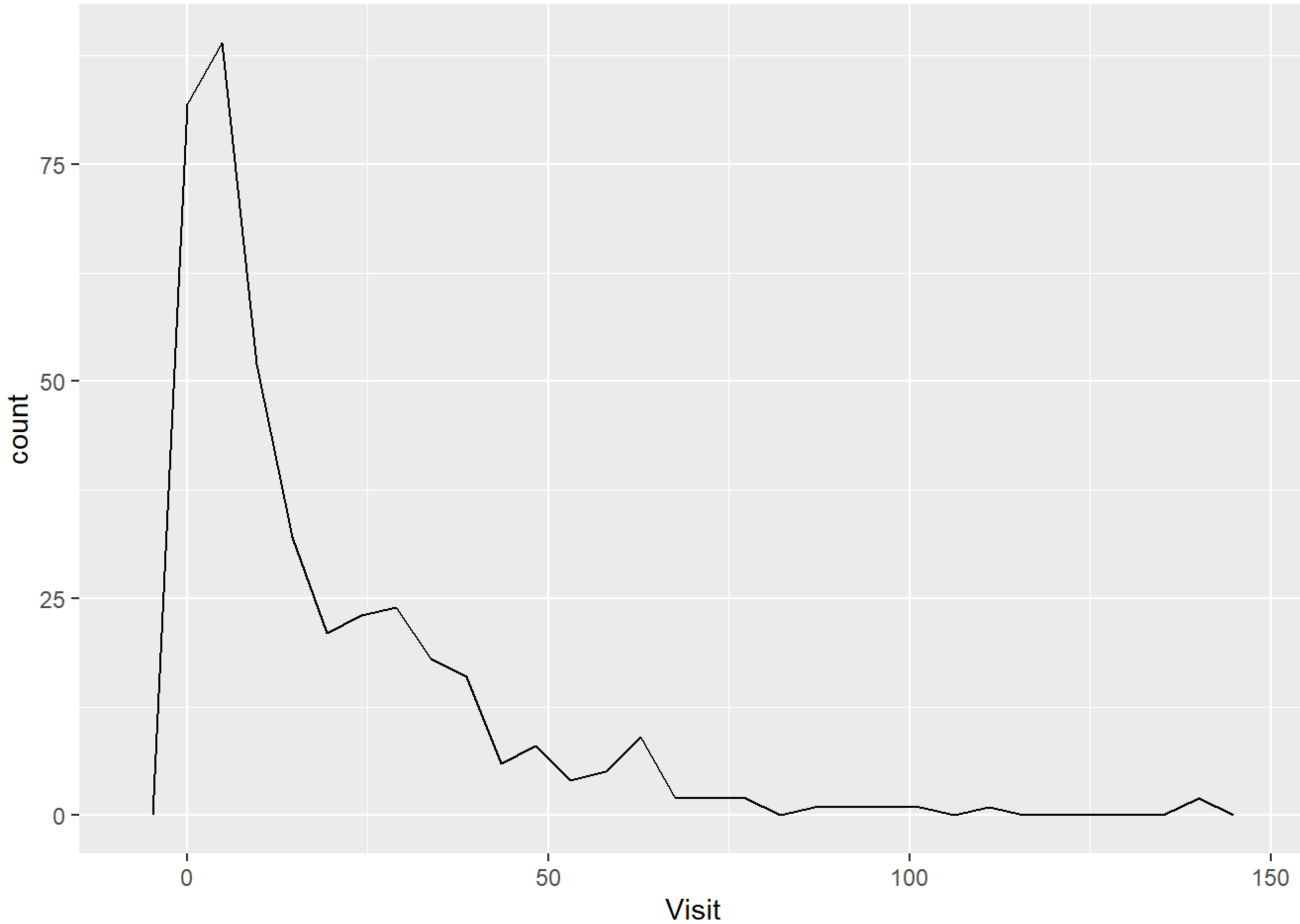

```
ggplot(df_visits) + geom_histogram(aes(x=Average_Duration))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



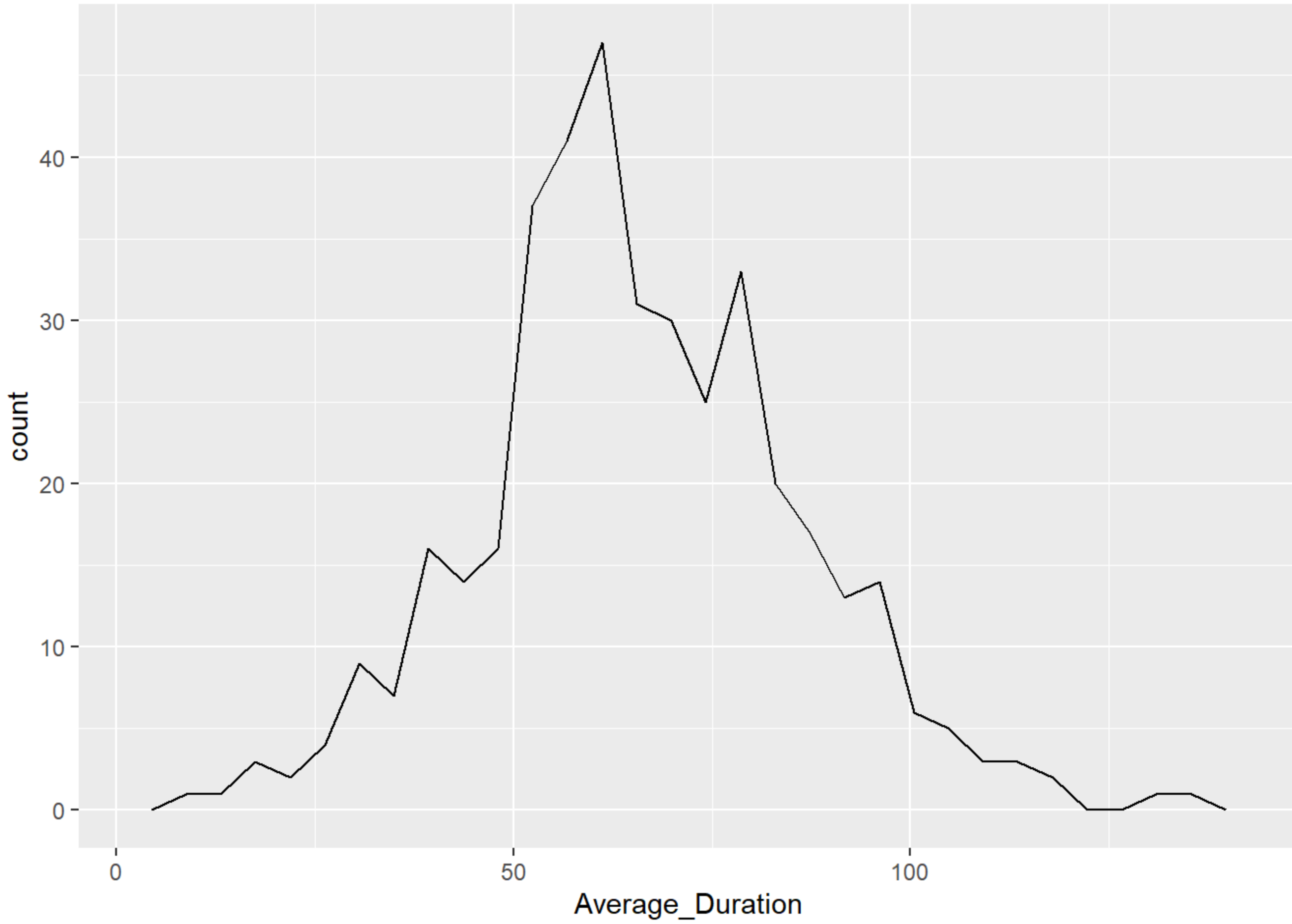

```
## replace geom_histogram with geom_freqpoly.
ggplot(df_visits) + geom_freqpoly(aes(x=Visit))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(df_visits) + geom_freqpoly(aes(x=Average_Duration))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Insights: 1. Visits taper off over number of visits 2. Average Duration faces normal distribution.

How can we use RFM model to segment and profile our gym visitors?

RFM Analysis

```
# Determine the analysis date
analysis_date <- lubridate::as_date('2022-06-30')

# Create a new column customer_id
df$customer_id <- df$ID

# Perform RFM analysis
rfm_result <- rfm_table_order(df,customer_id, Date, Duration, analysis_date)

str(rfm_result)
```

```
## Classes 'rfm_table_order', 'tibble' and 'data.frame':    0 obs. of  6 variables:
##  $ rfm           : tibble [7,312 × 19] (S3: tbl_df/tbl/data.frame)
##   ..$ customer_id      : Factor w/ 402 levels "05Pj","0d2j",..: 1 1 1 1 1 2 2 2 2 2 ...
##   ..$ recency_days     : num  1 1 1 1 1 0 0 0 0 0 ...
##   ..$ transaction_count: int  5 5 5 5 5 7 7 7 7 7 ...
##   ..$ amount           : num  269 269 269 269 269 ...
##   ..$ recency_score    : int  5 5 5 5 5 5 5 5 5 5 ...
##   ..$ frequency_score  : int  2 2 2 2 2 2 2 2 2 2 ...
##   ..$ monetary_score   : int  2 2 2 2 2 3 3 3 3 3 ...
##   ..$ rfm_score        : num  522 522 522 522 522 523 523 523 523 523 ...
##   ..$ ID               : Factor w/ 402 levels "05Pj","0d2j",..: 1 1 1 1 1 2 2 2 2 2 ...
##   ..$ Venue            : Factor w/ 4 levels "Kent Ridge - Fitness gym @MPSH3",..: 2 2 2 2 2 4
4 4 4 4 ...
##   ..$ Passtype         : Factor w/ 2 levels "Fitness Gym Membership",..: 1 1 1 1 1 2 2 2 2 2
...
##   ..$ Checkin          : POSIXct, format: "1899-12-31 12:48:37" "1899-12-31 12:08:23" ...
##   ..$ Checkout         : POSIXct, format: "1899-12-31 13:40:23" "1899-12-31 13:04:57" ...
##   ..$ Domain           : Factor w/ 5 levels "Academic","Administration",..: 1 1 1 1 1 1 1 1 1
1 ...
##   ..$ ULU              : Factor w/ 7 levels "Arts & Social Sciences",..: 2 2 2 2 2 7 7 7 7 7
...
##   ..$ FDLU             : Factor w/ 117 levels "Academic Affairs",..: 53 53 53 53 53 42 42 42
42 42 ...
##   ..$ Gender           : Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ Age              : Factor w/ 5 levels "<30",">=60","30-39",..: 4 4 4 4 4 1 1 1 1 1 ...
##   ..$ duration         : num  51.8 56.6 62.9 45.7 51.6 ...
##  $ analysis_date : Date, format: "2022-06-30"
##  $ frequency_bins: num 5
##  $ recency_bins  : num 5
##  $ monetary_bins : num 5
##  $ threshold     :'data.frame':  5 obs. of  6 variables:
##   ..$ recency_lower  : num  0 2 9 25 84.4
##   ..$ recency_upper  : num  2 9 25 84.4 179
##   ..$ frequency_lower: num  1 3 8 15 32
##   ..$ frequency_upper: num  3 8 15 32 142
##   ..$ monetary_lower : num  9.32 152.66 416.42 961.99 2210.12
##   ..$ monetary_upper : num  153 416 962 2210 12967
```

```
# View(rfm_result$rfm)
```

Customer Segmentation

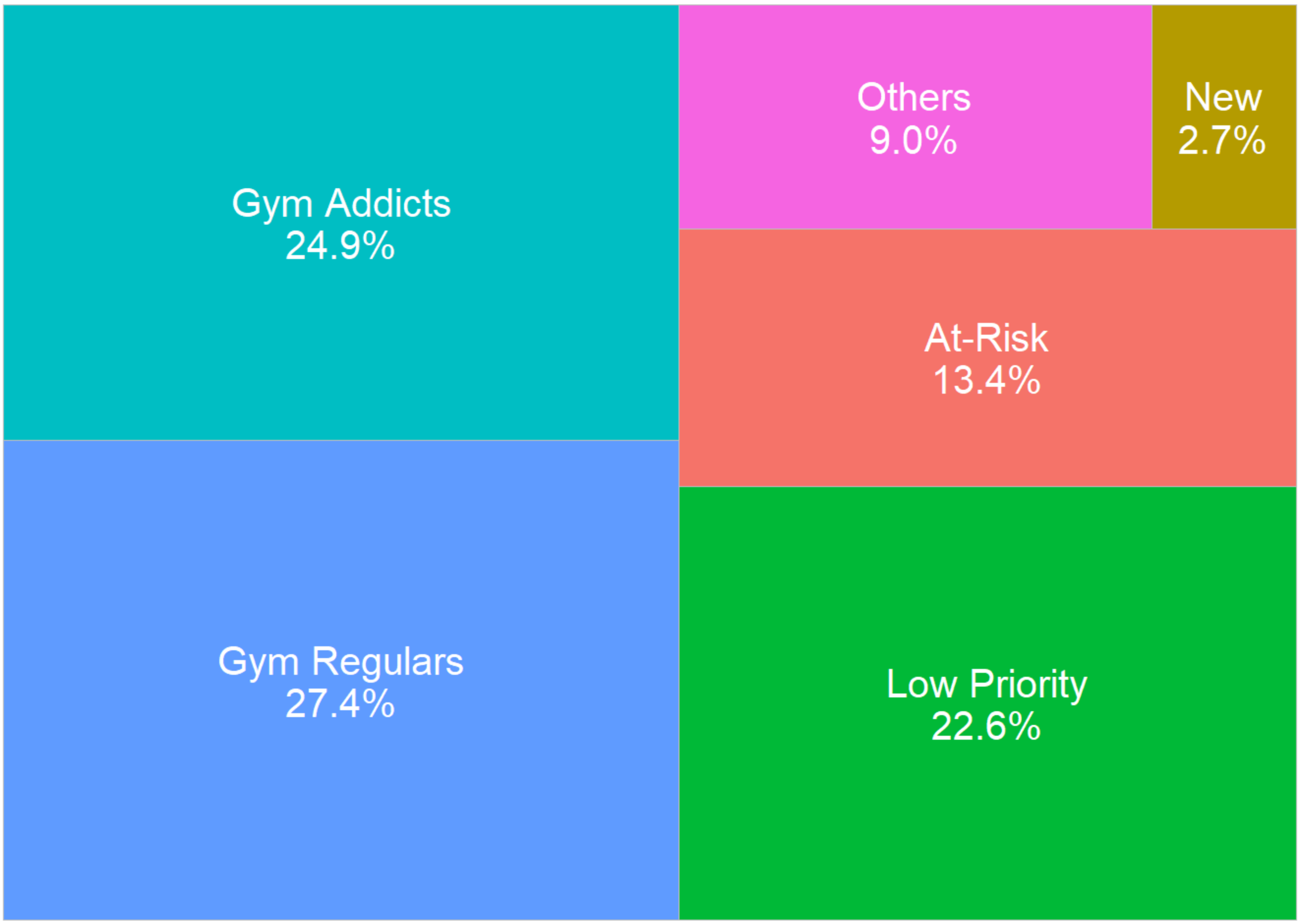
```
# Check threshold for each score
rfm_result$threshold
```

```
##   recency_lower recency_upper frequency_lower frequency_upper monetary_lower
## 1           0.0           2.0               1               3       9.316667
## 2           2.0           9.0               3               8     152.663333
## 3           9.0          25.0               8              15     416.423333
## 4          25.0          84.4              15              32     961.990000
## 5          84.4         179.0              32             142    2210.116667
##   monetary_upper
## 1       152.6633
## 2       416.4233
## 3       961.9900
## 4      2210.1167
## 5     12967.2667
```

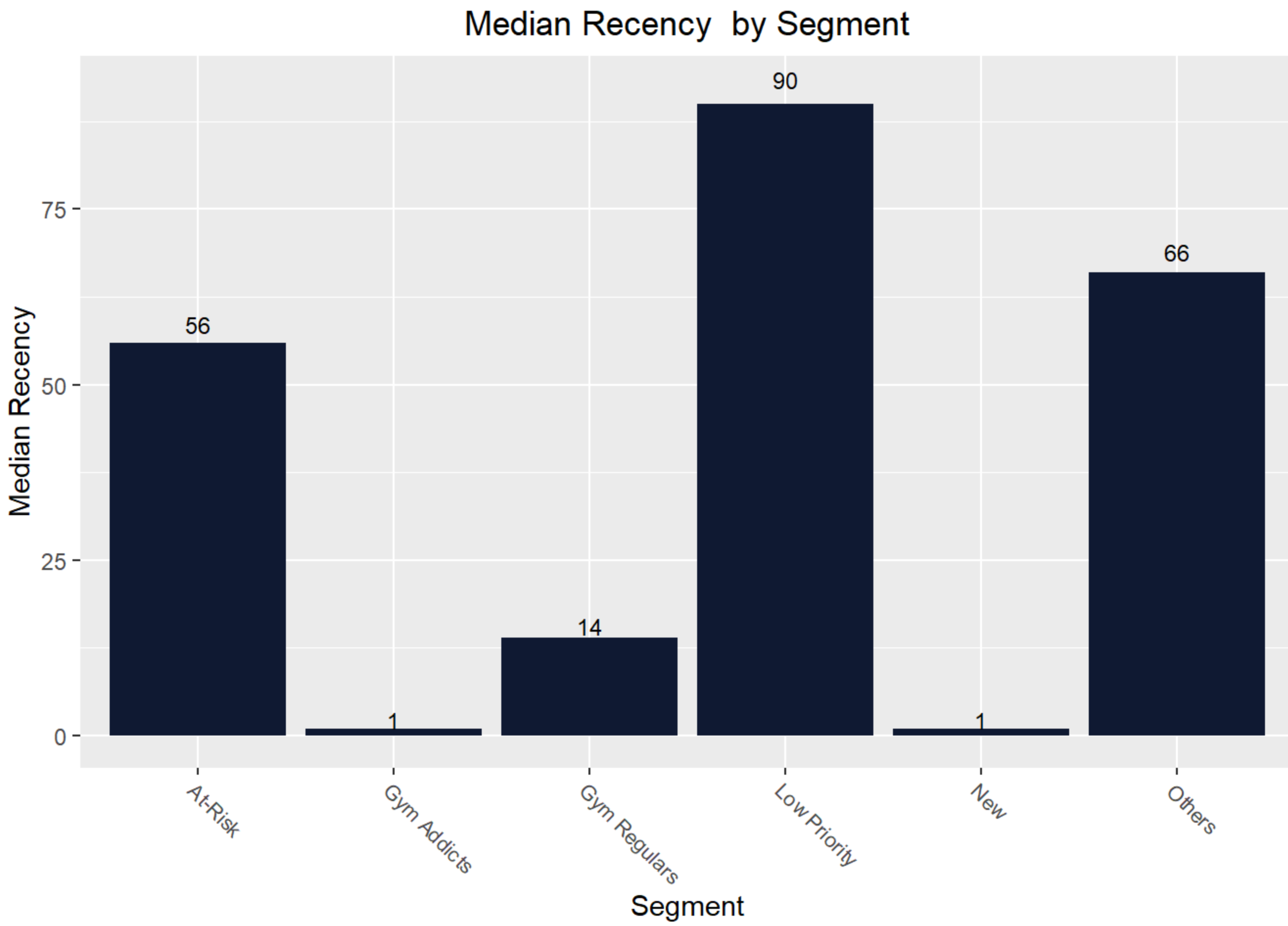
```
#Gym Addicts = super committed and hardworking
#Fat Burners = do not come regularly, but spend long duration to burn fat at every visit
#Gym Regulars = your average regular gym goers
#At-Risk = used to be frequent goers, but have not visited the gym recently
#Low Priority = rarely goes to the gym and does not spend much time there.
#New = newcomers
#Others = other gym visitors (no intention to focus on them at the moment)
segment_names <- c("Gym Addicts", "Fat Burners", "Gym Regulars", "At-Risk", "Low Priority", "Ne
w", "Others")

# Set the boundary for each of the scores
recency_lower <- c(4, 1, 3, 1, 1, 4, 1)
recency_upper <- c(5, 2, 5, 2, 2, 5, 5)
frequency_lower <- c(4, 1, 2, 3, 1, 1, 1)
frequency_upper <- c(5, 2, 5, 5, 2, 2, 5)
monetary_lower <- c(4, 4, 2, 3, 1, 1, 1)
monetary_upper <- c(5, 5, 5, 5, 2, 2, 5)

# Create segments based on recency, frequency and monetary scores.
segments <- rfm_segment(rfm_result, segment_names, recency_lower, recency_upper,
frequency_lower, frequency_upper, monetary_lower, monetary_upper)

# Create summary table of the different segments
segments_table <- segments %>%
  distinct(ID, .keep_all = TRUE) %>%
  group_by(segment) %>%
  summarise(Recency_mean = mean(recency_days), Frequency_mean = mean(transaction_count), Duratio
n_mean = mean(amount), Count = n()) %>%
  mutate(Proportion=Count/sum(Count),Percentage=scales::percent(Proportion))
```

