Data Mining/Discovery Techniques

Segmentação de clientes e análise do cabaz de compras

Alunos Ivo Nunes n20202567, Gonçalo Bernardo n20202467, André Costa n20202601.
Docente João Caldeira*, jcaldeira@uatlantica.pt, **Univ. Atlântica**

21 January 2023

Contents

*Invited Professor. LinkedIn. https://www.linkedin.com/in/joao-carlos-caldeira/

1 Environment Setup

1.1 Install Required Libraries

```
install.packages("DataExplorer")
install.packages("Hmisc")
install.packages("BBmisc")
install.packages("vioplot")
install.packages("moments")
install.packages("readxl")
install.packages("datetime")
install.packages("lubridate")
install.packages("cluster")
install.packages("factoextra")
install.packages("gridExtra")
install.packages("purrr")
install.packages("arules")
install.packages("arulesViz")
install.packages("data.table")
```

1.2 Load Required Libraries

```
library(DataExplorer)
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(BBmisc)
```

```
##
## Attaching package: 'BBmisc'

## The following object is masked from 'package:Hmisc':
##
##     %nin%
```

```
## The following object is masked from 'package:base':
##
##     isFALSE
```

```
library(vioplot)
```

```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.7: type help(sm) for summary information
```

```
##
## Attaching package: 'sm'
```

```
## The following object is masked from 'package:BBmisc':
##
##     pause
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(moments)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:BBmisc':
##
##     coalesce, collapse
```

```
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(datetime)
library(lubridate)
```

```
## Loading required package: timechange
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(cluster)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(purrr)
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arulesViz)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:purrr':
##
##     transpose

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

2 Data Mining/Discovery

2.1 Loading the Dataset(s)

```
# Lê o ficheiro chamado "Online Retail.xlsx" e coloca os dados na variável "my_data".
# Em seguida, o comando "str (my_data)" descreve resumidamente os dados armazenados na variável "my_dat
# incluindo o número de observações e colunas, e os nomes e tipos de variáveis.
my_data <- read_excel("Online Retail.xlsx")
str(my_data)
```

```
## tibble [541,909 x 8] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : chr [1:541909] "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr [1:541909] "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr [1:541909] "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPI
##  $ Quantity   : num [1:541909] 6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct[1:541909], format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num [1:541909] 2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num [1:541909] 17850 17850 17850 17850 17850 ...
##  $ Country    : chr [1:541909] "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" .
```

2.2 Exploratory Data Analysis(EDA)

2.2.1 Getting to Know the Dataset

```
# Mostra as primeiras 10 linhas do dataset my_data.
head(my_data, 10)
```

```
## # A tibble: 10 x 8
##    InvoiceNo Stock~1 Descr~2 Quant~3 InvoiceDate         UnitP~4 Custo~5 Country
##    <chr>     <chr>   <chr>     <dbl> <dttm>                <dbl>   <dbl> <chr>
##  1 536365    85123A  WHITE ~       6 2010-12-01 08:26:00    2.55   17850 United~
##  2 536365    71053   WHITE ~       6 2010-12-01 08:26:00    3.39   17850 United~
##  3 536365    84406B  CREAM ~       8 2010-12-01 08:26:00    2.75   17850 United~
##  4 536365    84029G  KNITTE~       6 2010-12-01 08:26:00    3.39   17850 United~
##  5 536365    84029E  RED WO~       6 2010-12-01 08:26:00    3.39   17850 United~
##  6 536365    22752   SET 7 ~       2 2010-12-01 08:26:00    7.65   17850 United~
##  7 536365    21730   GLASS ~       6 2010-12-01 08:26:00    4.25   17850 United~
##  8 536366    22633   HAND W~       6 2010-12-01 08:28:00    1.85   17850 United~
##  9 536366    22632   HAND W~       6 2010-12-01 08:28:00    1.85   17850 United~
## 10 536367    84879   ASSORT~      32 2010-12-01 08:34:00    1.69   13047 United~
## # ... with abbreviated variable names 1: StockCode, 2: Description,
## #   3: Quantity, 4: UnitPrice, 5: CustomerID
```