Data Visualisation

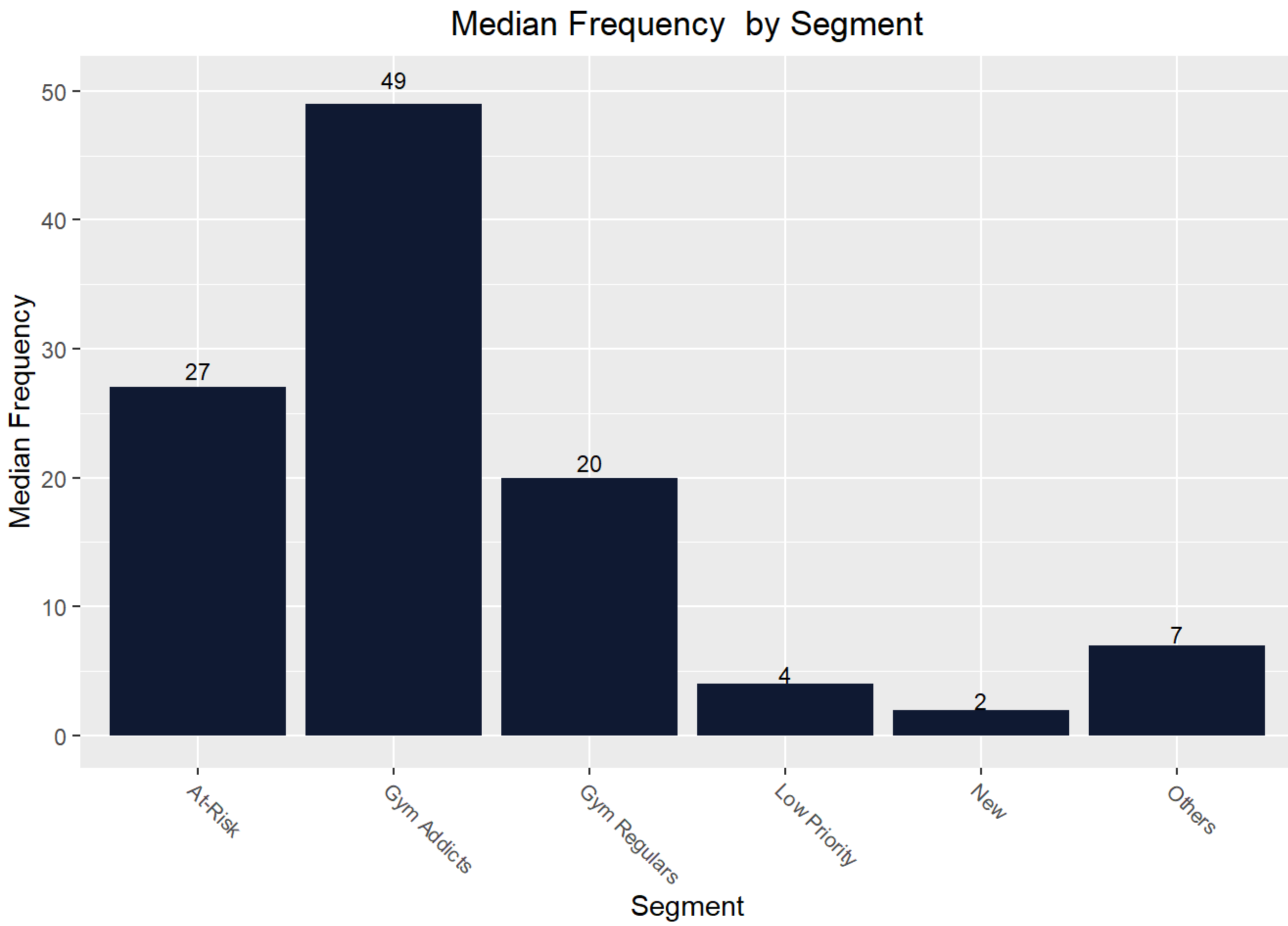
```
# Treemap
ggplot(segments_table, aes(area = Count, fill= Percentage, label = paste(segment,Percentage,sep
="\n"))) +
   geom_treemap() +
   geom_treemap_text(
     colour = "white",
     place = "centre",
     size = 15,
   ) + theme(legend.position="none")
```



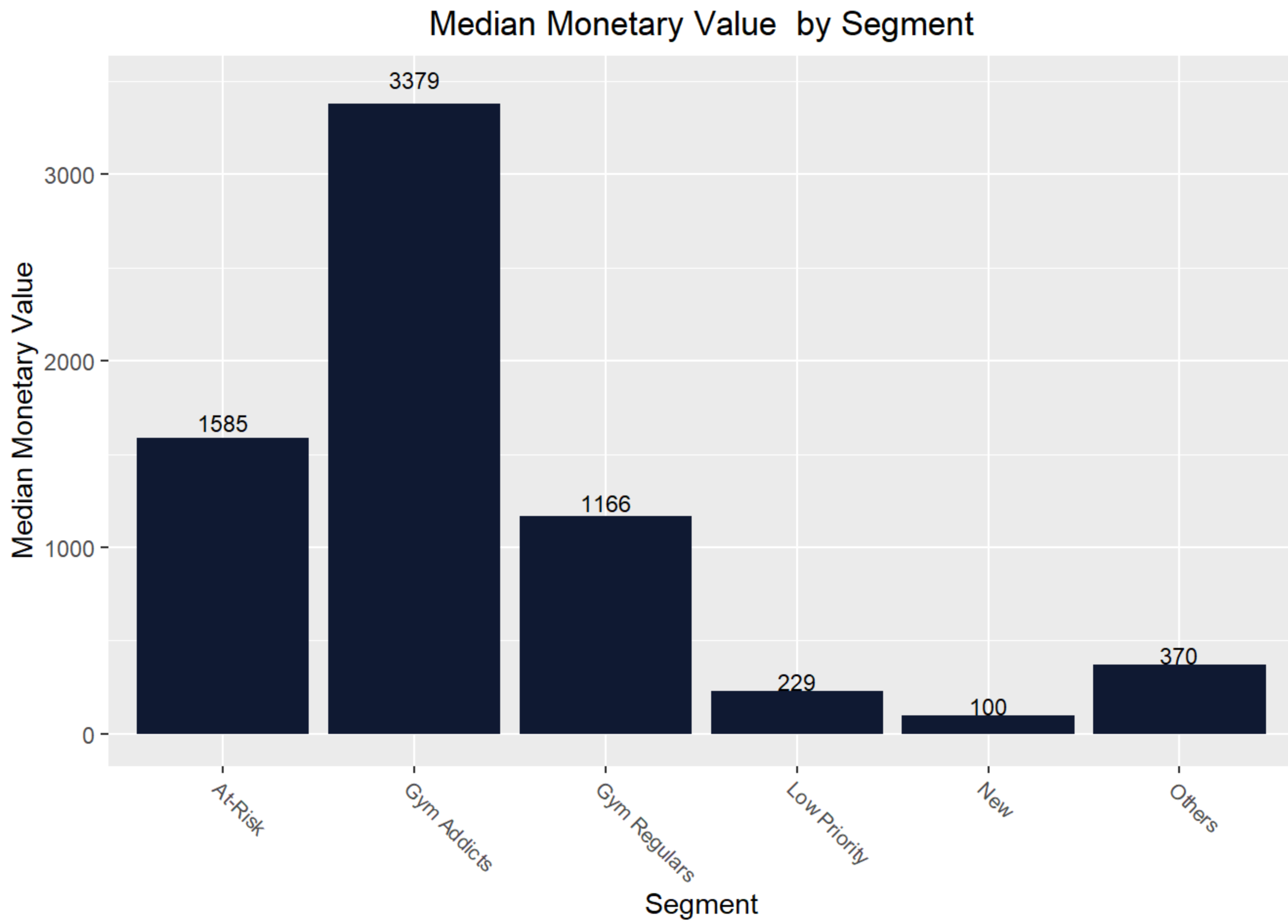

```
# The RFM package also includes several functions to plot charts
rfm_plot_median_recency(segments)
```




```
rfm_plot_median_frequency(segments)
```

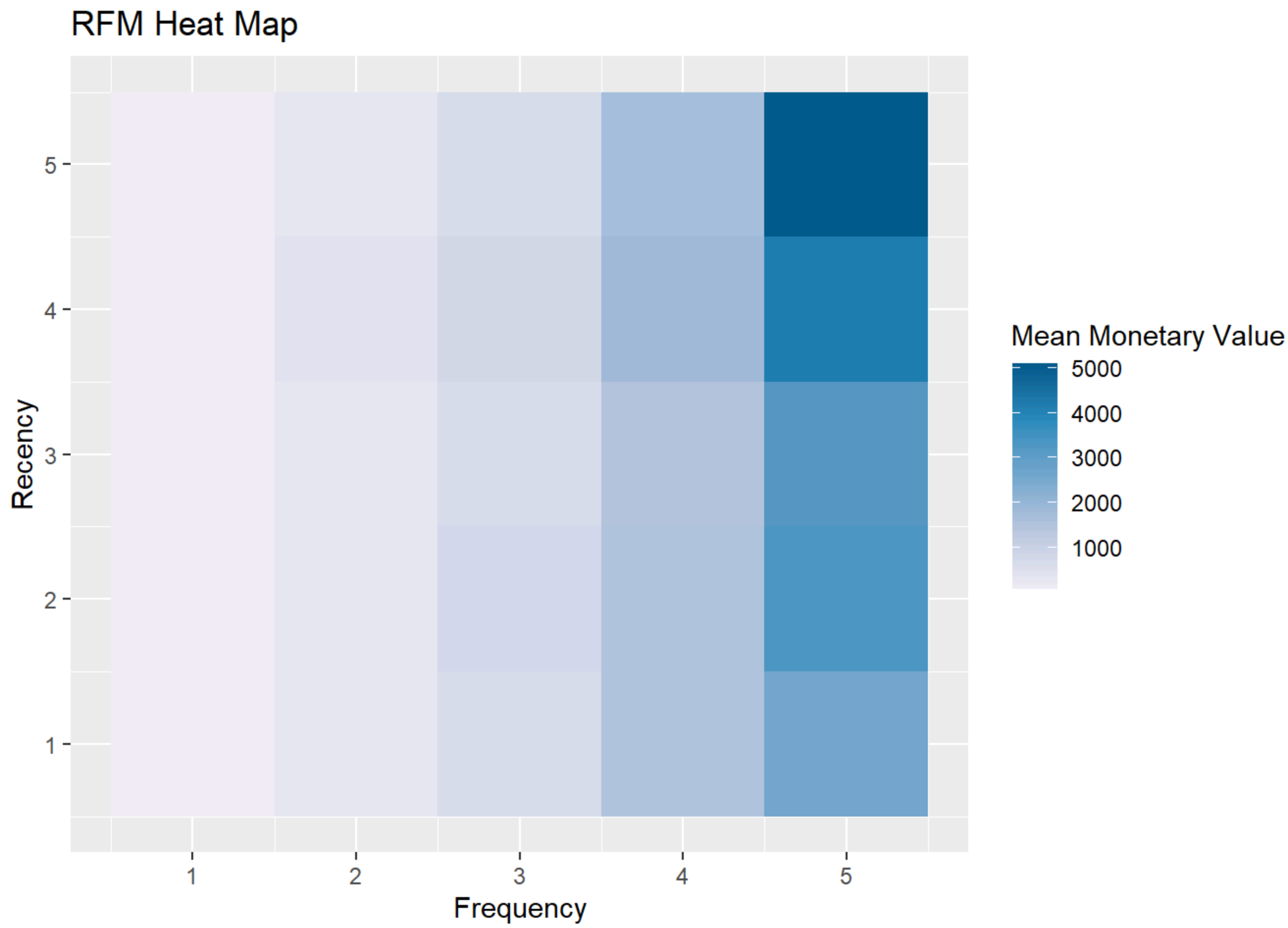



```
rfm_plot_median_monetary(segments)
```



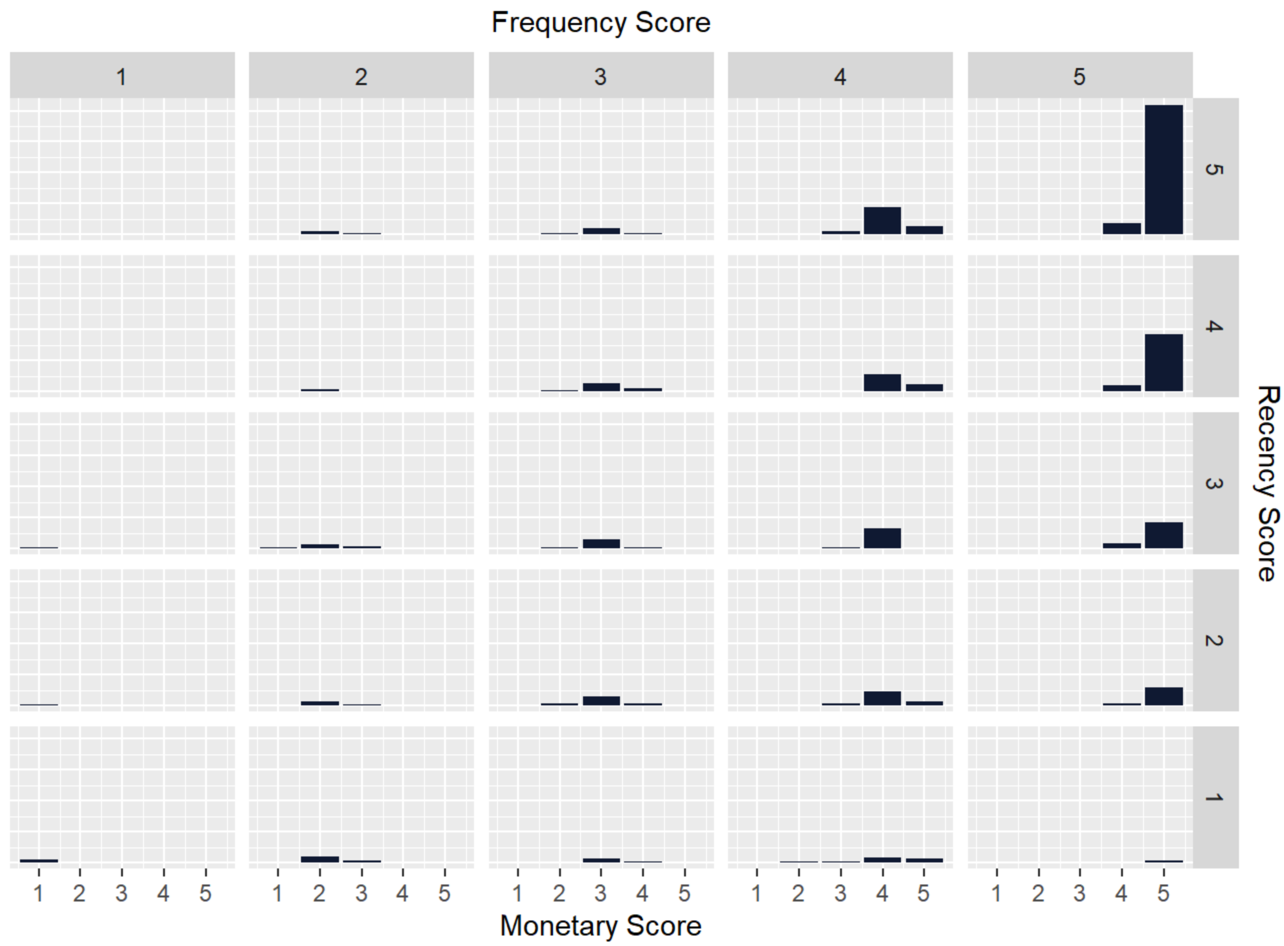

```
rfm_heatmap(rfm_result)
```

```
## Warning in rfm_heatmap(rfm_result): 'rfm_heatmap' is deprecated.
## Use 'rfm_plot_heatmap()' instead.
## See help("Deprecated")
```



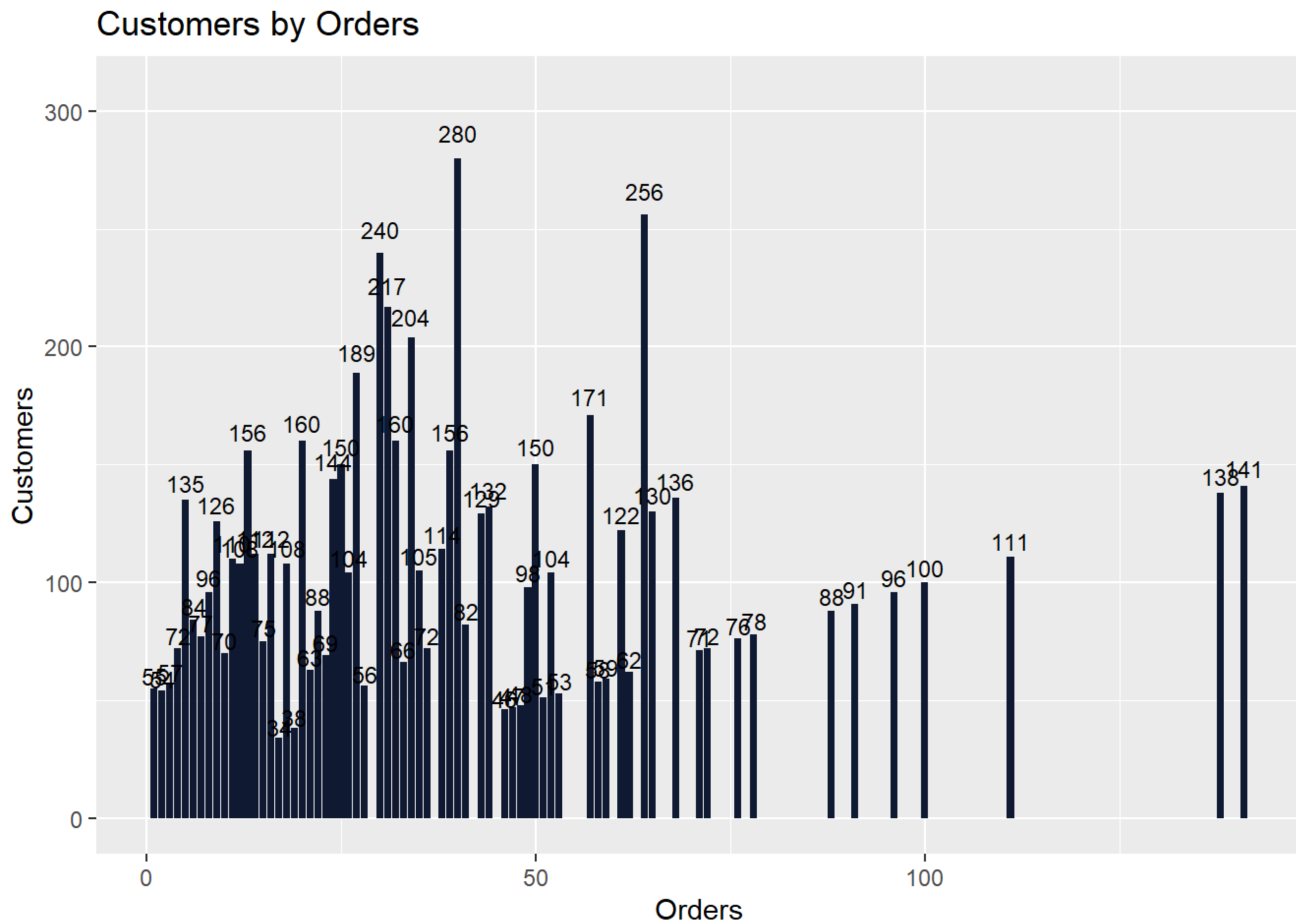

```
rfm_bar_chart(rfm_result)
```

```
## Warning in rfm_bar_chart(rfm_result): 'rfm_bar_chart' is deprecated.
## Use 'rfm_plot_bar_chart()' instead.
## See help("Deprecated")
```




```
rfm_order_dist(rfm_result)
```

```
## Warning in rfm_order_dist(rfm_result): 'rfm_order_dist' is deprecated.
## Use 'rfm_plot_order_dist()' instead.
## See help("Deprecated")
```



How does the profile of our gym users change over time?

For certain purposes, it may be more insightful to see the change in RFM score. Let us see the change in customer segments based on RFM score between Q1 2022 and Q2 2022.

Separate the dataset to Q1 2022 and Q2 2022

```
#  Mutate a new column to specify Q1 or Q2
df <- df %>% mutate (Quarter = case_when(Date < "2022-04-01" ~ "Q1 2022", Date < "2022-07-01" ~
"Q2 2022"))

## Filter data for Q1 and Q2
df_Q1 <- df %>% filter (Quarter== 'Q1 2022') #Filter data for Q1
df_Q2 <- df %>% filter(Quarter== "Q2 2022") #Filter data for Q1
```

Perform RFM analysis on Q1 2022

```
# Using Q1 2022 data

## Determine the analysis date
analysis_date <- lubridate::as_date('2022-03-31') #last day of Q1

## Perform RFM analysis
rfm_result_Q1 <- rfm_table_order(df_Q1,ID, Date, Duration, analysis_date)

# Customer segmentation
segments_Q1 <- rfm_segment(rfm_result_Q1, segment_names, recency_lower, recency_upper,
frequency_lower, frequency_upper, monetary_lower, monetary_upper)

# Customer Segmentation Table
segments_table_Q1 <- segments_Q1 %>%
  distinct(ID, .keep_all = TRUE) %>%
  group_by(segment) %>%
  summarise(Recency_mean = mean(recency_days), Frequency_mean = mean(transaction_count), Duratio
n_mean = mean(amount), Count = n()) %>%
  mutate(Proportion=Count/sum(Count))
```

Perform RFM analysis on Q2 2022

```
# Using Q2 2022 data

## Determine the analysis date
analysis_date <- lubridate::as_date('2022-06-30') #take last day of Q2

## Perform RFM analysis
rfm_result_Q2 <- rfm_table_order(df_Q2,ID, Date, Duration, analysis_date)

# Customer segmentation
segments_Q2 <- rfm_segment(rfm_result_Q2, segment_names, recency_lower, recency_upper,
frequency_lower, frequency_upper, monetary_lower, monetary_upper)

# Customer Segmentation Table
segments_table_Q2 <- segments_Q2 %>%
  distinct(ID, .keep_all = TRUE) %>%
  group_by(segment) %>%
  summarise(Recency_mean = mean(recency_days), Frequency_mean = mean(transaction_count), Duratio
n_mean = mean(amount), Count = n())  %>%
  mutate(Proportion=Count/sum(Count))

segments_table_Q1
```

```
## # A tibble: 6 × 6
##   segment      Recency_mean Frequency_mean Duration_mean Count Proportion
##   <chr>               <dbl>          <dbl>         <dbl> <int>      <dbl>
## 1 At-Risk            24.8           11.9            819.    37     0.130
## 2 Gym Addicts         0.986         27.0           1846.    72     0.253
## 3 Gym Regulars        3.26          10.4            652.    80     0.281
## 4 Low Priority       42.8            1.94           120.    69     0.242
## 5 New                 1.4            1.73           110.    15     0.0526
## 6 Others             24.8            3.33           212.    12     0.0421
```

```
segments_table_Q2
```

```
## # A tibble: 6 × 6
##   segment      Recency_mean Frequency_mean Duration_mean Count Proportion
##   <chr>               <dbl>          <dbl>         <dbl> <int>      <dbl>
## 1 At-Risk            33.1           13.9           972.     29     0.0868
## 2 Gym Addicts         0.978         24.7          1889.     91     0.272
## 3 Gym Regulars        6.75          10.3           709.     92     0.275
## 4 Low Priority       52.6            2.12          124.     82     0.246
## 5 New                 1              2              96.8     9     0.0269
## 6 Others             24.6            3.06          210.     31     0.0928
```

Evaluate the change

```
# Summarise the aggregate changes between Q1 and Q2.
quarter_change <- cbind(segments_table_Q2[1],segments_table_Q2[,-1] - segments_table_Q1[,-1])

# Explore individual changes  between Q1 and Q2.

## One approach that we could use to identify valuable customers who may need more encouragement
or personal follow up is to look at the change in the RFM score.  An increase in RFM score can b
e an indicator of increase in customer's attachment towards your business. On the other hand, a
decrease in RFM score can be an indicator of decrease in your customer's attachment.

## Using change in RFM identify the number of customers who were initially Gym Addicts in Q1 but
have their RFM score decreased more than 200 in Q2.



unique_segments_Q1 <- segments_Q1 %>%  distinct(ID, .keep_all = TRUE)
unique_segments_Q2 <- segments_Q2 %>%  distinct(ID, .keep_all = TRUE)

rfm_change <- full_join(unique_segments_Q1, unique_segments_Q2, by = "customer_id") %>% #merge t
he 2 quarter segments
select(c("customer_id","rfm_score.x","segment.x", "rfm_score.y","segment.y")) %>%
  rename(rfm_score_Q1 = rfm_score.x, segment_Q1 = segment.x, rfm_score_Q2 = rfm_score.y, segment
_Q2 = segment.y ) %>%
  mutate(change = rfm_score_Q2-rfm_score_Q1)

rfm_change %>% filter(segment_Q1 == "Gym Addicts", change < -200)
```

```
## # A tibble: 21 × 6
##    customer_id rfm_score_Q1 segment_Q1  rfm_score_Q2 segment_Q2   change
##    <fct>              <dbl> <chr>              <dbl> <chr>         <dbl>
##  1 0n6e                 555 Gym Addicts          323 Gym Regulars   -232
##  2 16JW                 555 Gym Addicts          344 Gym Regulars   -211
##  3 1ZWDP                554 Gym Addicts          343 Gym Regulars   -211
##  4 1ZzeP                555 Gym Addicts          223 Others         -332
##  5 66aO3                555 Gym Addicts          345 Gym Regulars   -210
##  6 75dx                 545 Gym Addicts          144 At-Risk        -401
##  7 7vRo                 555 Gym Addicts          344 Gym Regulars   -211
##  8 A6Y7                 454 Gym Addicts          122 Low Priority   -332
##  9 ade0Q                544 Gym Addicts          255 At-Risk        -289
## 10 bD45j                444 Gym Addicts          243 At-Risk        -201
## # ℹ 11 more rows
```

```
## Alternatively, we can also identify who are the Q1 Gym Addicts that are now At-Risk customers
rfm_change %>% filter(segment_Q1 == "Gym Addicts", segment_Q2== "At-Risk")
```

```
## # A tibble: 10 × 6
##    customer_id rfm_score_Q1 segment_Q1  rfm_score_Q2 segment_Q2 change
##    <fct>              <dbl> <chr>              <dbl> <chr>       <dbl>
##  1 75dx                 545 Gym Addicts          144 At-Risk      -401
##  2 ade0Q                544 Gym Addicts          255 At-Risk      -289
##  3 b1oM                 455 Gym Addicts          255 At-Risk      -200
##  4 bD45j                444 Gym Addicts          243 At-Risk      -201
##  5 bDnz7                555 Gym Addicts          133 At-Risk      -422
##  6 GdlmX                454 Gym Addicts          143 At-Risk      -311
##  7 mX0d                 555 Gym Addicts          255 At-Risk      -300
##  8 n504                 544 Gym Addicts          233 At-Risk      -311
##  9 r4m34                544 Gym Addicts          155 At-Risk      -389
## 10 Rq6M                 455 Gym Addicts          255 At-Risk      -200
```

```
## We can also identify who are the Q1 Gym Addicts who no longer visited the gym in Q2.
rfm_change %>% filter(segment_Q1 == "Gym Addicts", is.na(segment_Q2))
```

```
## # A tibble: 2 × 6
##   customer_id rfm_score_Q1 segment_Q1  rfm_score_Q2 segment_Q2 change
##   <fct>              <dbl> <chr>              <dbl> <chr>       <dbl>
## 1 7QjV                 555 Gym Addicts           NA <NA>           NA
## 2 e0Vr                 455 Gym Addicts           NA <NA>           NA
```

K-Means Clustering

Merging the dataframes/variables of interest

```
# Change the factor order of the age category
df$Age <- factor(df$Age, levels = c("<30", "30-39", "40-49", "50-59", ">=60"))

# Merging

## Select columns that contain background information
background <- df %>%
  group_by(ID) %>%
  slice(which.max(Date)) %>%
  select(ID,Passtype,ULU, Gender, Age)
## some customer's demographics get updated, e.g. 16JW Gym Membership ended in June 2022, so we
select their most updated demographics information by using "slice(which.max(Date))"

## Left join
segments_with_info <- left_join(segments,background, by=c("customer_id"="ID"))
unique_df <- segments_with_info %>% distinct(ID, .keep_all = TRUE)
unique_df <- rename (unique_df, Passtype = Passtype.x, ULU = ULU.x, Gender = Gender.x, Age = Ag
e.x)

u_df <- unique_df %>%
  select(customer_id, segment, rfm_score, transaction_count, recency_days, amount, recency_scor
e, frequency_score, monetary_score, Passtype, ULU, Gender, Age)
```

Data Exploration

```
## Obtain the counts, averages, and five number summaries
summary(u_df)
```

```
##   customer_id    segment           rfm_score     transaction_count
##  05Pj   :  1   Length:402         Min.   :111.0   Min.   :  1.00
##  0d2j   :  1   Class :character   1st Qu.:222.0   1st Qu.:  3.25
##  01Xl   :  1   Mode  :character   Median :333.0   Median : 10.00
##  0n6e   :  1                      Mean   :337.4   Mean   : 18.19
##  0p2JO  :  1                      3rd Qu.:455.0   3rd Qu.: 27.00
##  0p2QN  :  1                      Max.   :555.0   Max.   :141.00
##  (Other):396
##   recency_days       amount          recency_score   frequency_score
##  Min.   :  0.0   Min.   :    9.317   Min.   :1.000   Min.   :1.000
##  1st Qu.:  2.0   1st Qu.:  214.287   1st Qu.:2.000   1st Qu.:2.000
##  Median : 16.0   Median :  602.375   Median :3.000   Median :3.000
##  Mean   : 38.6   Mean   : 1270.568   Mean   :3.047   Mean   :2.963
##  3rd Qu.: 66.0   3rd Qu.: 1778.329   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :178.0   Max.   :12966.267   Max.   :5.000   Max.   :5.000
##
##  monetary_score                             Passtype
##  Min.   :1      Fitness Gym Membership        : 57
##  1st Qu.:2      Per-Entry Ticket (Fitness Gym):345
##  Median :3
##  Mean   :3
##  3rd Qu.:4
##  Max.   :5
##
##                                 ULU      Gender     Age
##  Arts & Social Sciences           : 17   F:100   <30  :148
##  College of Design and Engineering:120   M:302   >=60 :  8
##  Engineering                      :  1           30-39:170
##  Others                           :161           40-49: 53
##  School of Computing              : 16           50-59: 23
##  Science                          : 46
##  Yong Loo Lin Sch of Medicine     : 41
```

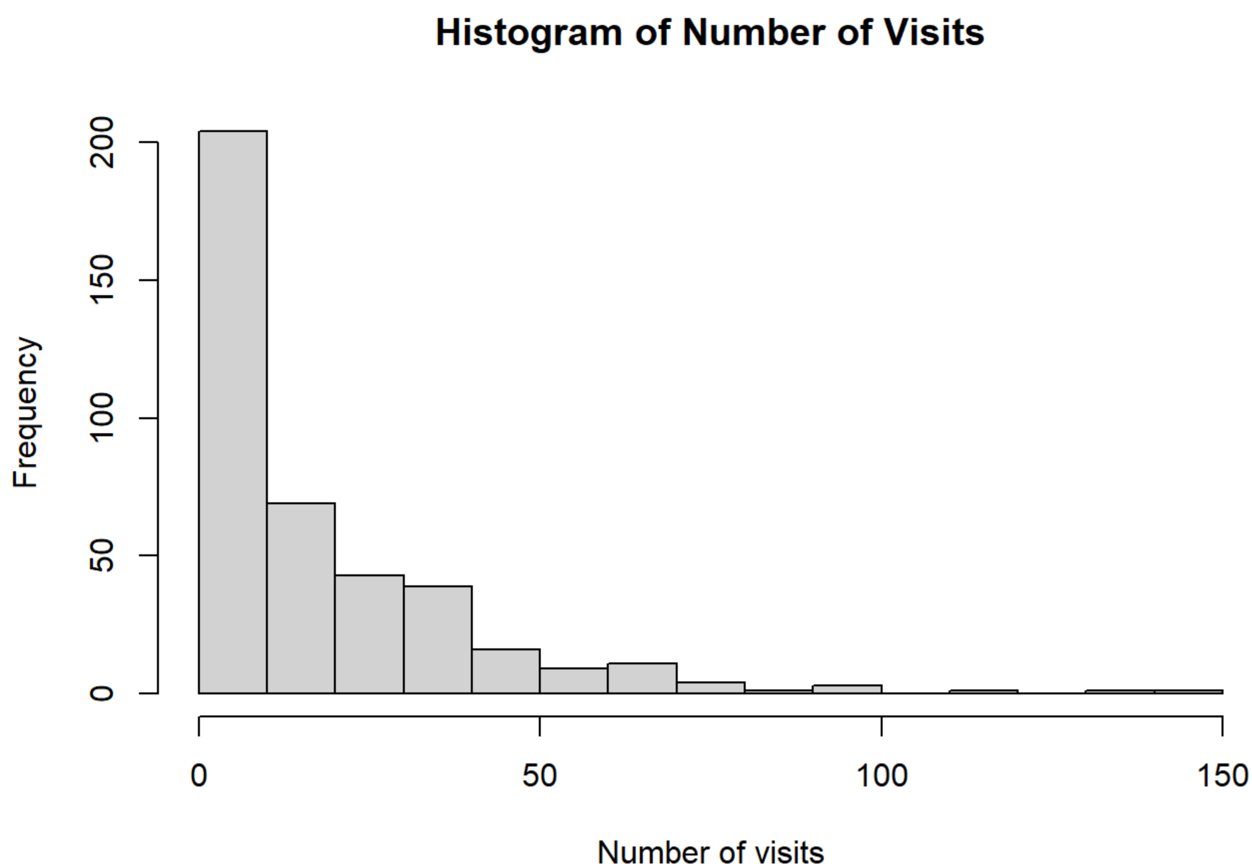
```
## Check correlation between variables used for RFM, closing it is to 1, the stronger the correl
ation
cor(u_df[,c("recency_days","transaction_count", "amount")])
```

```
##                   recency_days transaction_count     amount
## recency_days         1.0000000        -0.3807663 -0.3642133
## transaction_count   -0.3807663         1.0000000  0.9564623
## amount              -0.3642133         0.9564623  1.0000000
```

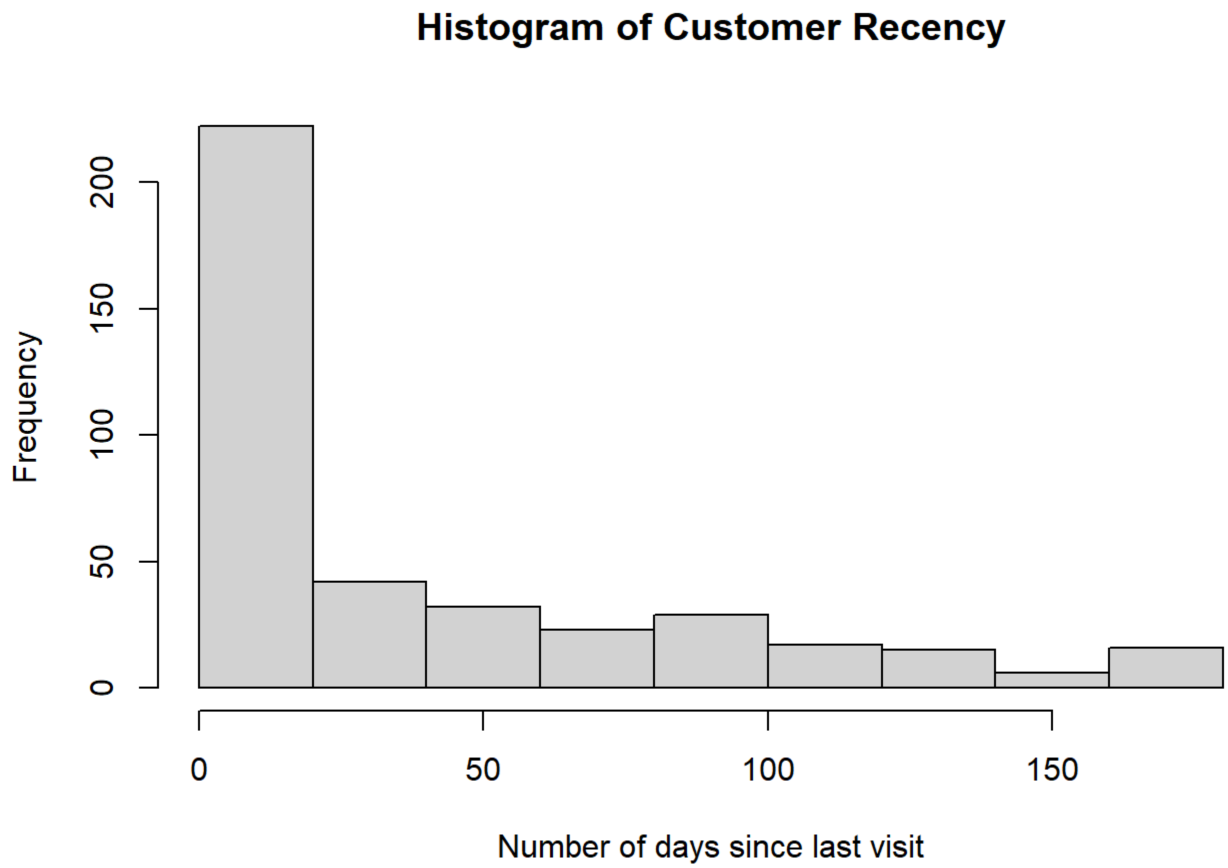
```
## Create a new variable for average duration per visit
u_df <- u_df %>% mutate(avg_duration = amount/transaction_count)

## Check correlation again
cor(u_df[,c("recency_days", "transaction_count","avg_duration")])
```

```
##                   recency_days transaction_count avg_duration
## recency_days         1.0000000        -0.3807663   -0.1149104
## transaction_count   -0.3807663         1.0000000    0.1791194
## avg_duration        -0.1149104         0.1791194    1.0000000
```