```
# Mostra as últimas 10 linhas do dataset my_data.
tail(my_data, 10)
```

```
## # A tibble: 10 x 8
##    InvoiceNo Stock~1 Descr~2 Quant~3 InvoiceDate         UnitP~4 Custo~5 Country
##    <chr>     <chr>   <chr>     <dbl> <dttm>                <dbl>   <dbl> <chr>
##  1 581587    22726   ALARM ~       4 2011-12-09 12:50:00    3.75   12680 France
##  2 581587    22730   ALARM ~       4 2011-12-09 12:50:00    3.75   12680 France
##  3 581587    22367   CHILDR~       8 2011-12-09 12:50:00    1.95   12680 France
##  4 581587    22629   SPACEB~      12 2011-12-09 12:50:00    1.95   12680 France
##  5 581587    23256   CHILDR~       4 2011-12-09 12:50:00    4.15   12680 France
##  6 581587    22613   PACK O~      12 2011-12-09 12:50:00    0.85   12680 France
##  7 581587    22899   CHILDR~       6 2011-12-09 12:50:00    2.1    12680 France
##  8 581587    23254   CHILDR~       4 2011-12-09 12:50:00    4.15   12680 France
##  9 581587    23255   CHILDR~       4 2011-12-09 12:50:00    4.15   12680 France
## 10 581587    22138   BAKING~       3 2011-12-09 12:50:00    4.95   12680 France
## # ... with abbreviated variable names 1: StockCode, 2: Description,
## #   3: Quantity, 4: UnitPrice, 5: CustomerID
```

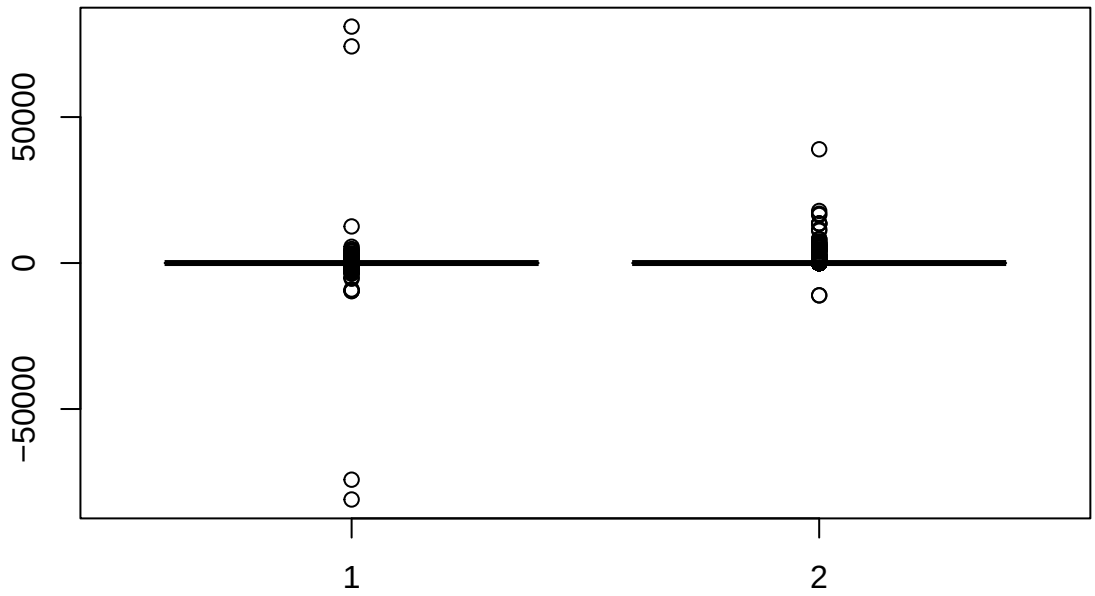
```
# Descrição resumida dos dados "my_data", incluindo estatísticas resumidas, como a média, mediana, desv
# variáveis numéricas, e contagem e frequência para variáveis categóricas.
summary(my_data)
```

```
##   InvoiceNo          StockCode         Description           Quantity
##  Length:541909      Length:541909      Length:541909      Min.   :-80995.00
##  Class :character   Class :character   Class :character   1st Qu.:     1.00
##  Mode  :character   Mode  :character   Mode  :character   Median :     3.00
##                                                           Mean   :     9.55
```

```
##                                                    3rd Qu.:    10.00
##                                                    Max.   : 80995.00
##
##   InvoiceDate                      UnitPrice           CustomerID
##  Min.   :2010-12-01 08:26:00.00   Min.   :-11062.06   Min.   :12346
##  1st Qu.:2011-03-28 11:34:00.00   1st Qu.:     1.25   1st Qu.:13953
##  Median :2011-07-19 17:17:00.00   Median :     2.08   Median :15152
##  Mean   :2011-07-04 13:34:57.16   Mean   :     4.61   Mean   :15288
##  3rd Qu.:2011-10-19 11:27:00.00   3rd Qu.:     4.13   3rd Qu.:16791
##  Max.   :2011-12-09 12:50:00.00   Max.   : 38970.00   Max.   :18287
##                                                       NA's   :135080
##    Country
##  Length:541909
##  Class :character
##  Mode  :character
##
##
##
##
```

```
# Visualizar os nomes das colunas.
plot_str(my_data)
```

```
# Verificação dos valores negativos.
boxplot(my_data$Quantity,my_data$UnitPrice)
```



3 Data Preprocessing

3.1 Handling Missing Data

3.1.1 Summary of Incomplete Cases(NAs)

```
# Contamos aqui todos os NA's existentes na Coluna de CustomerID.
sum(is.na(my_data$CustomerID))
```

```
## [1] 135080
```

3.2 Data Transformation

```
# Visto que vamos iniciar a manipulação e tratamento, deixaremos iremos guardar a variavel my_data com
# originais e atribuiremos a outra variavel os dados para podermos efetuar transformações.
# Removemos os NA's e confirmamos com o sum para verificar que a quantidade de NA's é 0.
dataset <- my_data
dataset <- na.omit(dataset)
sum(is.na(dataset$CustomerID))
```

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

```
# Verificamos a quantidade de valores negativos nas colunas Quantity e UnitPrice.
sum(dataset$Quantity<0)
```