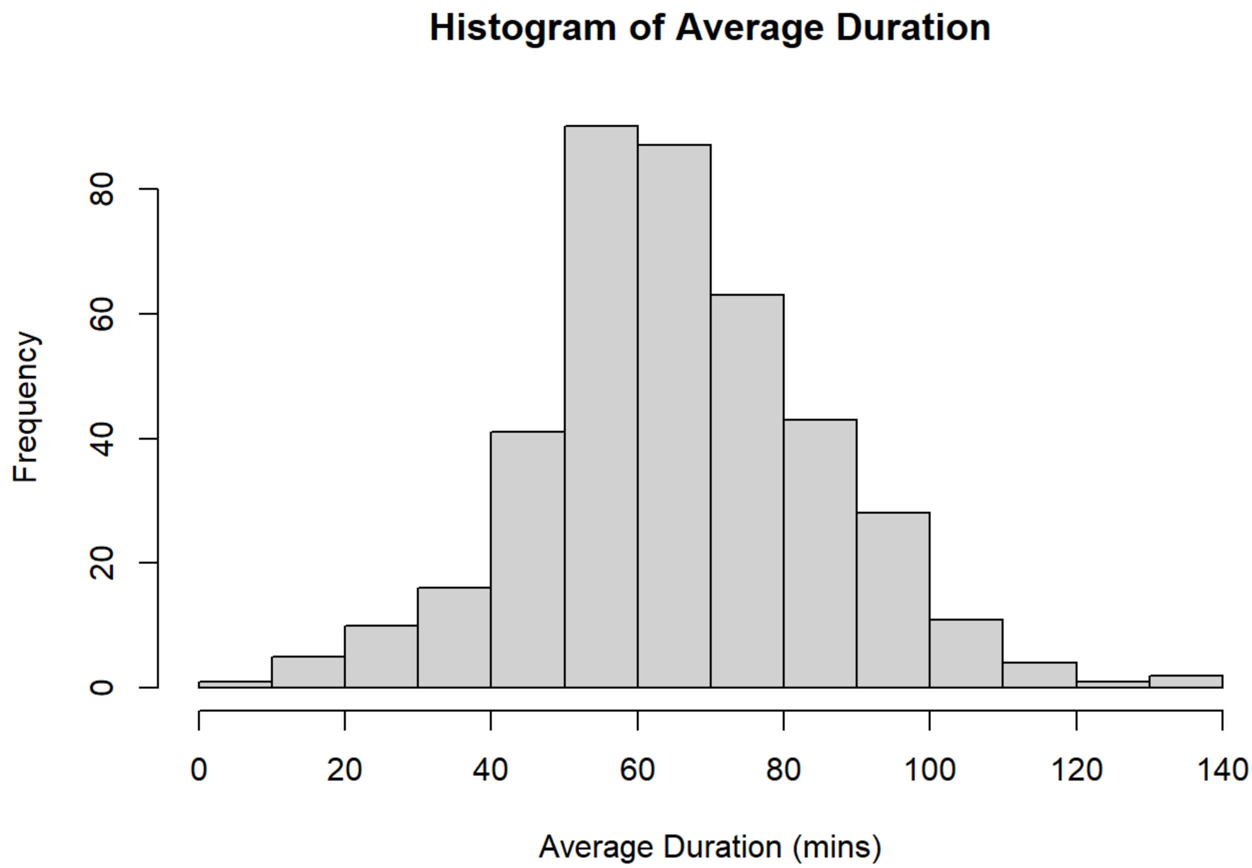
```
## Plot the distributions using histograms
hist(u_df$transaction_count, main = paste("Histogram of Number of Visits"), xlab = "Number of vi
sits")
```




```
hist(u_df$recency_days, main = paste("Histogram of Customer Recency"), xlab = "Number of days si
nce last visit")
```




```
hist(u_df$avg_duration, main = paste("Histogram of Average Duration"), xlab = "Average Duration (mins)")
```




```
# Outliers defined as > 2 standard deviations
```

Data Proprocessing: Scale the variables using z-score standardisation

```
## Standardise the recency_days, transaction_count and avg_duration using preProcess() from the Caret library
preProcValues <- preProcess(u_df[,c("recency_days", "transaction_count","avg_duration")], method =c("center", "scale"))

# New u_df, where values are standardised
u_df_std <- predict(preProcValues,u_df[,c("recency_days", "transaction_count","avg_duration")])

colMeans(u_df_std)
```

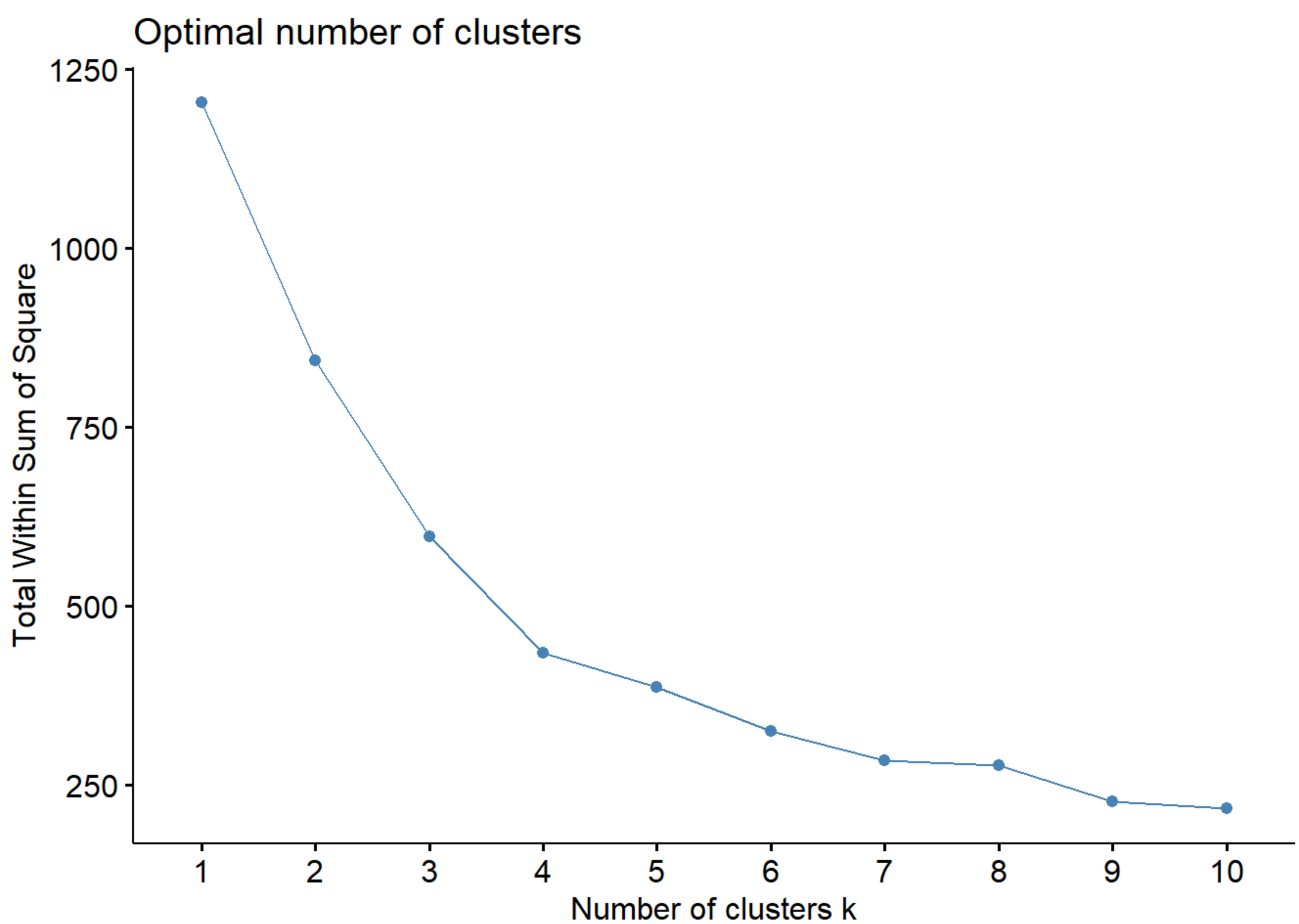
```
##      recency_days transaction_count      avg_duration
##     -3.174285e-17      1.972924e-17      2.969053e-16
```

```
summary(u_df_std)
```

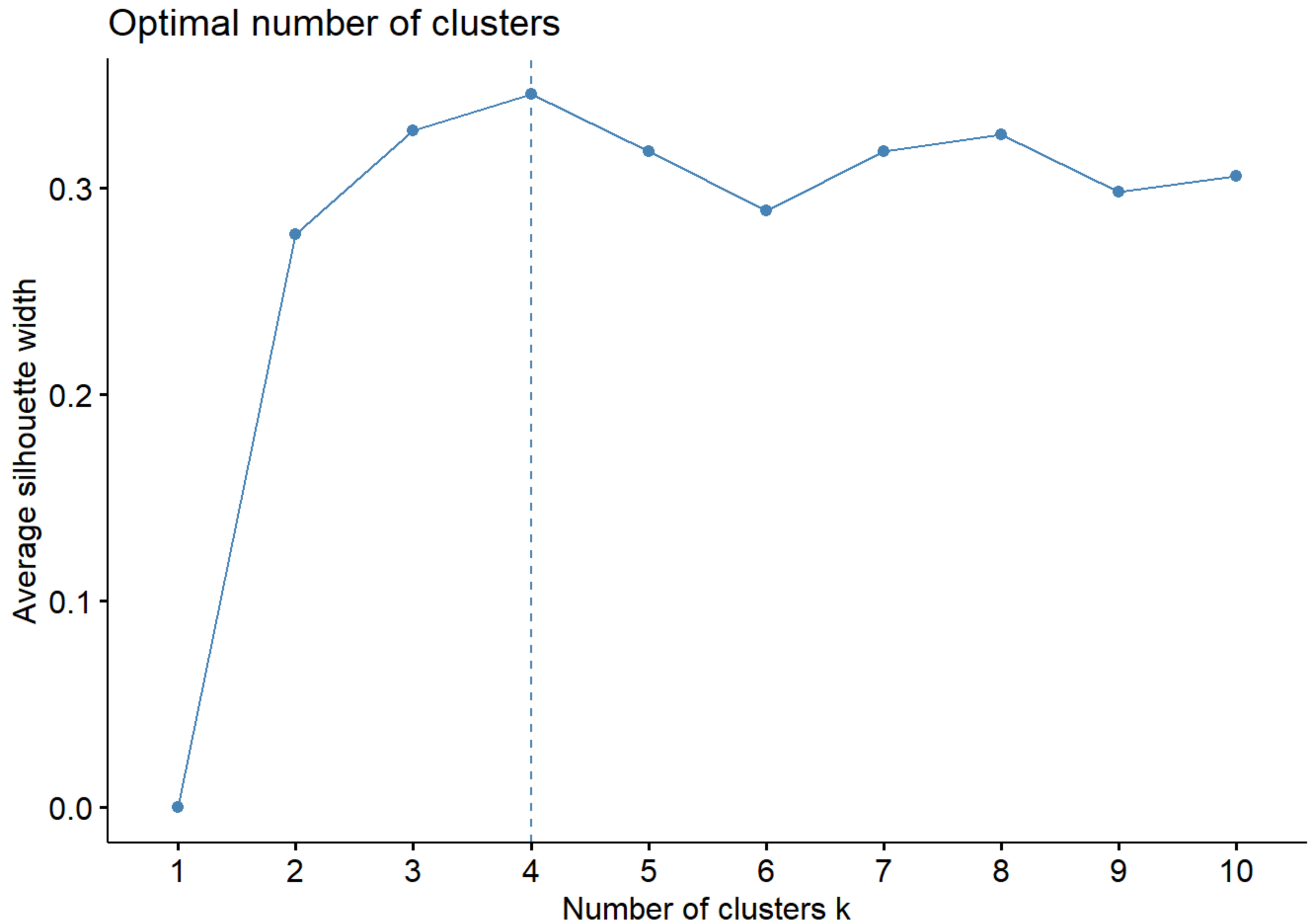
```
##   recency_days     transaction_count  avg_duration
##  Min.   :-0.8013   Min.   :-0.8116   Min.   :-2.83219
##  1st Qu.:-0.7598   1st Qu.:-0.7054   1st Qu.:-0.60111
##  Median :-0.4691   Median :-0.3867   Median :-0.09377
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.5688   3rd Qu.: 0.4160   3rd Qu.: 0.60834
##  Max.   : 2.8938   Max.   : 5.7986   Max.   : 3.52583
```

K-means Clustering: Find the optimal number of clusters

```
# The Elbow Graph
fviz_nbclust(u_df_std[,c("recency_days", "transaction_count","avg_duration")], kmeans, method =
"wss")
```




```
# The Silhouette Graph
fviz_nbclust(u_df_std[,c("recency_days", "transaction_count","avg_duration")], kmeans, method =
"silhouette")
```

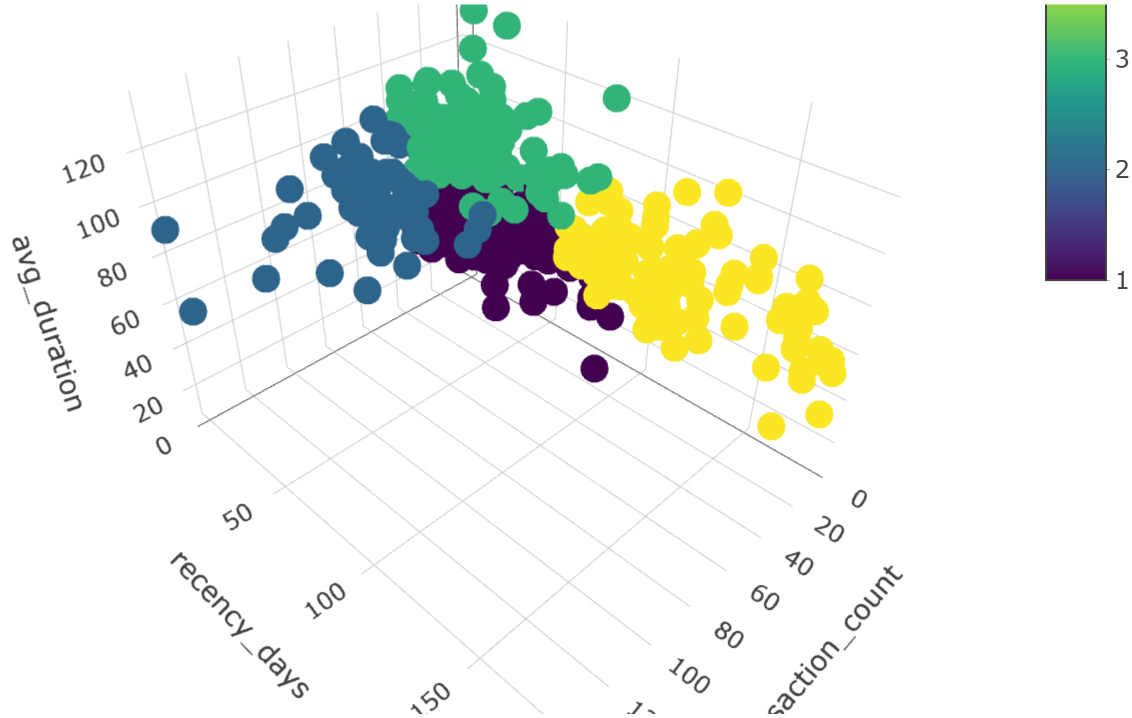


K-means Clustering with k = 4

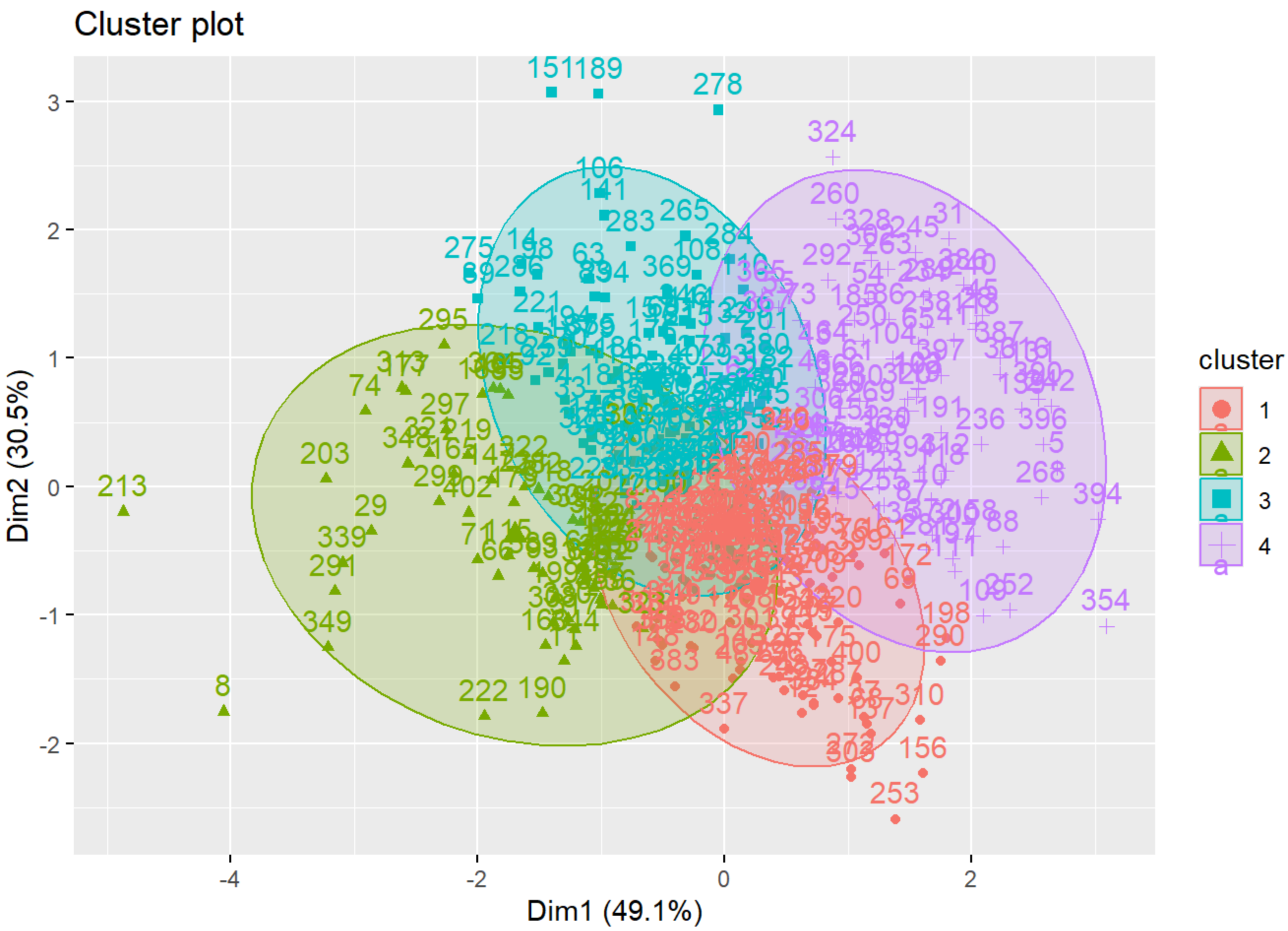
```
set.seed(10) #set seed number
kmeans_4 <- kmeans(u_df_std, 4)

# Size of each cluster
kmeans_4$size
```

```
## [1] 144  66 101  91
```

```
# Visualising the k-means clusters
## 1. Plot the clusters using a 3D scatterplot by using plot_ly from the plotly library

u_df <- u_df %>% mutate(kcluster4 = kmeans_4$cluster) #create an additional column to indicate e
ach customer's cluster assignment (1-4)

plot_ly(u_df, x= ~transaction_count, y= ~recency_days, z= ~avg_duration, type="scatter3d", mode
="markers", color =~kcluster4)
```

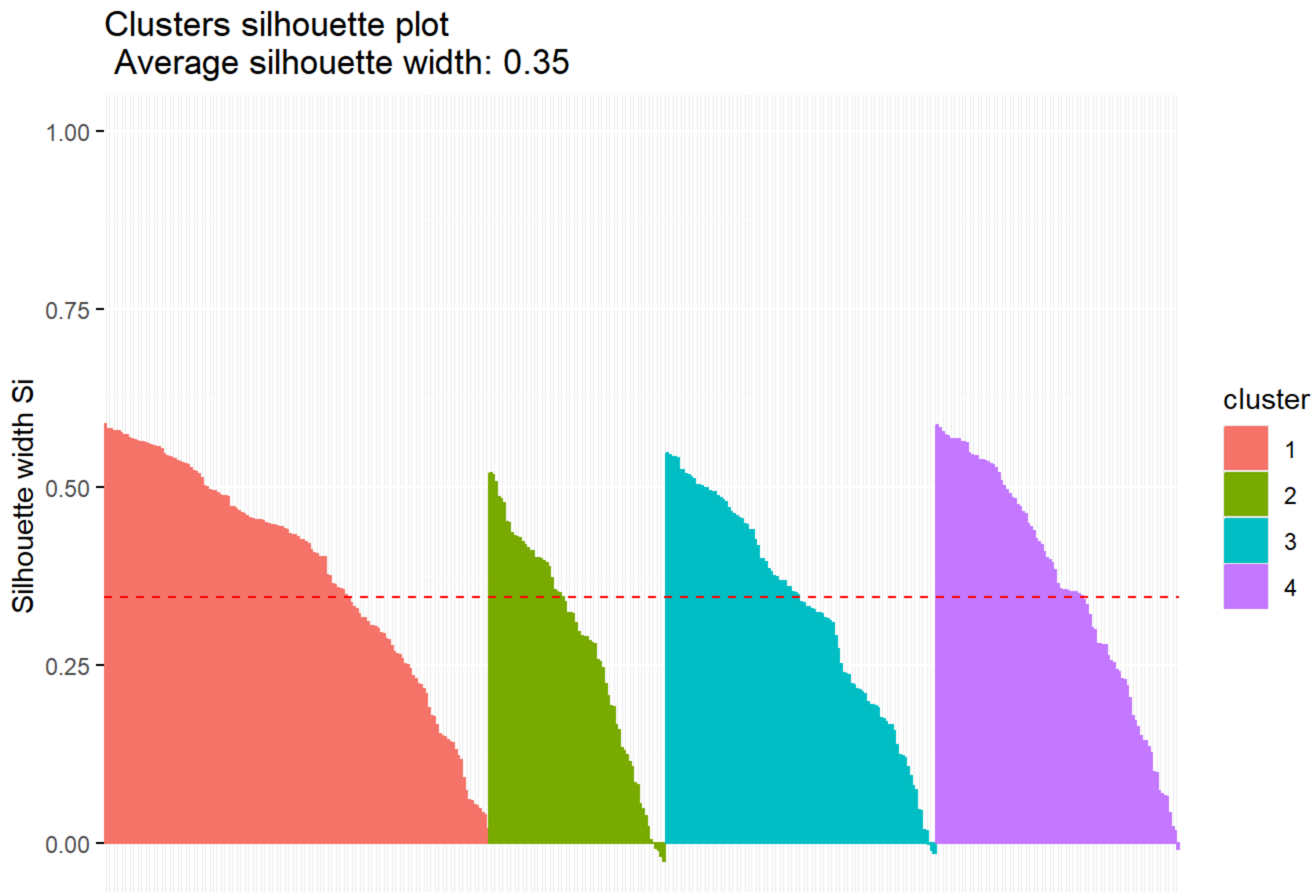



```
## 2. Using fviz_cluster() from the factoExtra library
fviz_cluster(kmeans_4,u_df_std[,c("recency_days", "transaction_count","avg_duration")], ellipse.
type = "norm")
```




```
## Visualise the Silhouette Information
sil <- silhouette(kmeans_4$cluster, dist(u_df_std[,c("recency_days", "transaction_count","avg_du
ration")]))
fviz_silhouette(sil)
```

```
##   cluster size ave.sil.width
## 1       1  144          0.38
## 2       2   66          0.28
## 3       3  101          0.32
## 4       4   91          0.36
```




```
## View the observations with negative Silhouette width
sil
```

```
##        cluster neighbor    sil_width
##   [1,]       1        3  0.5270779855
##   [2,]       3        1  0.3691813571
##   [3,]       1        3  0.4920723517
##   [4,]       3        2  0.3313610390
##   [5,]       4        1  0.4657647717
##   [6,]       1        3  0.3052590975
##   [7,]       1        3  0.4725060919
##   [8,]       2        1  0.3239246503
##   [9,]       1        3  0.5543362207
##  [10,]       4        1  0.3578632913
##  [11,]       2        1  0.3462554058
##  [12,]       1        3  0.5409606228
##  [13,]       3        1  0.3236170880
##  [14,]       3        2  0.3858967577
##  [15,]       4        1  0.2564616737
##  [16,]       1        2  0.1531858788
##  [17,]       3        2  0.3988108627
##  [18,]       3        1  0.5148207549
##  [19,]       1        4  0.5741562641
##  [20,]       2        3  0.1917318548
##  [21,]       1        3  0.5610139235
##  [22,]       2        1  0.2910852585
##  [23,]       1        3  0.4073865432
##  [24,]       1        4  0.5438333486
##  [25,]       1        3  0.5339916862
##  [26,]       2        1  0.2235981507
##  [27,]       1        3  0.5596063933
##  [28,]       4        1  0.5736837549
##  [29,]       2        3  0.4513648688
##  [30,]       1        3  0.3324388525
##  [31,]       4        3  0.4721014382
##  [32,]       1        3  0.1225359251
##  [33,]       3        2  0.1940703364
##  [34,]       4        1  0.0990200523
##  [35,]       1        3  0.3563260735
##  [36,]       4        1  0.5625512367
##  [37,]       1        4  0.4555006203
##  [38,]       3        1  0.1699783275
##  [39,]       3        2  0.1936304305
##  [40,]       3        1  0.4799370688
##  [41,]       4        1  0.5724991179
##  [42,]       1        3  0.4631207274
##  [43,]       4        3  0.3479709852
##  [44,]       1        2  0.2869557196
##  [45,]       4        1  0.5685532374
##  [46,]       4        3  0.0699169009
##  [47,]       3        1  0.2167984112
##  [48,]       1        2  0.4336544686
##  [49,]       1        4  0.4643076267
##  [50,]       1        2  0.1448709226
##  [51,]       3        1  0.2094533930
```

```
## [52,]       1       4  0.4468342203
## [53,]       2       3  0.0230844027
## [54,]       4       3  0.4487692921
## [55,]       4       3  0.2793882766
## [56,]       3       2  0.2222229057
## [57,]       3       1  0.3095928318
## [58,]       3       1  0.2128778847
## [59,]       3       1  0.4603617310
## [60,]       3       1  0.5115344089
## [61,]       4       3  0.4835561915
## [62,]       4       3  0.1511425316
## [63,]       3       1  0.5241340101
## [64,]       2       1  0.3563832396
## [65,]       4       3  0.5829038808
## [66,]       2       3  0.5206522694
## [67,]       1       3  0.5733978409
## [68,]       1       4  0.4477682169
## [69,]       1       4  0.0608254293
## [70,]       1       3  0.4948974611
## [71,]       2       3  0.5180044521
## [72,]       1       3  0.5350358005
## [73,]       4       3  0.0225478592
## [74,]       2       3  0.3519171458
## [75,]       1       4  0.2451170539
## [76,]       1       4  0.3041673404
## [77,]       3       1  0.4647262969
## [78,]       1       3  0.5399031406
## [79,]       3       2  0.5476964200
## [80,]       2       1  0.4011011902
## [81,]       3       1  0.0803832526
## [82,]       1       3  0.0595543628
## [83,]       3       2  0.5174885462
## [84,]       1       3  0.5421517912
## [85,]       2       1  0.1240856635
## [86,]       4       3  0.4910515507
## [87,]       4       1  0.3213822490
## [88,]       4       1  0.3527858510
## [89,]       4       1  0.3506457040
## [90,]       3       1  0.3745728010
## [91,]       1       3  0.4563724899
## [92,]       3       2  0.1070675405
## [93,]       2       1  0.4503000556
## [94,]       4       1  0.4093374887
## [95,]       1       2  0.4015650181
## [96,]       4       1 -0.0084837030
## [97,]       1       3  0.5191716616
## [98,]       3       2  0.4396488467
## [99,]       1       3  0.0432313617
## [100,]      2       1  0.0489976963
## [101,]      1       4  0.1495052598
## [102,]      4       1  0.5636669106
## [103,]      4       1  0.5635521549
```

```
## [104,]       4        3  0.5205746886
## [105,]       2        3 -0.0243669368
## [106,]       3        1  0.4476239296
## [107,]       2        1  0.1072268166
## [108,]       3        1  0.4927706250
## [109,]       4        1  0.1268731596
## [110,]       3        4  0.1657781849
## [111,]       4        1  0.2429128516
## [112,]       1        2  0.3169121667
## [113,]       4        1  0.5776268914
## [114,]       3        1  0.5452982279
## [115,]       2        3  0.5069455934
## [116,]       1        3  0.5682968587
## [117,]       1        3  0.2103127178
## [118,]       4        1  0.4388394906
## [119,]       4        1  0.5439736587
## [120,]       1        3  0.4950075074
## [121,]       1        3  0.2684098471
## [122,]       3        1  0.1988922387
## [123,]       4        1  0.2783533149
## [124,]       3        1 -0.0003638172
## [125,]       1        3  0.5584820878
## [126,]       1        3  0.2584757496
## [127,]       1        2  0.2773354238
## [128,]       4        1  0.2800014920
## [129,]       1        3  0.1506008459
## [130,]       4        1  0.4753818357
## [131,]       4        1  0.5327018565
## [132,]       3        1  0.4549615019
## [133,]       1        4  0.4308626622
## [134,]       1        3  0.3426337421
## [135,]       4        1  0.1720330316
## [136,]       2        1  0.0816240191
## [137,]       1        4  0.4349681712
## [138,]       1        3  0.2936643475
## [139,]       4        1  0.5386916670
## [140,]       1        3  0.0522787843
## [141,]       3        2  0.4820911781
## [142,]       4        1  0.2296867771
## [143,]       1        2  0.1792308444
## [144,]       3        1  0.4952243617
## [145,]       3        1  0.1376730361
## [146,]       3        2  0.4266214645
## [147,]       2        3  0.3875035466
## [148,]       1        2  0.2304014580
## [149,]       1        2  0.4535581322
## [150,]       3        1  0.0465768559
## [151,]       3        2  0.3684501989
## [152,]       2        1  0.1337123348
## [153,]       4        1  0.3529909203
## [154,]       1        3  0.5816570731
## [155,]       4        1  0.2411834315
```

```
## [156,]       1        4  0.3052429315
## [157,]       3        1  0.5029408275
## [158,]       4        1  0.3556264745
## [159,]       3        1  0.2516713206
## [160,]       2        1  0.3935562712
## [161,]       1        4  0.1416192887
## [162,]       1        3  0.5623294500
## [163,]       1        3  0.5022380327
## [164,]       4        3  0.1440720675
## [165,]       2        3  0.4312994191
## [166,]       2        1  0.0847030575
## [167,]       3        1  0.3124074683
## [168,]       1        3  0.3592565585
## [169,]       2        3  0.1139379055
## [170,]       1        3  0.4403310401
## [171,]       1        3  0.1901320723
## [172,]       1        4  0.0918220740
## [173,]       2        3  0.3992998401
## [174,]       1        3  0.4539922053
## [175,]       1        4  0.5311450867
## [176,]       3        1  0.5189512457
## [177,]       2        3  0.2895224694
## [178,]       3        1  0.0747325486
## [179,]       3        2  0.3986792436
## [180,]       1        3  0.2644738390
## [181,]       3        1  0.4992649407
## [182,]       3        1  0.2726654644
## [183,]       1        3  0.4260148792
## [184,]       2        1 -0.0089035061
## [185,]       4        3  0.3422623223
## [186,]       3        1  0.5422042395
## [187,]       3        2  0.4938023194
## [188,]       2        1  0.0041418390
## [189,]       3        4  0.3812340986
## [190,]       2        1  0.3223543308
## [191,]       4        1  0.5480011962
## [192,]       1        3  0.5219170843
## [193,]       1        3  0.5003039029
## [194,]       3        2  0.3286563647
## [195,]       3        1  0.4402805586
## [196,]       1        4  0.4721092405
## [197,]       4        1  0.2210613946
## [198,]       1        4  0.0205777851
## [199,]       2        1  0.4149735182
## [200,]       3        1  0.3319979810
## [201,]       3        4  0.3165793904
## [202,]       3        1  0.1243375462
## [203,]       2        3  0.4013118149
## [204,]       1        3  0.2522997414
## [205,]       1        3  0.5653714209
## [206,]       1        3  0.3625795521
## [207,]       1        3  0.5785225225
```

```
## [208,]        1         3  0.4458394049
## [209,]        1         4  0.4016739040
## [210,]        3         1  0.3604665673
## [211,]        1         3  0.5368746897
## [212,]        2         1  0.1932010914
## [213,]        2         3  0.2847574880
## [214,]        1         3  0.4121264014
## [215,]        3         4  0.1234168899
## [216,]        1         3  0.4505820361
## [217,]        4         1  0.2787502692
## [218,]        3         2  0.1897176026
## [219,]        2         3  0.4228495385
## [220,]        1         4  0.5229325645
## [221,]        3         2  0.4174471753
## [222,]        2         1  0.3729198162
## [223,]        3         1 -0.0086176766
## [224,]        1         3  0.0733006340
## [225,]        1         3  0.3294884328
## [226,]        2         1 -0.0179286848
## [227,]        1         3  0.5468313597
## [228,]        3         1  0.5036827064
## [229,]        3         2  0.0953666477
## [230,]        4         1  0.4271820625
## [231,]        2         3  0.3097119510
## [232,]        1         3  0.5795619279
## [233,]        2         1  0.1294341273
## [234,]        4         3  0.5376672050
## [235,]        4         1  0.2996891036
## [236,]        4         1  0.5357518708
## [237,]        2         1  0.2066332627
## [238,]        4         3  0.5672475224
## [239,]        4         1  0.0646253668
## [240,]        4         3  0.5450539723
## [241,]        3         1  0.0190777904
## [242,]        4         1  0.5094865559
## [243,]        1         3  0.4867905768
## [244,]        1         3  0.4700489181
## [245,]        4         3  0.4627690438
## [246,]        1         2  0.1669827996
## [247,]        1         3  0.3166115675
## [248,]        1         4  0.5566416900
## [249,]        3         4  0.2908851196
## [250,]        4         3  0.3980185989
## [251,]        3         1  0.2397070033
## [252,]        4         1  0.2041791344
## [253,]        1         4  0.3566399184
## [254,]        3         1  0.3228569965
## [255,]        4         1  0.0997173300
## [256,]        1         3  0.0537149022
## [257,]        1         3  0.4026528897
## [258,]        1         3  0.2353117880
## [259,]        3         2  0.3521648777
```

```
## [260,]        4        3  0.1793701310
## [261,]        1        4  0.2656432306
## [262,]        1        3  0.5334148641
## [263,]        4        3  0.4237887065
## [264,]        4        1  0.1348817824
## [265,]        3        4  0.4571902313
## [266,]        1        3  0.4884791879
## [267,]        3        1  0.1754687454
## [268,]        4        1  0.4439100995
## [269,]        4        1  0.4003504261
## [270,]        3        2  0.4620262213
## [271,]        1        4  0.1309393785
## [272,]        1        4  0.4485466488
## [273,]        3        1  0.4706679595
## [274,]        4        1  0.0170766109
## [275,]        3        2  0.2153333768
## [276,]        4        1  0.0426062612
## [277,]        1        3  0.2166433016
## [278,]        3        4  0.2372711497
## [279,]        3        1  0.3597852214
## [280,]        1        2  0.3637308258
## [281,]        4        1  0.1434641761
## [282,]        2        3  0.2553746811
## [283,]        3        1  0.4853007641
## [284,]        3        4  0.1578496155
## [285,]        1        4  0.2510035670
## [286,]        3        2  0.3388234462
## [287,]        1        4  0.4882209339
## [288,]        3        1  0.3954755799
## [289,]        4        3  0.5378669141
## [290,]        1        4  0.1425244201
## [291,]        2        3  0.4288560056
## [292,]        4        3  0.2314999587
## [293,]        3        4  0.3280943747
## [294,]        3        1  0.5419822185
## [295,]        2        3  0.0550289743
## [296,]        1        3  0.5662338281
## [297,]        2        3  0.3394600745
## [298,]        1        3  0.5889374810
## [299,]        2        3  0.4784001263
## [300,]        4        1  0.3529457612
## [301,]        1        3  0.5786233418
## [302,]        4        3  0.3935924036
## [303,]        1        4  0.4440003587
## [304,]        2        3  0.2467094068
## [305,]        3        1  0.1198537082
## [306,]        4        1  0.3342025702
## [307,]        3        1  0.4993420865
## [308,]        2        1  0.4193319213
## [309,]        2        3  0.0387307906
## [310,]        1        4  0.0480965045
## [311,]        3        1  0.0449408190
```

```
## [312,]       4        1  0.5023363837
## [313,]       2        3  0.2899762000
## [314,]       1        2  0.4255510078
## [315,]       1        3  0.4529168898
## [316,]       4        1  0.5443356700
## [317,]       2        1  0.3228633273
## [318,]       2        3  0.2815260706
## [319,]       3        1  0.4485749887
## [320,]       4        1  0.5676908043
## [321,]       2        3  0.4101755066
## [322,]       2        3  0.4110411324
## [323,]       2        1 -0.0068254788
## [324,]       4        3  0.1641648823
## [325,]       3        2  0.3382691236
## [326,]       1        3  0.5697391304
## [327,]       1        3  0.2222186948
## [328,]       4        3  0.3546417433
## [329,]       1        3  0.2857108623
## [330,]       2        1  0.3537031636
## [331,]       1        3  0.4536137879
## [332,]       1        2  0.3376420861
## [333,]       1        3  0.5768236179
## [334,]       1        2  0.2958106947
## [335,]       3        1  0.1752245682
## [336,]       2        1  0.2793517725
## [337,]       1        2  0.4412179627
## [338,]       3        1 -0.0127920958
## [339,]       2        3  0.4360709089
## [340,]       1        3  0.0399876822
## [341,]       3        1  0.2237051801
## [342,]       1        3  0.4200752522
## [343,]       1        3  0.5637902250
## [344,]       2        1  0.2967910946
## [345,]       4        1  0.0659740117
## [346,]       3        1  0.5248201063
## [347,]       4        1  0.3022740272
## [348,]       2        3  0.4296213049
## [349,]       2        3  0.3962499587
## [350,]       3        1  0.2370433268
## [351,]       1        3  0.4674374545
## [352,]       2        3  0.1593603096
## [353,]       3        1  0.4873336763
## [354,]       4        1  0.2527899395
## [355,]       3        2  0.5406398339
## [356,]       1        3  0.5642982179
## [357,]       1        3  0.4067569508
## [358,]       1        2  0.3216416229
## [359,]       2        1  0.1667677875
## [360,]       4        1  0.4191239834
## [361,]       3        1  0.3218112083
## [362,]       1        4  0.4958468112
## [363,]       4        1  0.4852960645
```

```
## [364,]       2        3  0.0004687262
## [365,]       4        3  0.0733113740
## [366,]       4        3  0.3640533402
## [367,]       4        3  0.2635270459
## [368,]       3        1  0.3523396215
## [369,]       3        1  0.5018875227
## [370,]       1        3  0.4236533385
## [371,]       3        1  0.0172004720
## [372,]       4        1  0.3555686678
## [373,]       1        3  0.4331524497
## [374,]       1        3  0.3488581735
## [375,]       3        1  0.1664383922
## [376,]       3        1  0.1941902416
## [377,]       3        1  0.3686846997
## [378,]       1        3  0.5822569206
## [379,]       1        4  0.1177992020
## [380,]       3        1  0.3148216841
## [381,]       1        3  0.2232239637
## [382,]       1        3  0.4897845519
## [383,]       1        2  0.3102764171
## [384,]       1        4  0.5132421853
## [385,]       1        3  0.4439446147
## [386,]       4        3  0.5319514420
## [387,]       4        1  0.5674412546
## [388,]       1        3  0.3011158072
## [389,]       2        1  0.4830613211
## [390,]       4        1  0.5270426854
## [391,]       1        3  0.4592860630
## [392,]       2        3  0.2573031912
## [393,]       3        1  0.3761298853
## [394,]       4        1  0.3841010204
## [395,]       1        3  0.5566808679
## [396,]       4        1  0.4958700241
## [397,]       4        1  0.5873060697
## [398,]       3        2  0.3492527983
## [399,]       1        4  0.1770644168
## [400,]       1        4  0.3753552400
## [401,]       1        3  0.3766515800
## [402,]       2        3  0.4859517151
## attr(,"Ordered")
## [1] FALSE
## attr(,"call")
## silhouette.default(x = kmeans_4$cluster, dist = dist(u_df_std[,
##     c("recency_days", "transaction_count", "avg_duration")]))
## attr(,"class")
## [1] "silhouette"
```

What kind of customers does each cluster represent?

```
# Create a Summary Table for the clusters
u_df %>% group_by(u_df$kcluster4) %>%
  summarise(mean_visitation_count = mean(transaction_count), mean_recency = mean(recency_days),
mean_duration = mean(avg_duration), n_customer = n())
```

```
## # A tibble: 4 × 5
##   `u_df$kcluster4` mean_visitation_count mean_recency mean_duration n_customer
##              <int>                 <dbl>        <dbl>         <dbl>      <int>
## 1                1                  11.0        18.7           50.7        144
## 2                2                  55.9         6.02          70.4         66
## 3                3                  14.9        18.2           86.7        101
## 4                4                   6         116.            62.7         91
```

```
# Cluster 2 - high visiting count of 55 times, can be Gym Addicts
# Cluster 4 - low visitation count, high recency, can be At-Risk customers

# How are the RFM customer segmentation and K-Means clustering different?
table(u_df$kcluster4, u_df$segment)
```

```
##
##     At-Risk Gym Addicts Gym Regulars Low Priority New Others
##   1      10          19           65           20   8     22
##   2       5          55            6            0   0      0
##   3      16          26           39            8   3      9
##   4      23           0            0           63   0      5
```