```
## [1] 8905
```

```
sum(dataset$UnitPrice<=0.0)
```

```
## [1] 40
```

```
# Filtramos o conjunto de dados "dataset" onde a quantidade (Quantity) é maior que zero e o preço unitá
# Vai então remover as linhas onde a quantidade ou preço unitário é negativo ou zero.
# Depois executamos o sum para confirmar a operação.
dataset <- filter(dataset, dataset$Quantity > 0, dataset$UnitPrice > 0.0)
sum(dataset$Quantity<0)
```

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

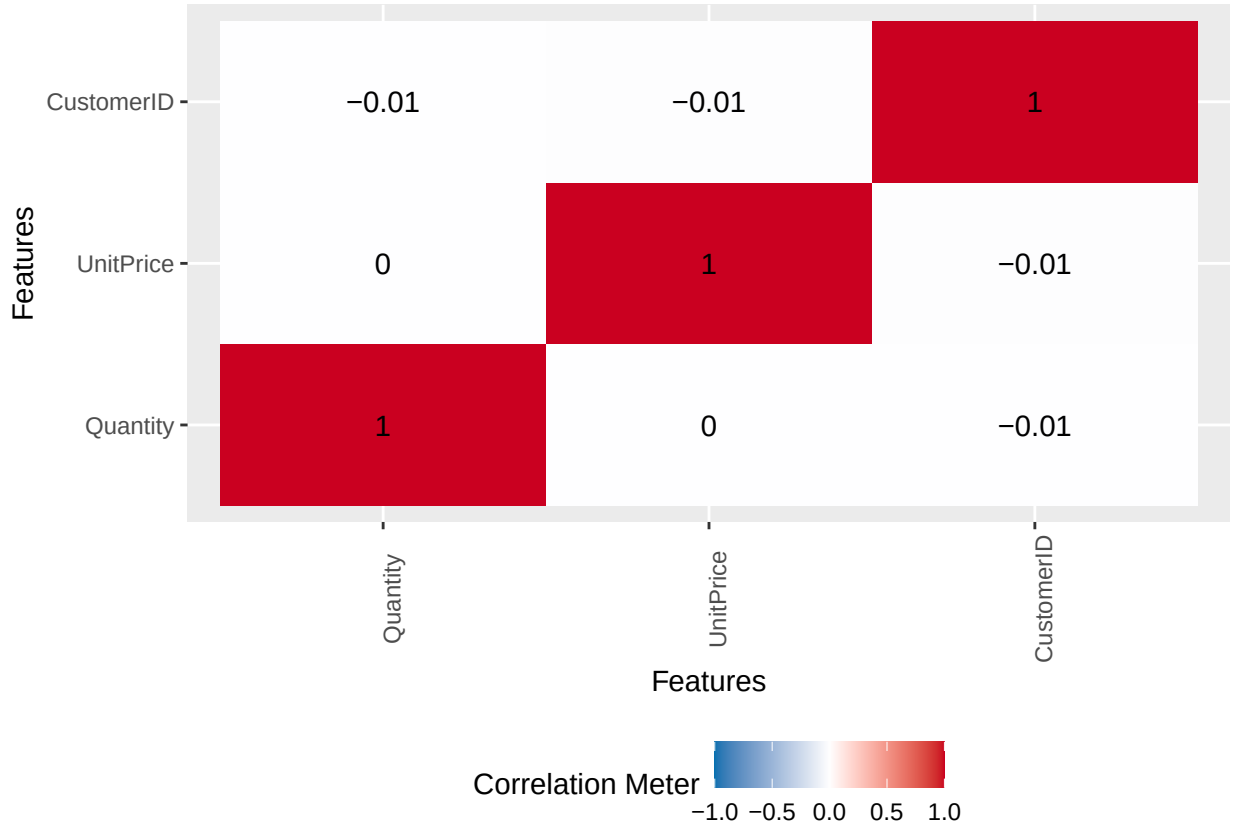
```
sum(dataset$UnitPrice<=0.0)
```

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

3.3 Using Correlation Heatmaps

```
# Verificar a correlação entre as variáveis do conjunto de dados do "dataset".
plot_correlation(dataset)
```

```
## Warning in dummify(data, maxcat = maxcat): Ignored all discrete features since
## 'maxcat' set to 20 categories!
```




```
# Confirmação de que não existem valores nulos.
any(is.null(dataset))
```

```
## [1] FALSE
```

```
# Remove todas as linhas do conjunto de dados que contenham valores ausentes (NA).
# A função "dim()" imprime o número de linhas e colunas do conjunto de dados "dataset".

dataset = na.omit(dataset)
```

```
# "unique()" para remover as linhas duplicadas do conjunto de dados "dataset".
dim(unique(dataset))[1]
```

```
## [1] 392692
```

```
# Cria uma nova coluna chamada "date", no conjunto de dados "dataset", extraída da coluna "InvoiceDate"
dataset$date <- sapply(as.character(dataset$InvoiceDate), FUN = function(x) {strsplit(x, split = '[ ]')
# Cria uma nova coluna chamada "date", no conjunto de dados "dataset", extraída da coluna "InvoiceDate"
dataset$time <- sapply(as.character(dataset$InvoiceDate), FUN = function(x) {strsplit(x, split = '[ ]')
# Cria uma nova coluna chamada "date", no conjunto de dados "dataset", extraída da coluna "InvoiceDate"
dataset$month <- sapply(as.character(dataset$InvoiceDate), FUN = function(x) {strsplit(x, split = '[-]'
# Cria uma nova coluna chamada "date", no conjunto de dados "dataset", extraída da coluna "InvoiceDate"
dataset$year <- sapply(as.character(dataset$InvoiceDate), FUN = function(x) {strsplit(x, split = '[-]')
# Cria uma nova coluna chamada "date", no conjunto de dados "dataset", extraída da coluna "InvoiceDate"
dataset$hourOfDay <- sapply(dataset$time, FUN = function(x) {strsplit(x, split = '[:]')[[1]][1]})

# Função "mutate()" adiciona "TotalSales" ao conjunto de dados "dataset" com os valores calculados, mul
dataset = mutate(dataset, TotalSales = Quantity*UnitPrice)
# Nova coluna chamada "dayOfWeek", no conjunto de dados "dataset", com o dia da semana extraído da colu
# "label = TRUE", o que retorna o dia da semana como um Char.
dataset$dayOfWeek <- wday(dataset$InvoiceDate)
```

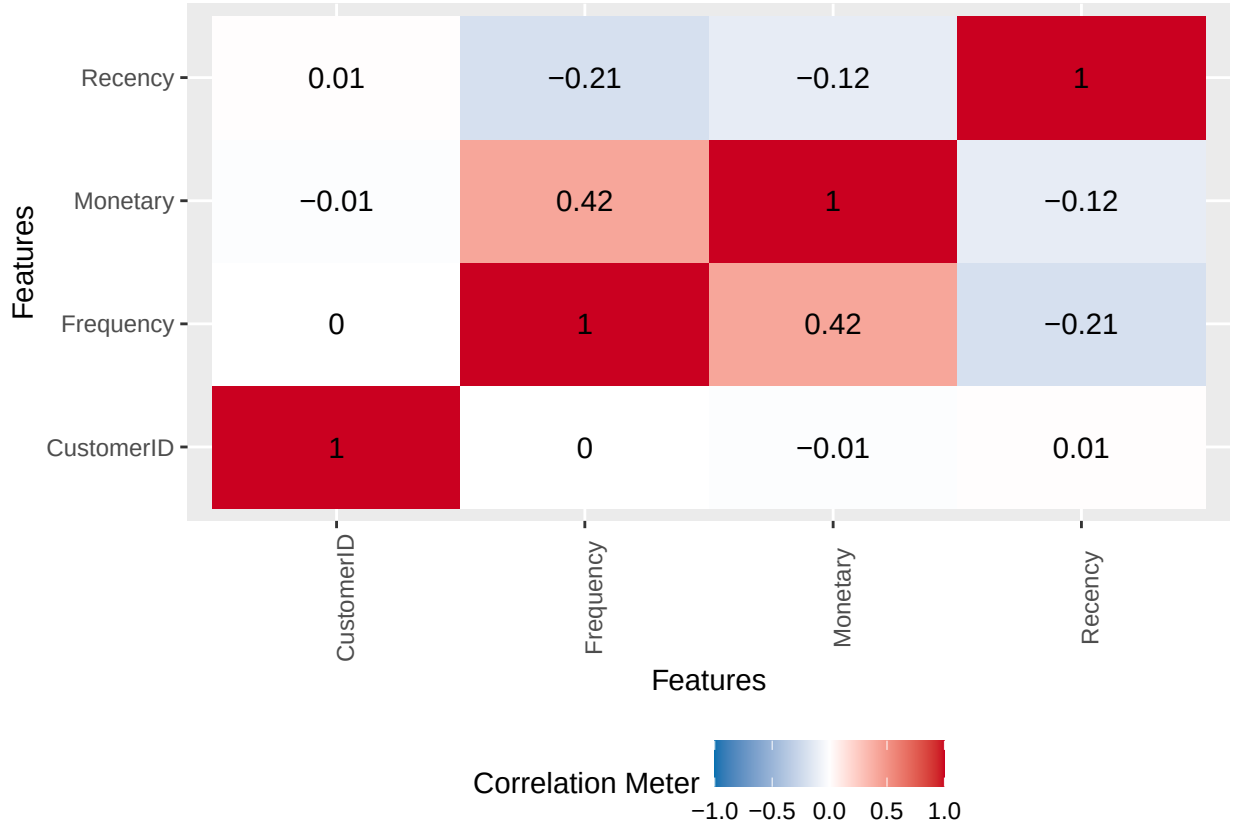
```
# Converter a coluna "Country", "month", "year", "hourOfDay" e "dayOfWeek" em fator, usando a função "a
dataset$Country <- as.factor(dataset$Country)
dataset$month <- as.factor(dataset$month)
dataset$year <- as.factor(dataset$year)
# Altera os níveis da coluna para somente 2010 e 2011 com "levels()".
levels(dataset$year) <- c(2010,2011)
hourOfDay <- as.factor(dataset$hourOfDay)
dataset$dayOfWeek <- as.factor(dataset$dayOfWeek)
```

```
# Criando uma variável chamada "max_date" e armazenando a data mais recente, presente na coluna "Invoic
max_date <- max(dataset$InvoiceDate, na.rm = TRUE)
# Função "mutate()" para adicionar uma nova coluna chamada "Diff" ao conjunto de dados "dataset", e pre
dataset = mutate(dataset, Diff = difftime(max_date, InvoiceDate, units = "days"))
# Função "floor()" para arredondar para baixo o valor de cada célula da coluna "Diff".
dataset$Diff <- floor(dataset$Diff)
```