Hierarchical Clustering

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other

Hierarchical Clustering with k = 4

```
# Generate Distance Matrix using euclidean
hc_dist <- dist(u_df_std, method="euclidean")

# You can view the first 7 customers
as.matrix(hc_dist)[1:7, 1:7]
```

```
##           1         2          3        4        5         6          7
## 1 0.0000000 1.3020562 0.34715777 2.146059 3.513832 0.5941875 0.35827142
## 2 1.3020562 0.0000000 1.25618047 1.228840 3.953269 0.8643655 1.17157404
## 3 0.3471578 1.2561805 0.00000000 2.179085 3.262496 0.4410633 0.09175992
## 4 2.1460594 1.2288404 2.17908513 0.000000 4.312758 1.7650701 2.08897407
## 5 3.5138319 3.9532691 3.26249625 4.312758 0.000000 3.3300543 3.27944740
## 6 0.5941875 0.8643655 0.44106329 1.765070 3.330054 0.0000000 0.35034604
## 7 0.3582714 1.1715740 0.09175992 2.088974 3.279447 0.3503460 0.00000000
```

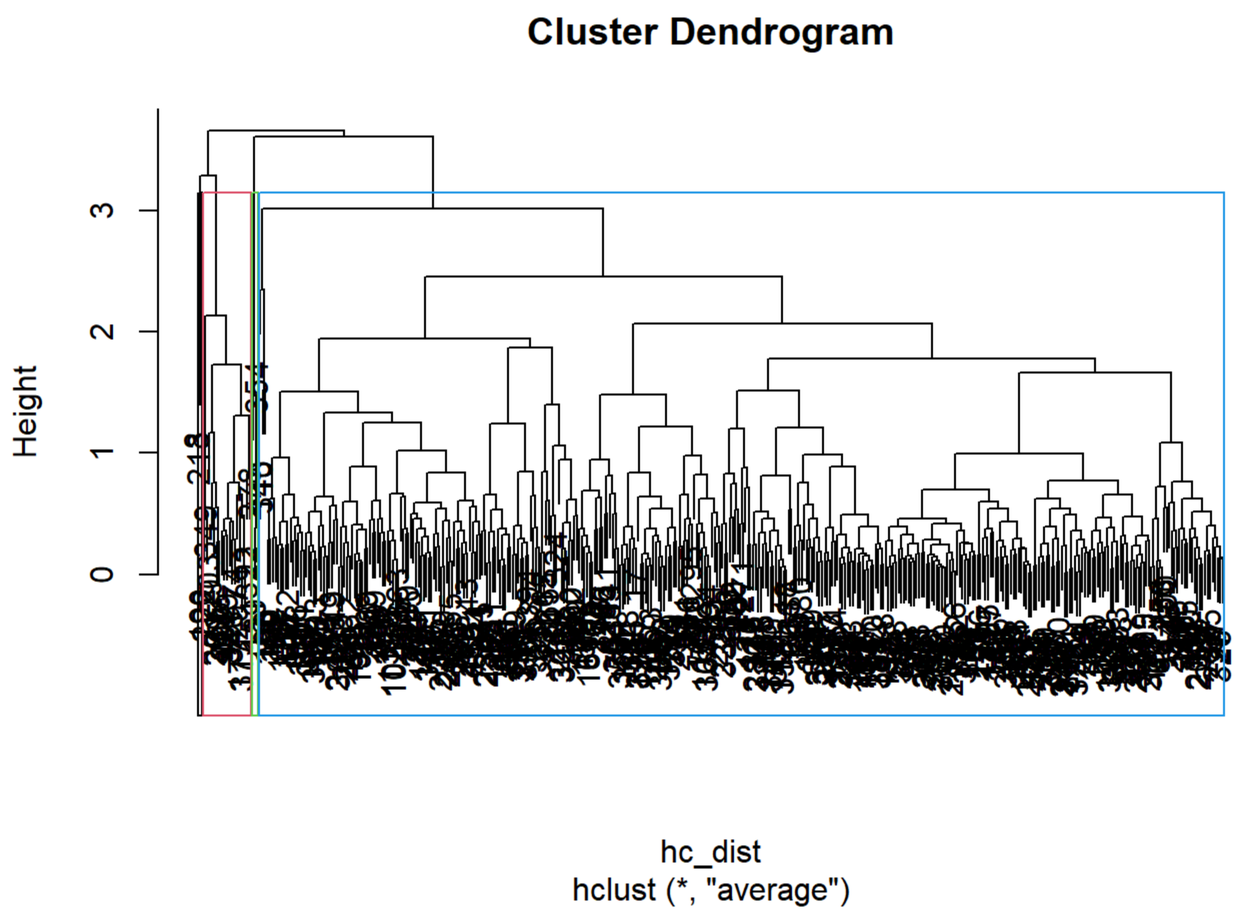
```
# Implement Hierarchical Clustering
hc_cluster <- hclust(d=hc_dist, method="average")
hc_cluster
```

```
##
## Call:
## hclust(d = hc_dist, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 402
```

```
# Calculate the cophenetic distance coefficient
cophenetic_distances <- cophenetic(hc_cluster)
cor(hc_dist, cophenetic_distances)
```

```
## [1] 0.6737833
```

```
# Plot the dendrogram
plot(hc_cluster)

# Plot the k=4 cluster
rect.hclust(hc_cluster, k=4, border = 1:9)
```




```
# As you can see, the dendrogram is very messy due to the large dataset.
# Let's try preforming the Hierarchical Clustering steps with a sample data from the gym.
#If you are working with a large dataset with many observations, you are generally better off av
oiding hierarchical clustering
```

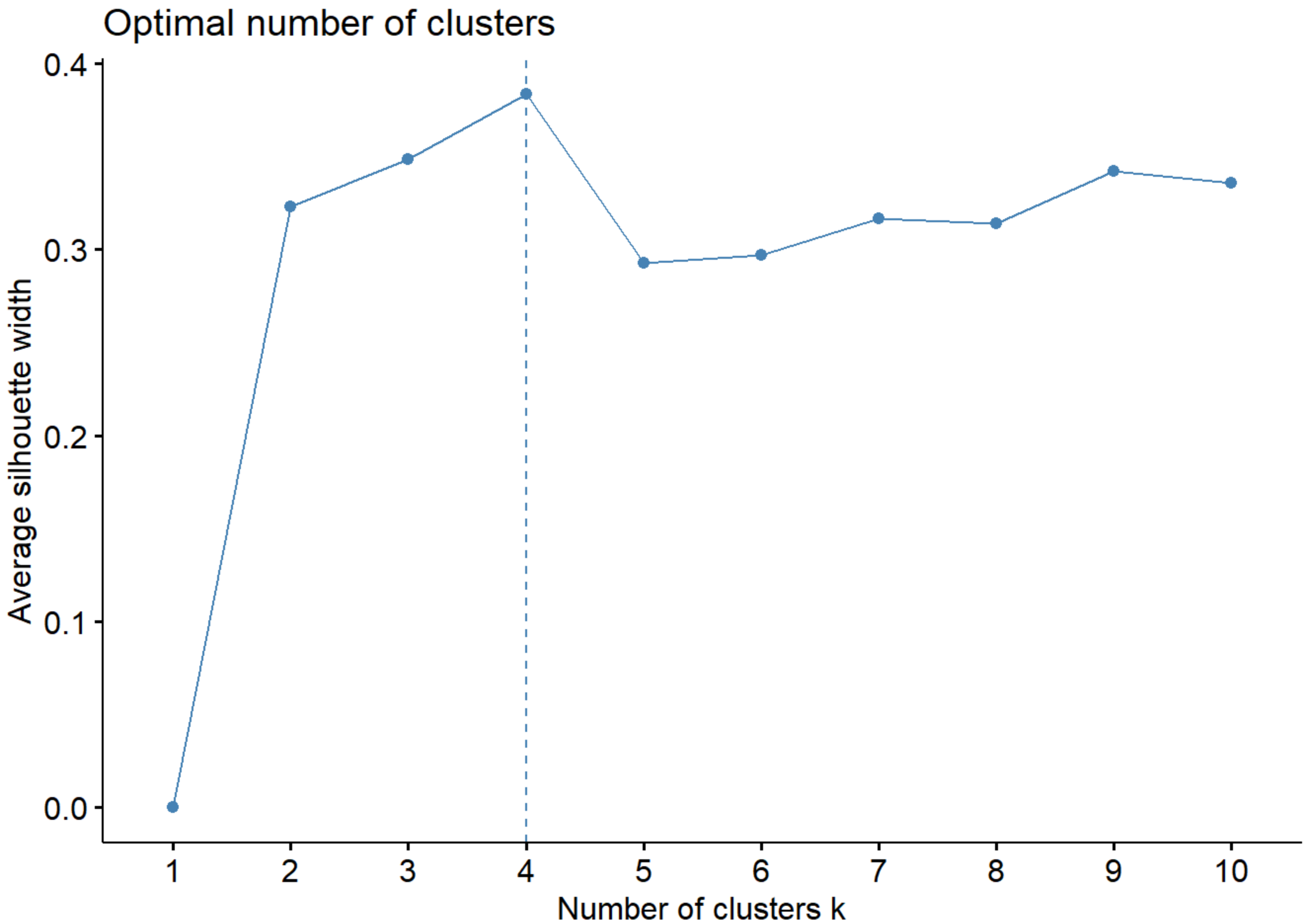
Hierarchical Clustering with 40 random samples from

original dataset.

```
# Randomly sample 40 from the u_df dataset (remember the set.seed)
set.seed(44)
u_df1 <- u_df[sample(nrow(u_df), 40),]

# Standardise the recency_days, transaction_count and avg_duration. Do remember to install and u
pload the "caret" library.
preProcValues <- preProcess(u_df1[,c("recency_days", "transaction_count","avg_duration")], metho
d=c("center", "scale"))

u_df1_std <- predict(preProcValues,u_df1[,c("recency_days", "transaction_count","avg_duratio
n")])

u_df1_dist <- dist(u_df1_std, method="euclidean")

# First 7 Customers
as.matrix(u_df1_dist)[1:7, 1:7]
```

```
##           1         2        3         4         5         6         7
## 1 0.0000000 1.1864881 1.754931 1.0103600 0.3525176 0.8846783 2.1432202
## 2 1.1864881 0.0000000 1.692367 0.1777101 1.3617245 1.7681102 0.9984679
## 3 1.7549313 1.6923674 0.000000 1.6693881 1.6283367 1.4036564 2.0764802
## 4 1.0103600 0.1777101 1.669388 0.0000000 1.1965084 1.6231713 1.1645889
## 5 0.3525176 1.3617245 1.628337 1.1965084 0.0000000 0.5844443 2.3209114
## 6 0.8846783 1.7681102 1.403656 1.6231713 0.5844443 0.0000000 2.6442724
## 7 2.1432202 0.9984679 2.076480 1.1645889 2.3209114 2.6442724 0.0000000
```

```
# The Silhouette Graph (hcut method)
fviz_nbclust(u_df1_std[,c("recency_days", "transaction_count","avg_duration")], hcut, method =
"silhouette")
```




```
# Implement Hierarchical Clustering
u_df1_hc <- hclust(d=u_df1_dist, method="average")
u_df1_hc
```

```
##
## Call:
## hclust(d = u_df1_dist, method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 40
```

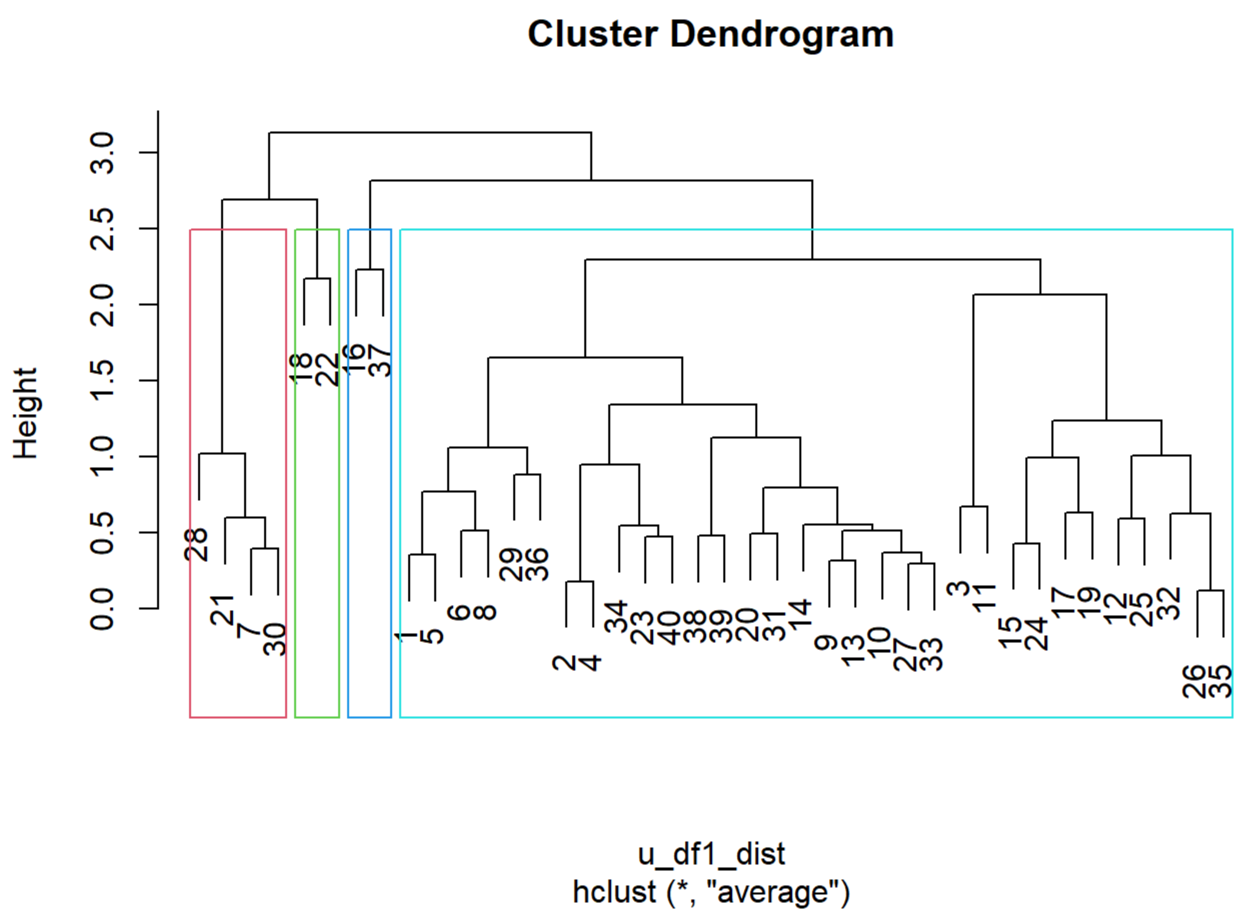
```
# Calculate the cophenetic distance coefficient
u_df1_cophenetic_distances <- cophenetic(u_df1_hc)
cor(u_df1_dist, u_df1_cophenetic_distances)
```

```
## [1] 0.7473425
```

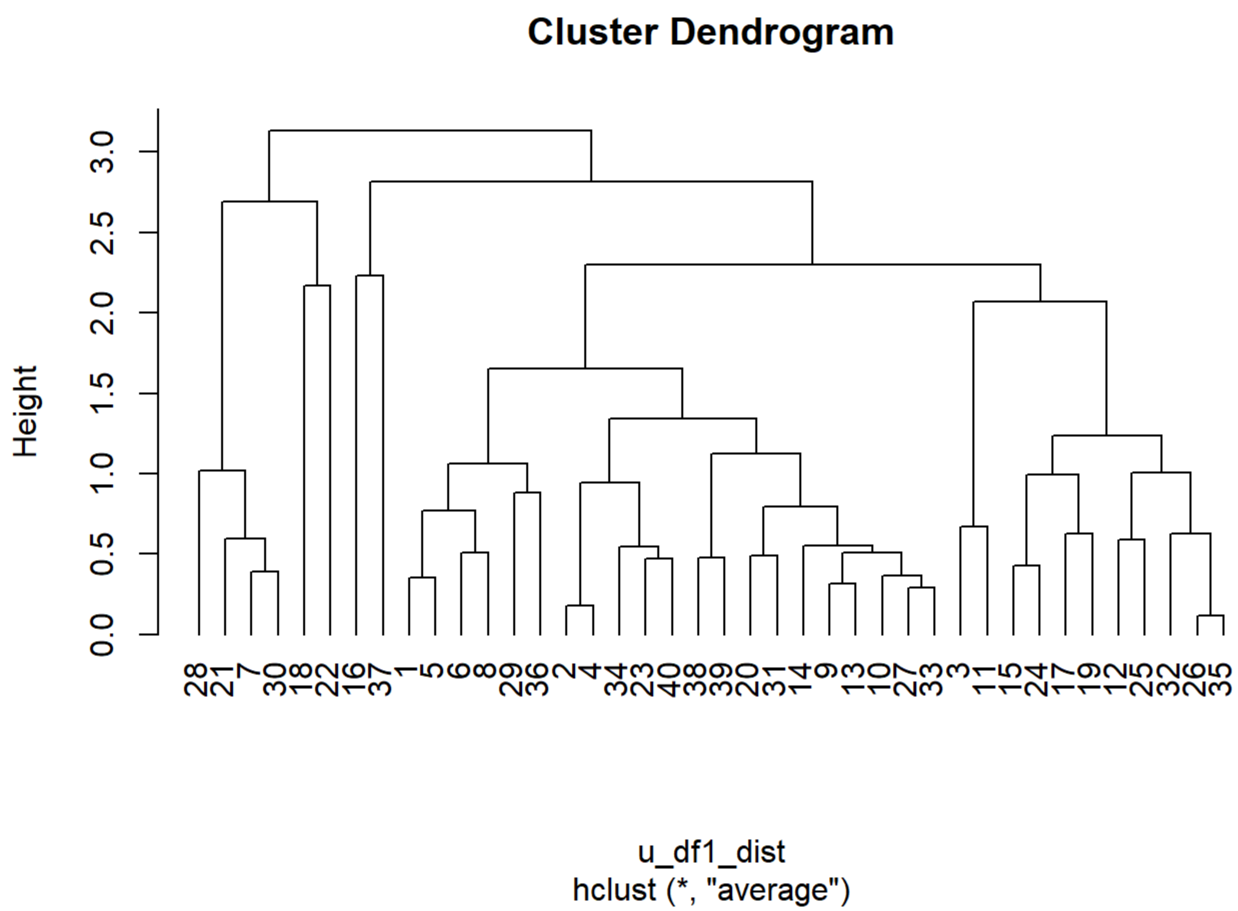
```
# Obtain K=4 Hierarchical Cluster Assignments
u_df1$hcluster4 <- cutree(tree=u_df1_hc, k=4)

# Show the number of observations in each cluster group
table(u_df1$hcluster4)
```

```
##
##  1  2  3  4
## 32  4  2  2
```

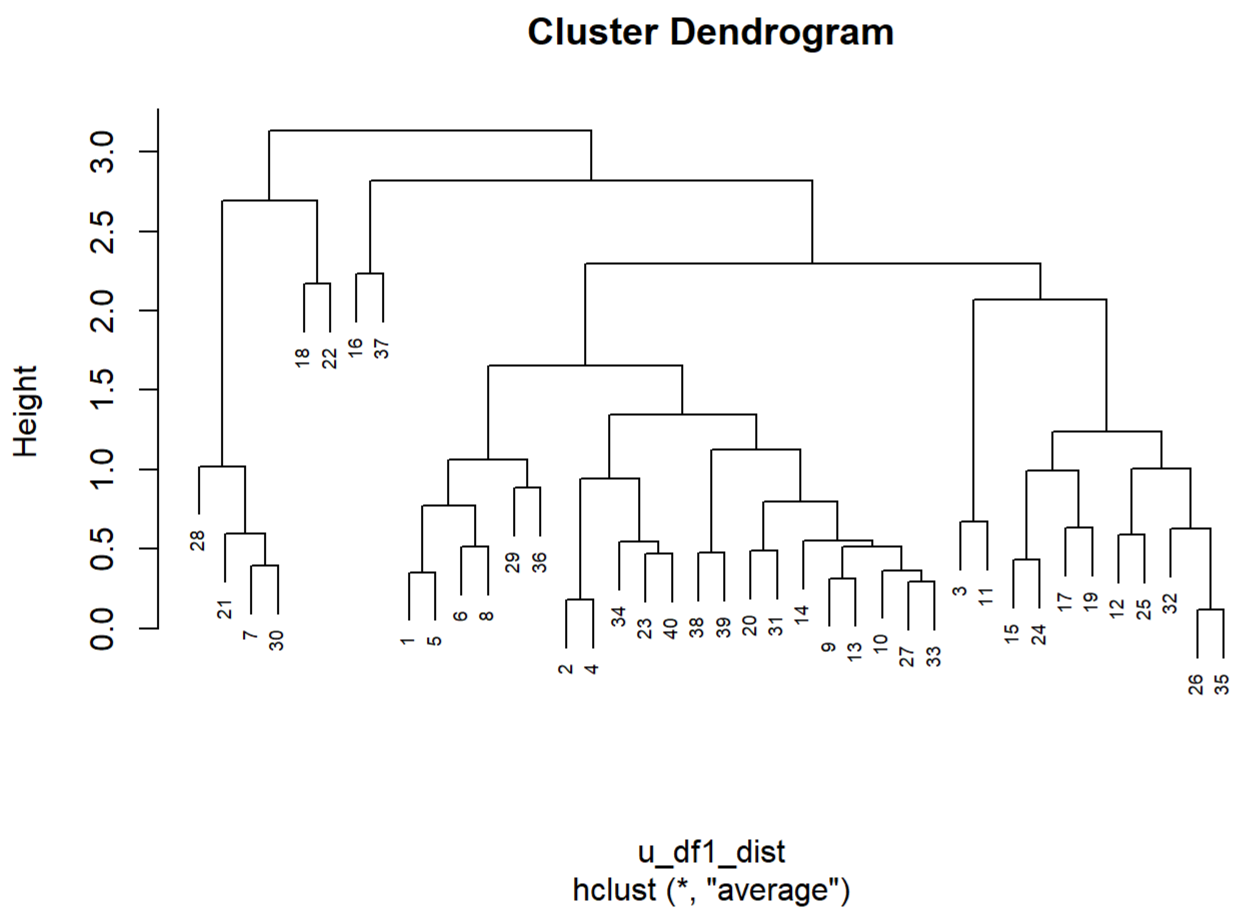
```
# Plot the dendrogram
plot(u_df1_hc)

# Plot rectangle for k=4 cluster
rect.hclust(u_df1_hc, k=4, border = 2:5)
```



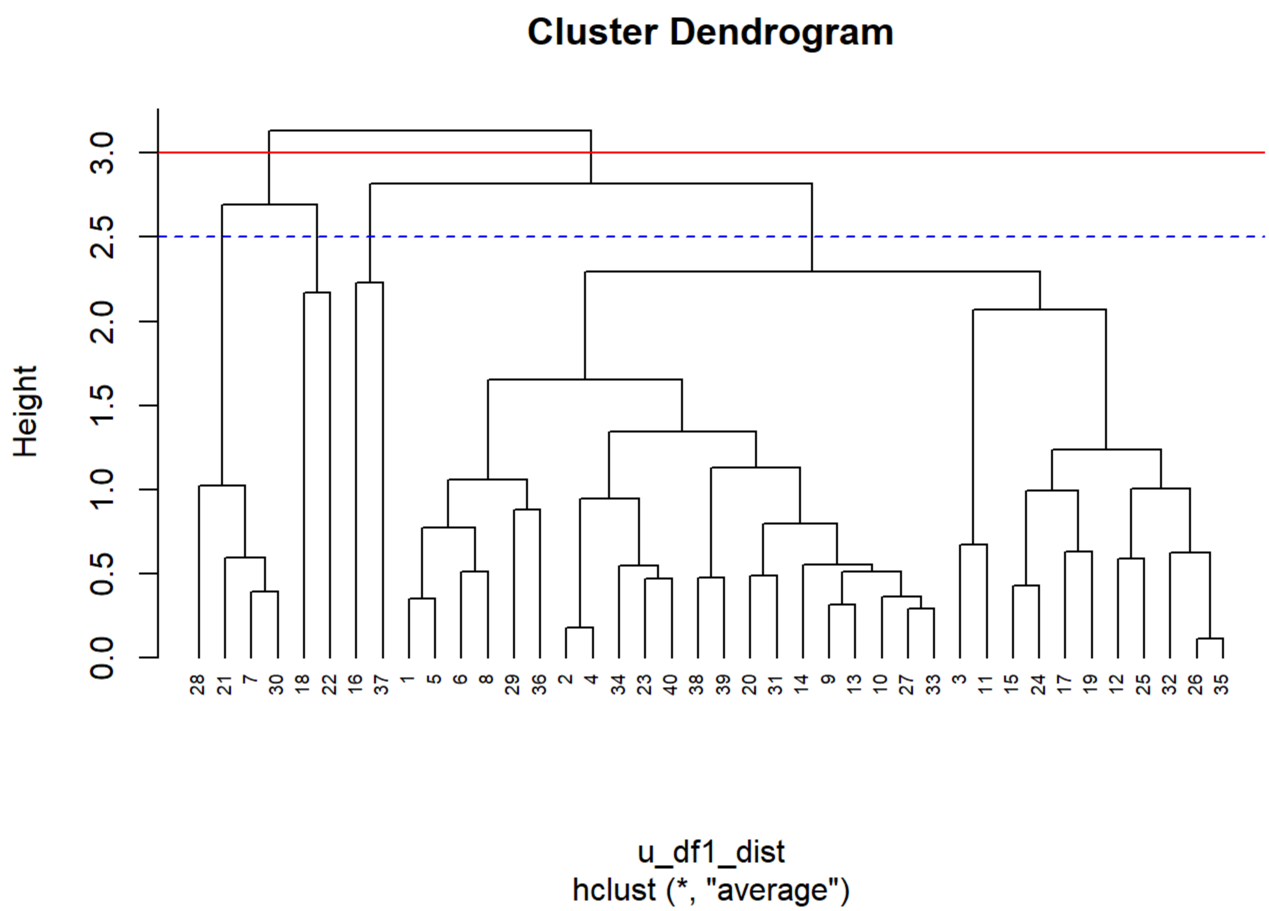

```
# Other variations for the dendrogram
plot(u_df1_hc, hang =-1)
```




```
plot(u_df1_hc, cex =0.6)  #text size
```



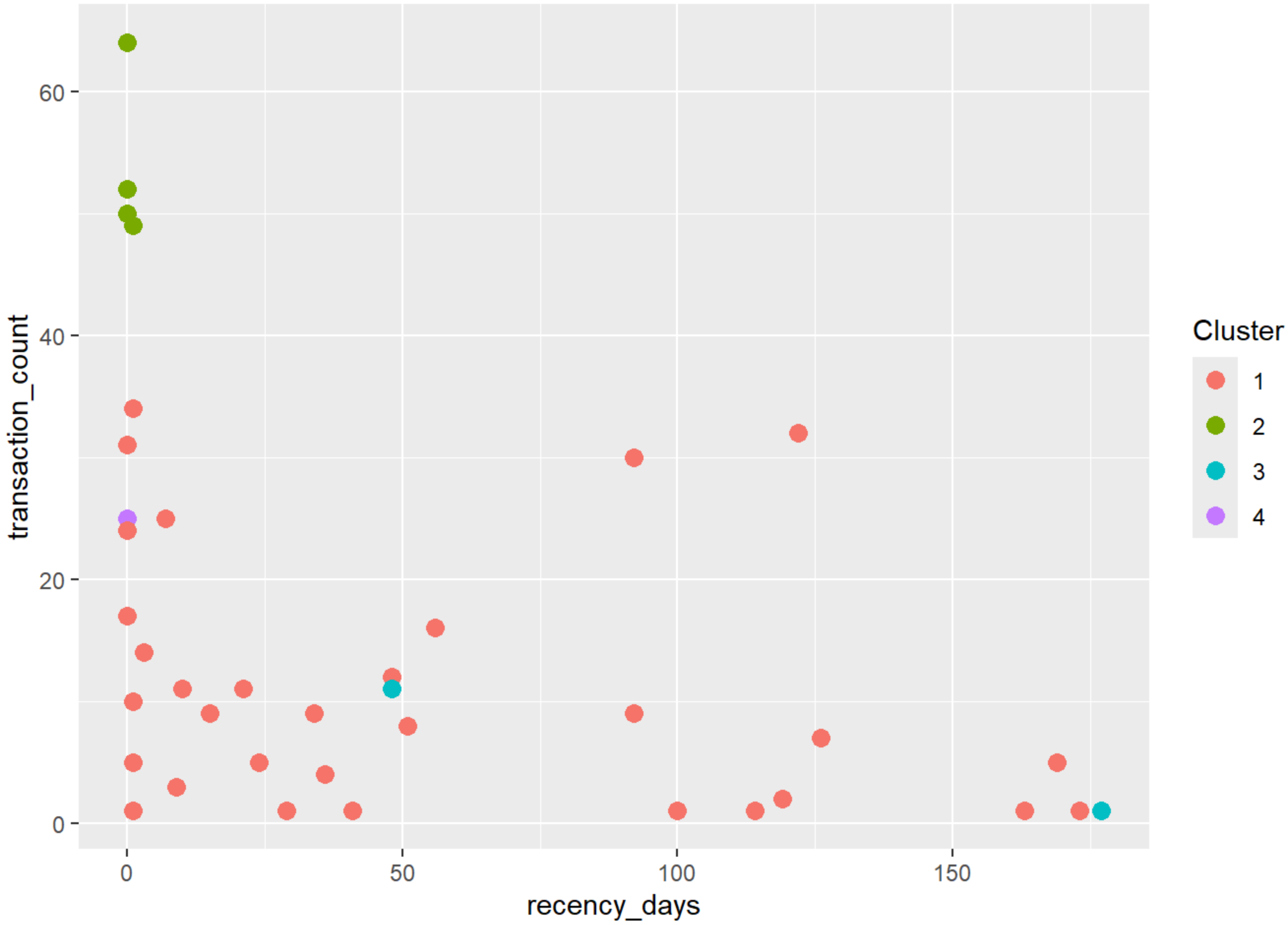

```
plot(u_df1_hc, hang = -1, cex =0.6) #alignment


# Examining the threshold and number of clusters
# To draw a threshold line
abline(h = 3, lty = 1, col="red")
# Clusters if the threshold is at 3 is 2

abline(h = 2.5, lty = 2, col="blue")
```



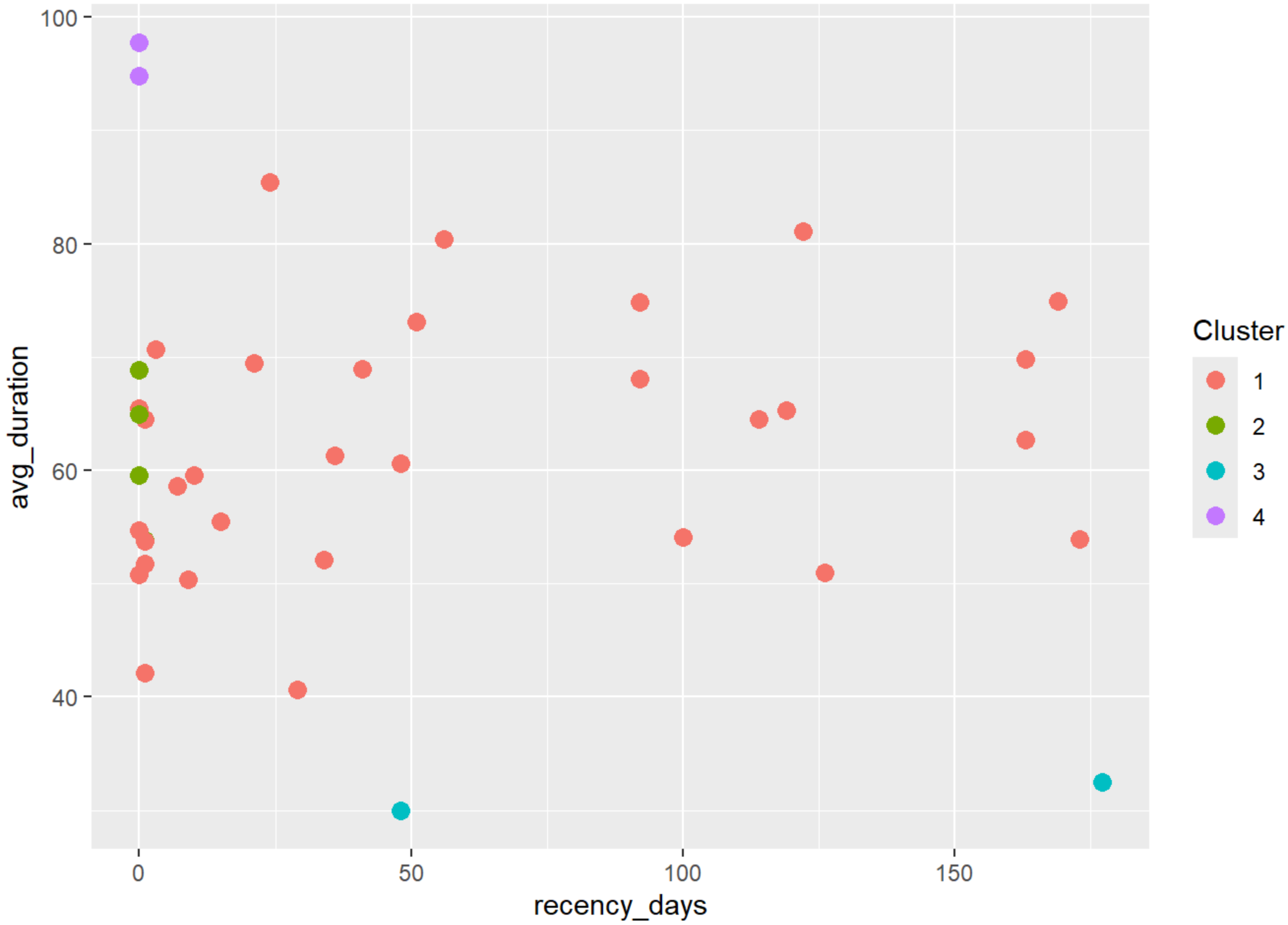

```
# Clusters if the threshold is at 2.5 is 4
```

Visualisation of Hierarchical Clustering (k=5)

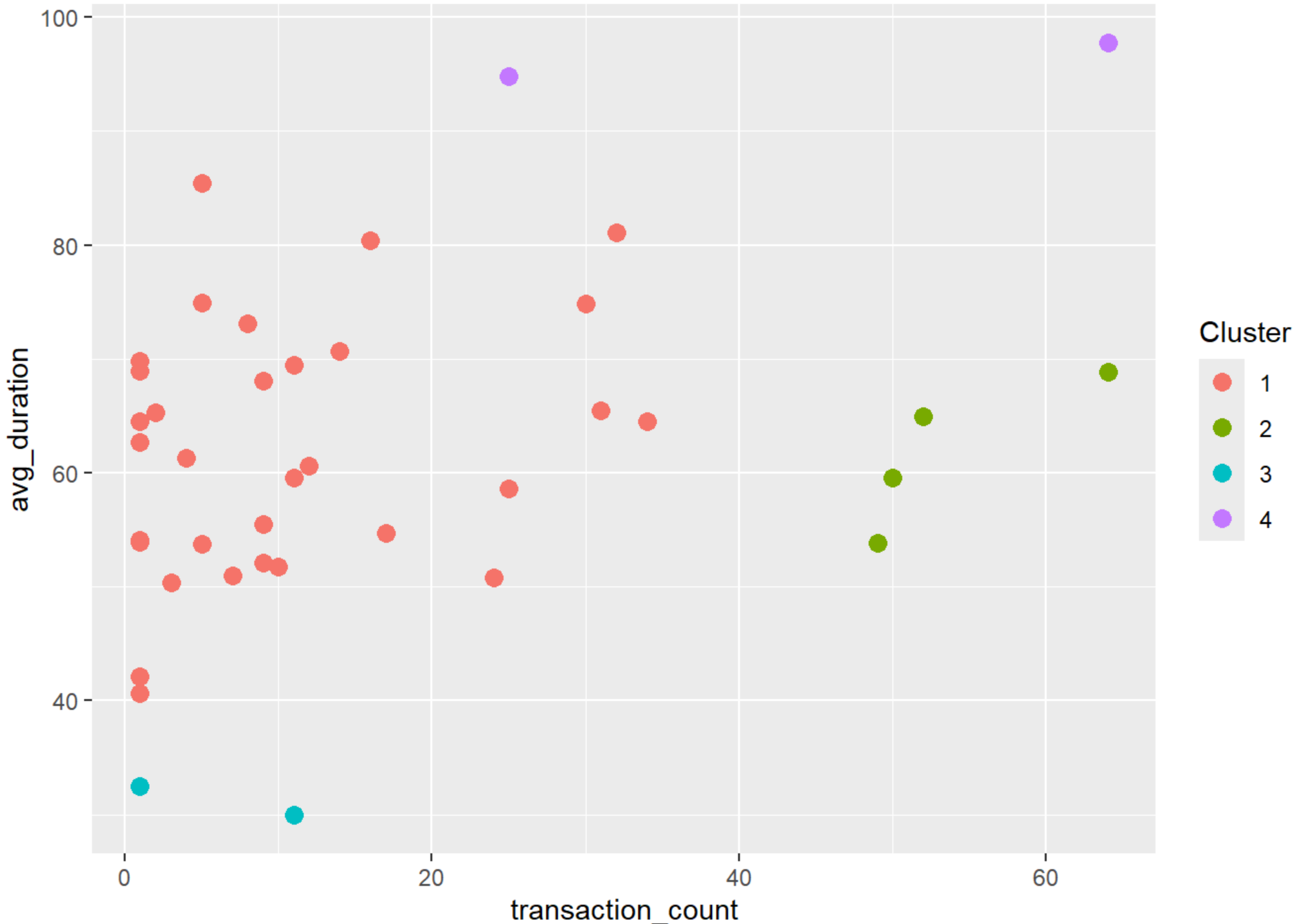
```
# Plot using the standard scatterplot
u_df1 %>%
    ggplot(aes(x=recency_days, y=transaction_count, color=factor(hcluster4)))+
    geom_point(size=3) + labs(color = "Cluster")
```




```
u_df1 %>%
    ggplot(aes(x=recency_days, y=avg_duration, color=factor(hcluster4)))+
    geom_point(size=3) + labs(color = "Cluster")
```

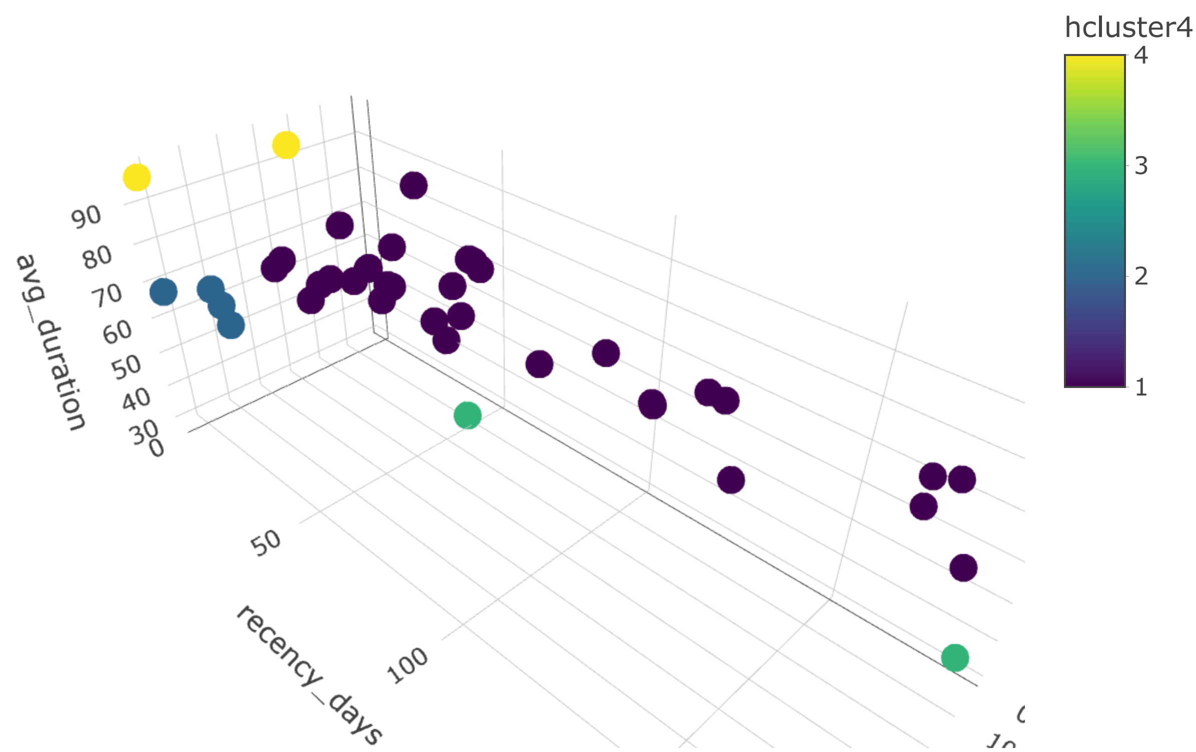



```
u_df1 %>%
    ggplot(aes(x=transaction_count, y=avg_duration, color=factor(hcluster4)))+
    geom_point(size=3) + labs(color = "Cluster")
```