```
# Recência, Frequência, Monetário = RFM
# Novo conjunto de dados chamado "RFM" que agrupa o conjunto de dados "dataset" por "CustomerID", utili
RFM <- summarise(group_by(dataset,CustomerID),Frequency = n(), Monetary = sum(TotalSales), Recency = mi
# Conversão da coluna "Recency" em tipo numérico, utilizando "as.numeric()".
RFM$Recency <- as.numeric(RFM$Recency)
# Substituindo todos os valores ausentes (NA) na coluna "Monetary" por 0, utilizando "RFM$Monetary[is.n
RFM$Monetary[is.na(RFM$Monetary)] <- 0
# "summary()" Resumo estatístico das colunas "Frequency", "Monetary" e "Recency".
summary(RFM)
```

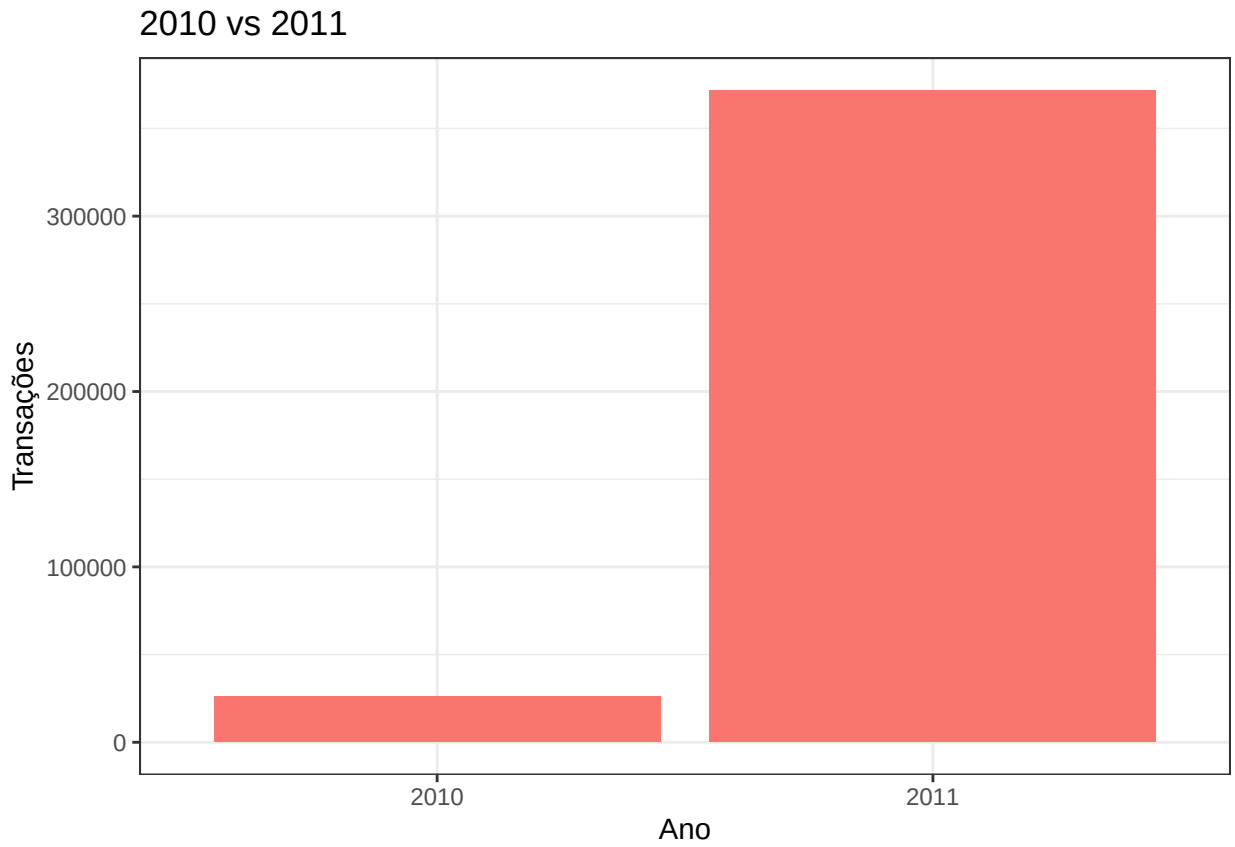
```
##    CustomerID      Frequency          Monetary            Recency
##  Min.   :12346   Min.   :   1.00   Min.   :     3.75   Min.   :  0.00
##  1st Qu.:13813   1st Qu.:  17.00   1st Qu.:   307.42   1st Qu.: 17.00
##  Median :15300   Median :  41.00   Median :   674.48   Median : 50.00
##  Mean   :15300   Mean   :  91.72   Mean   :  2054.27   Mean   : 91.54
##  3rd Qu.:16779   3rd Qu.: 100.00   3rd Qu.:  1661.74   3rd Qu.:141.00
##  Max.   :18287   Max.   :7847.00   Max.   :280206.02   Max.   :373.00
```

```
# Gráfico de correlação entre Frequency,  Monetary e  Recency.
plot_correlation(RFM)
```




```
# Criação de um gráfico de barras intitulado "2010 vs 2011", com o dataset especificado. O eixo "X" rep
ggplot(dataset, aes(year)) + geom_bar(aes(fill = "year")) + labs(title = "2010 vs 2011", x = "Ano", y =
```

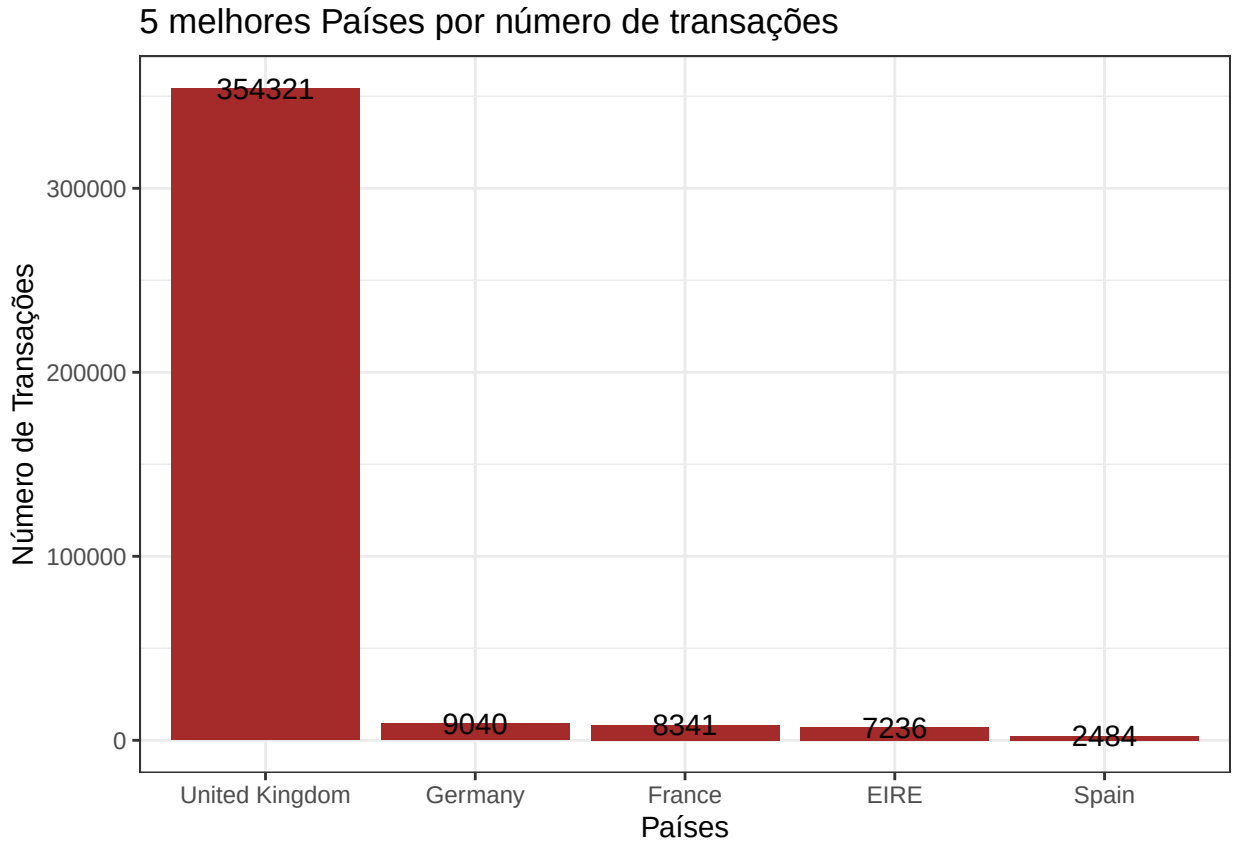
```
## Warning: The '<scale>' argument of 'guides()' cannot be 'FALSE'. Use "none" instead as
## of ggplot2 3.3.4.
```



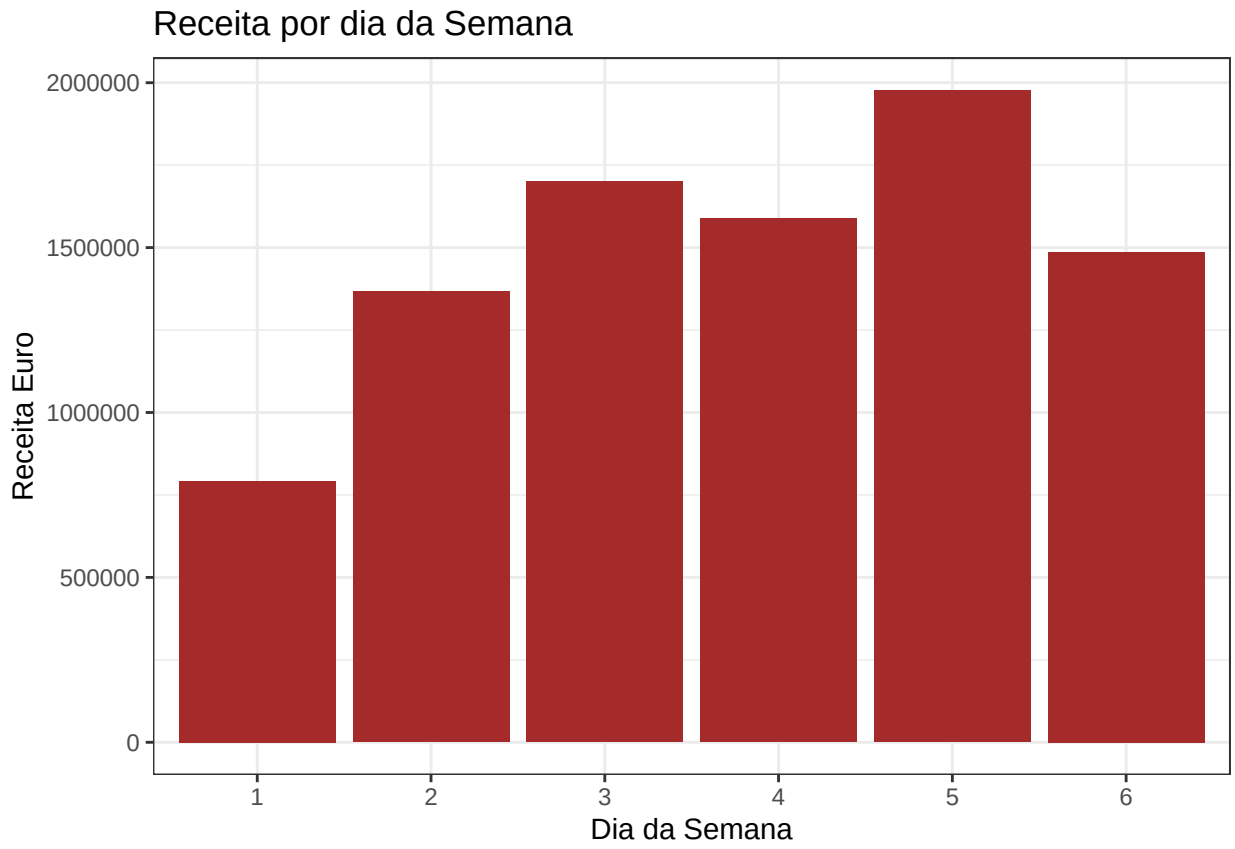