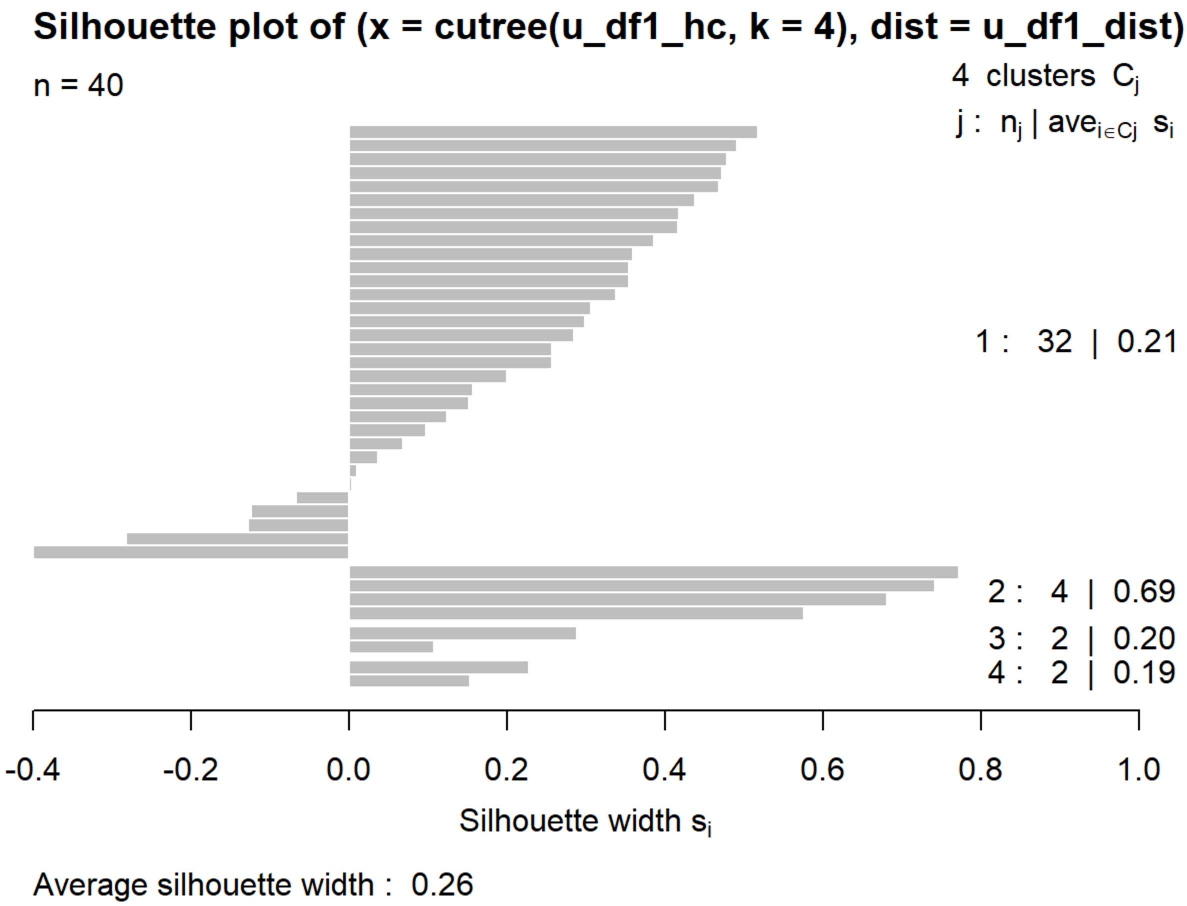



```
# Plot Hierarchical Clustering observations on a 2D plot
fviz_cluster(list(data=u_df1[,c("recency_days", "transaction_count","avg_duration")], cluster=u_
df1$hcluster4))
```


```
# Plot Hierarchical Clustering observations on a 3D plot
plot_ly(u_df1, x= ~transaction_count, y= ~recency_days, z= ~avg_duration, type="scatter3d", mode
="markers", color =~hcluster4)
```

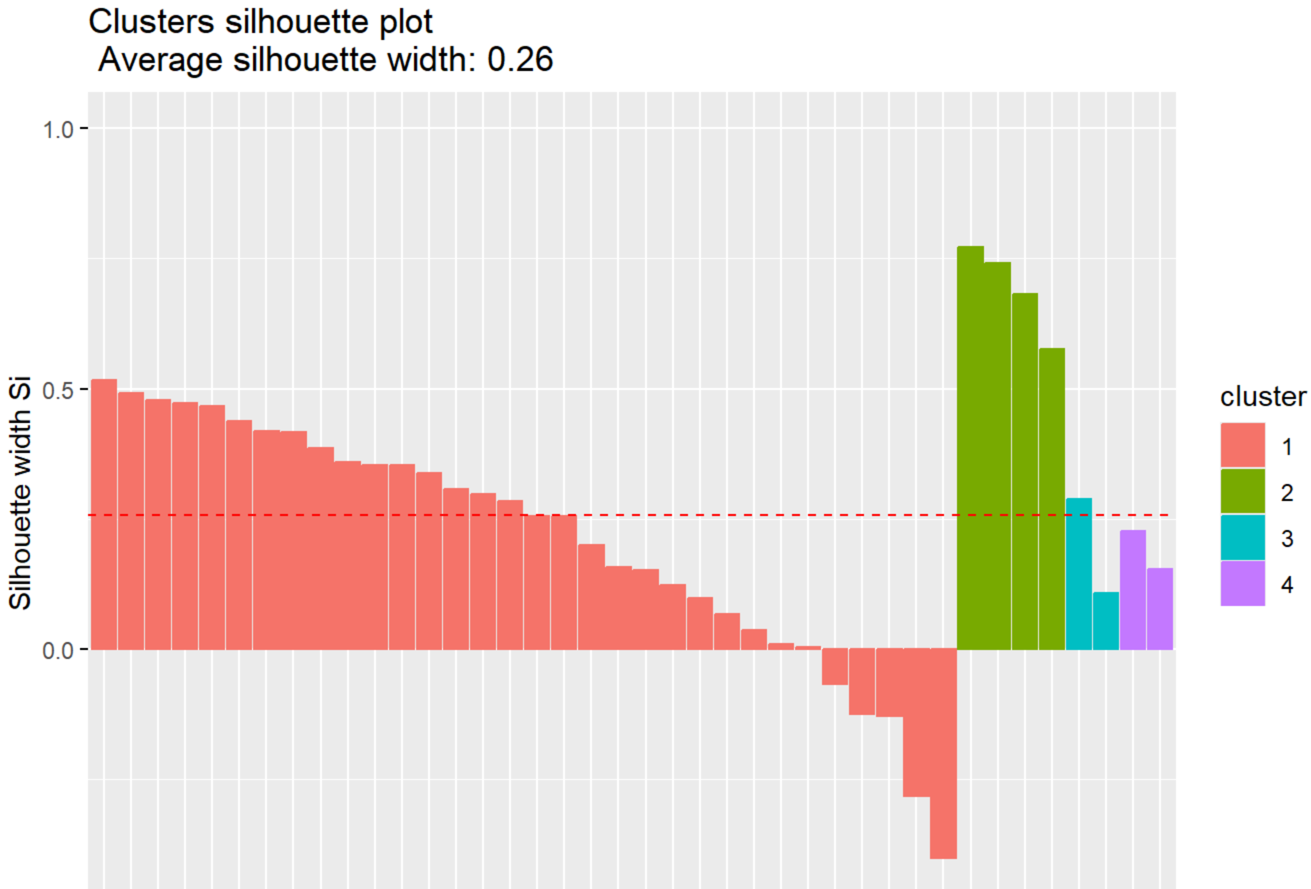



```
# Plot silhouette
plot(silhouette(cutree(u_df1_hc, k=4), u_df1_dist))
```




```
# Visualise the Silhouette Information
HC_sil<- silhouette(cutree(u_df1_hc, k=4), u_df1_dist)
fviz_silhouette(HC_sil)
```

```
##   cluster size ave.sil.width
## 1       1   32          0.21
## 2       2    4          0.69
## 3       3    2          0.20
## 4       4    2          0.19
```




```
# You can view the observations who are in the negative zone
HC_sil
```

```
##       cluster neighbor    sil_width
##  [1,]       1        2  0.306600410
##  [2,]       1        2 -0.400443049
##  [3,]       1        2  0.123897257
##  [4,]       1        2 -0.282518906
##  [5,]       1        2  0.416371957
##  [6,]       1        4  0.467749938
##  [7,]       2        4  0.742196208
##  [8,]       1        2  0.517153688
##  [9,]       1        2  0.417879906
## [10,]       1        3  0.359498607
## [11,]       1        4  0.067613741
## [12,]       1        3  0.157373147
## [13,]       1        3  0.437414182
## [14,]       1        3  0.337651231
## [15,]       1        3  0.256420559
## [16,]       3        1  0.288198985
## [17,]       1        3  0.151273433
## [18,]       4        2  0.153293508
## [19,]       1        3 -0.124471150
## [20,]       1        2  0.478937361
## [21,]       2        1  0.681411586
## [22,]       4        1  0.227275445
## [23,]       1        2  0.009377918
## [24,]       1        3  0.285027180
## [25,]       1        3 -0.066918631
## [26,]       1        3  0.354396116
## [27,]       1        3  0.298376471
## [28,]       2        4  0.575489164
## [29,]       1        4  0.199449541
## [30,]       2        1  0.771988212
## [31,]       1        3  0.491632677
## [32,]       1        3  0.471626473
## [33,]       1        3  0.385682430
## [34,]       1        2  0.004043981
## [35,]       1        3  0.354221797
## [36,]       1        4  0.097742799
## [37,]       3        1  0.107525331
## [38,]       1        3 -0.127598327
## [39,]       1        3  0.036306781
## [40,]       1        2  0.256588604
## attr(,"Ordered")
## [1] FALSE
## attr(,"call")
## silhouette.default(x = cutree(u_df1_hc, k = 4), dist = u_df1_dist)
## attr(,"class")
## [1] "silhouette"
```

Define linkage methods, how to choose the best linkage method for HC

The hclust and agnes functions behave very similarly. However, the agnes function also provides the agglomerative coefficient, which measures the amount of clustering structure found (values closer to 1 suggest strong clustering structure).

This allows us to find certain hierarchical clustering methods that can identify stronger clustering structures.

```
linkmethod <- c( "average", "single", "complete", "ward")
names(linkmethod) <- c( "average", "single", "complete", "ward")

ac <- function(x) {
  agnes(u_df1_std, method = x)$ac
}

#calculate agglomerative coefficient for each clustering linkage method
sapply(linkmethod, ac)
```

```
##   average    single  complete      ward
## 0.7915791 0.7245369 0.8933284 0.9221749
```

```
# Which is the best linkage method?
```

Using Diana and Agnes function

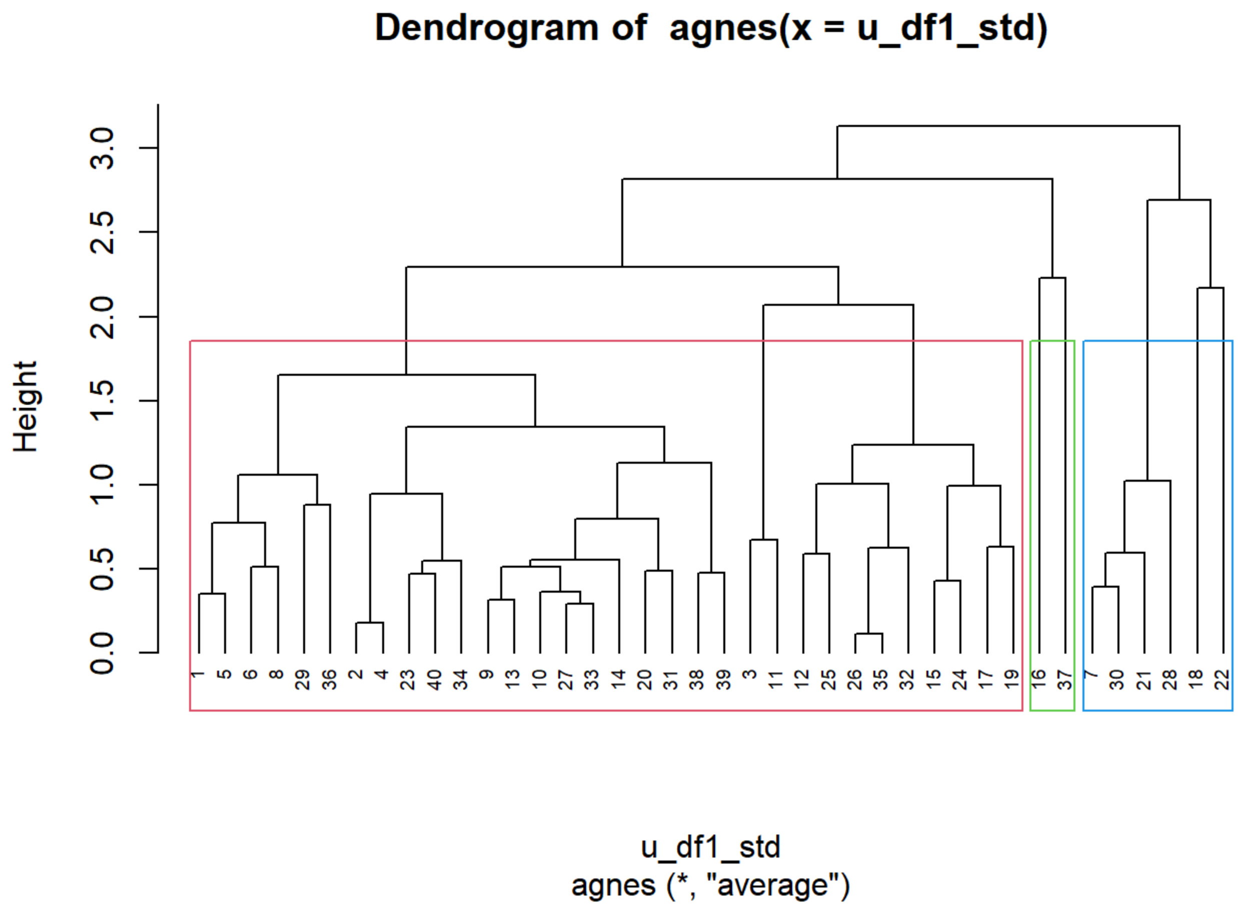
Top-down (Diana - divisive clustering), you would start with all the customers in a single cluster. Then you would recursively split the clusterrs into smaller subcluster based on certain criterias such as income or spending habits. For instance, you could split customers into high-income and low-income clusters, and then further divide each income group based on spending habits.

Bottom-up (Agnes - agglomerative clustering), you would start with each customer as an individual cluster. Then you would iteratively merge the most similar customer/cluster based on a distance metric, such as Euclidean distance.For example, you might merge customers with similar age range or similar spending habits.

```
# Agnes
## agnes(x, metric = “euclidean”, stand = FALSE, method = “average”)
HC_agnes <- agnes(u_df1_std)
HC_agnes$ac
```

```
## [1] 0.7915791
```

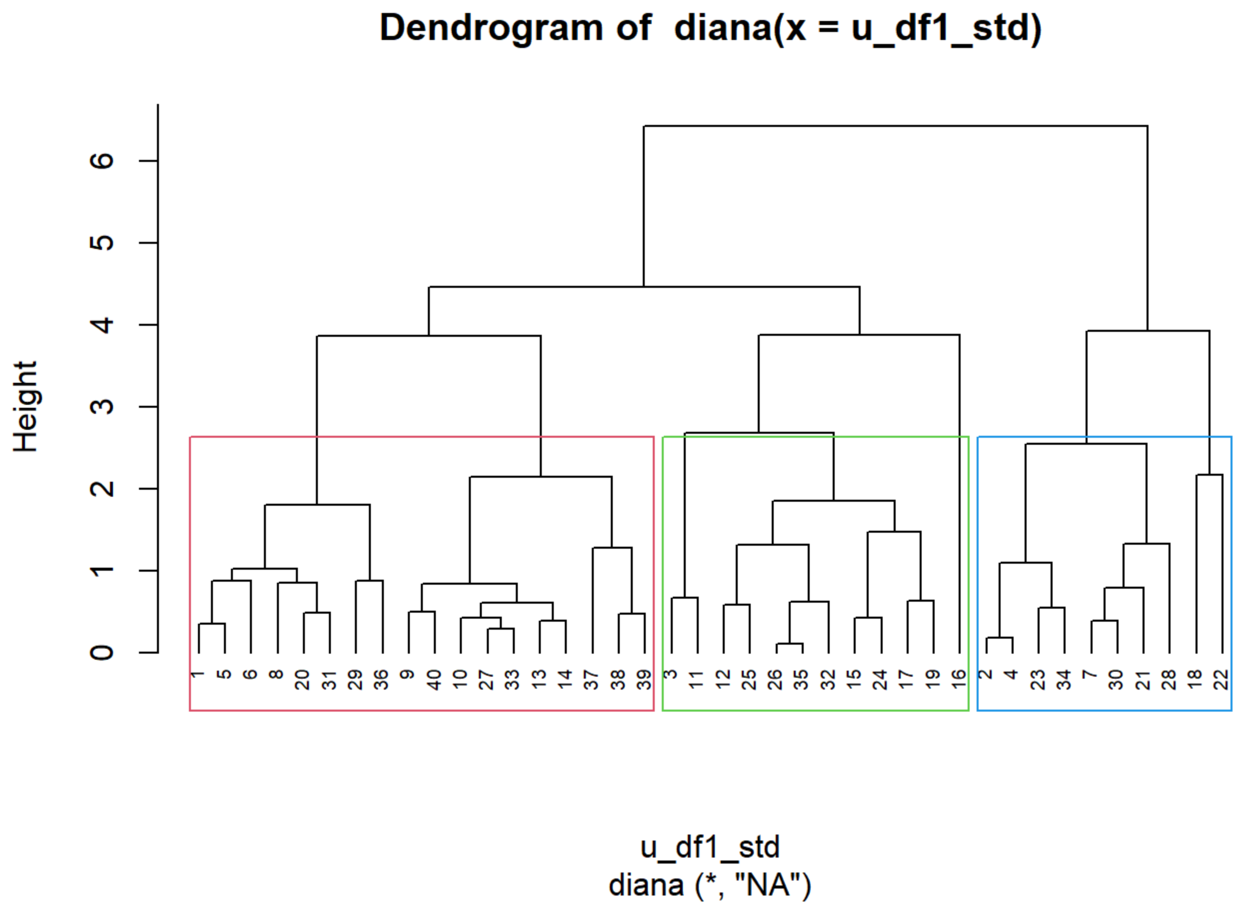
```
pltree(HC_agnes, cex=0.6, hang = -1)
rect.hclust(HC_agnes, k=3, border = 2:5)
```




```
# Diana
HC_diana <- diana(u_df1_std)
HC_diana$dc
```

```
## [1] 0.8898809
```

```
pltree(HC_diana, cex=0.6, hang = -1)
rect.hclust(HC_diana, k=3, border = 2:5)
```


Dendrogram of diana(x = u_df1_std)
Height
0
1
2
3
4
5
6
1 5 6 8 20 31 29 36 9 40 10 27 33 13 14 37 38 39 3 11 12 25 26 35 32 15 24 17 19 16 2 4 23 34 7 30 21 28 18 22
u_df1_std
diana (*, "NA")