```
# Criação da tabela "Transactions_per_Country". Primeiro é aplicado o group_by (dataset, Country) para
# Em seguida, é aplicado o summarise para contar o número de transações por país.
# Depois, é aplicado o arrange para ordenar a tabela de forma decrescente, pelo número de transações.
# Por fim, é aplicado o top_n para selecionar os 10 países com o maior número de transações, sendo as c
Transactions_per_Country <- top_n(arrange(summarise(group_by(dataset, Country), 'Number of Transcations
```

```
## Selecting by Number of Transcations
```

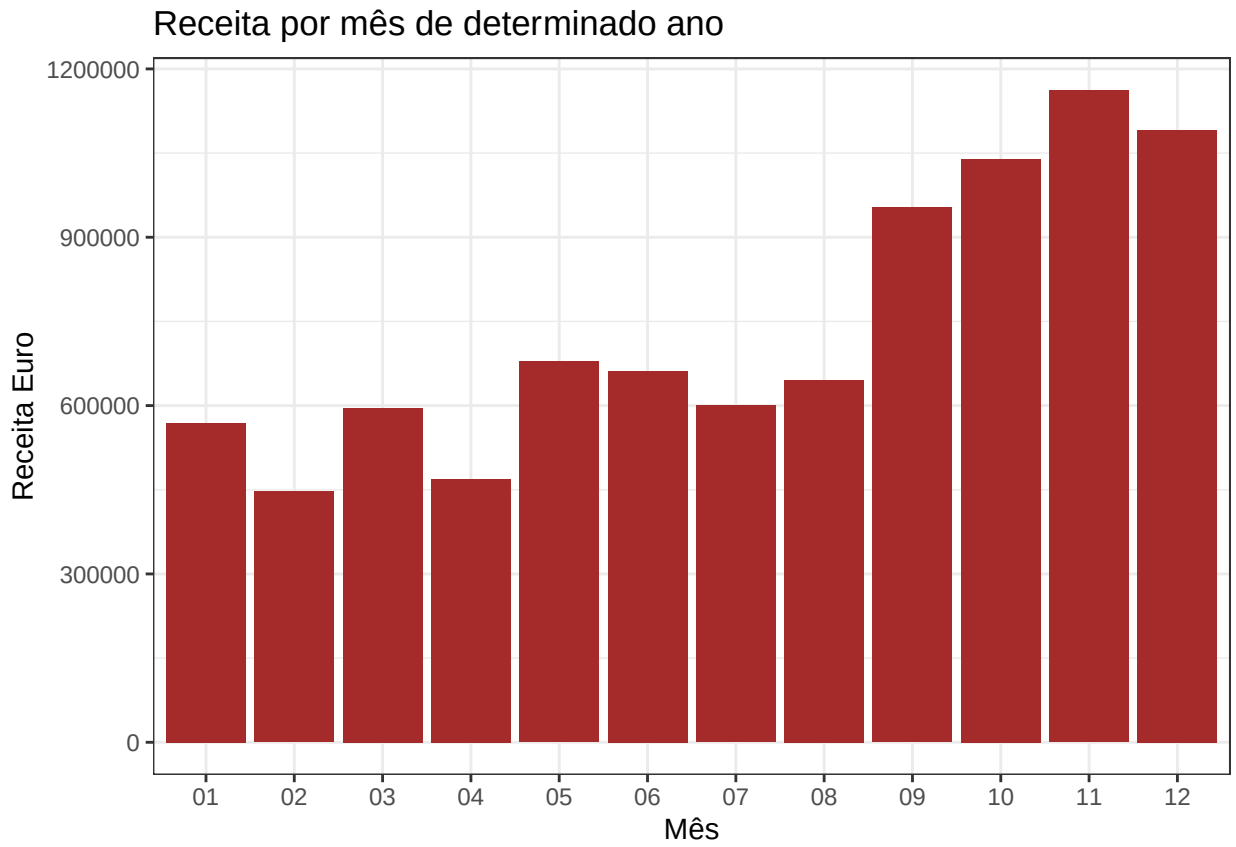
```
names(Transactions_per_Country) <- c("Country", "Number of Transactions")

# Criação do gráfico de barras  "5 melhores Países por número de transações", com os rótulos "Países" e
# respetivamente, exibindo os 5 primeiros países com o maior número de transações.
Transaction_per_Country_plot <- ggplot(head(Transactions_per_Country,5), aes(x = reorder(Country,-`Numb
  geom_text(aes(label = `Number of Transactions`)) +
  ggtitle('5 melhores Países por número de transações') + xlab('Países') +
  ylab('Número de Transações') +
  theme_bw()
print(Transaction_per_Country_plot)
```



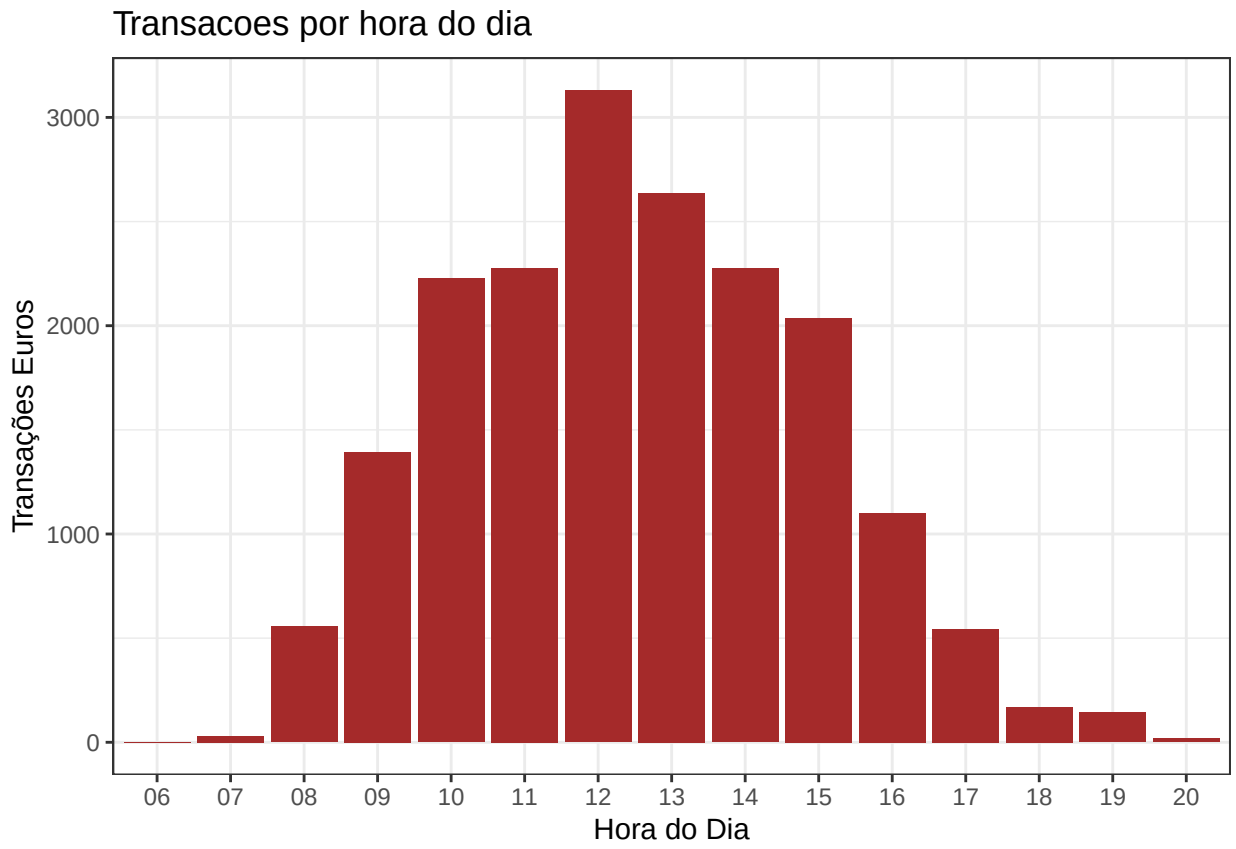

```
# Criação do gráfico de barras  "Receita por dia da Semana", com os rótulos "Dia da Semana" e "Receita
# Primeiro, é aplicado o group_by(dataset, dayOfWeek) para agrupar as transações por dia da semana.
# Depois, é aplicado o summarise para somar as vendas totais, por dia da semana.
# Aplicando o ggplot, o gráfico é criado. Aplicando o geom_bar as barras do gráfico são desenhadas e o
# O objetivo deste gráfico é mostrar a receita total, por dia da semana, ajudando desta forma a percebe
suppressWarnings(ggplot(summarise(group_by(dataset, dayOfWeek), revenue = sum(TotalSales)), aes(x = day
```




```
# Criação do gráfico de barras "Receita por mês de determinado ano", com os rótulos "Mês" e "Receita (€
# Primeiro, é aplicado o group_by(dataset, month) para agrupar as transações por mês.
# Depois, é aplicado o summarise para somar as vendas totais por mês.
# Aplicando o ggplot, o gráfico é criado. Aplicando o geom_bar as barras do gráfico são desenhadas e o
# Este gráfico mostra a receita total, por mês, de determinado ano, que ajuda a entender quais os meses
suppressWarnings(ggplot(summarise(group_by(dataset, month), revenue = sum(TotalSales)), aes(x = month,
```

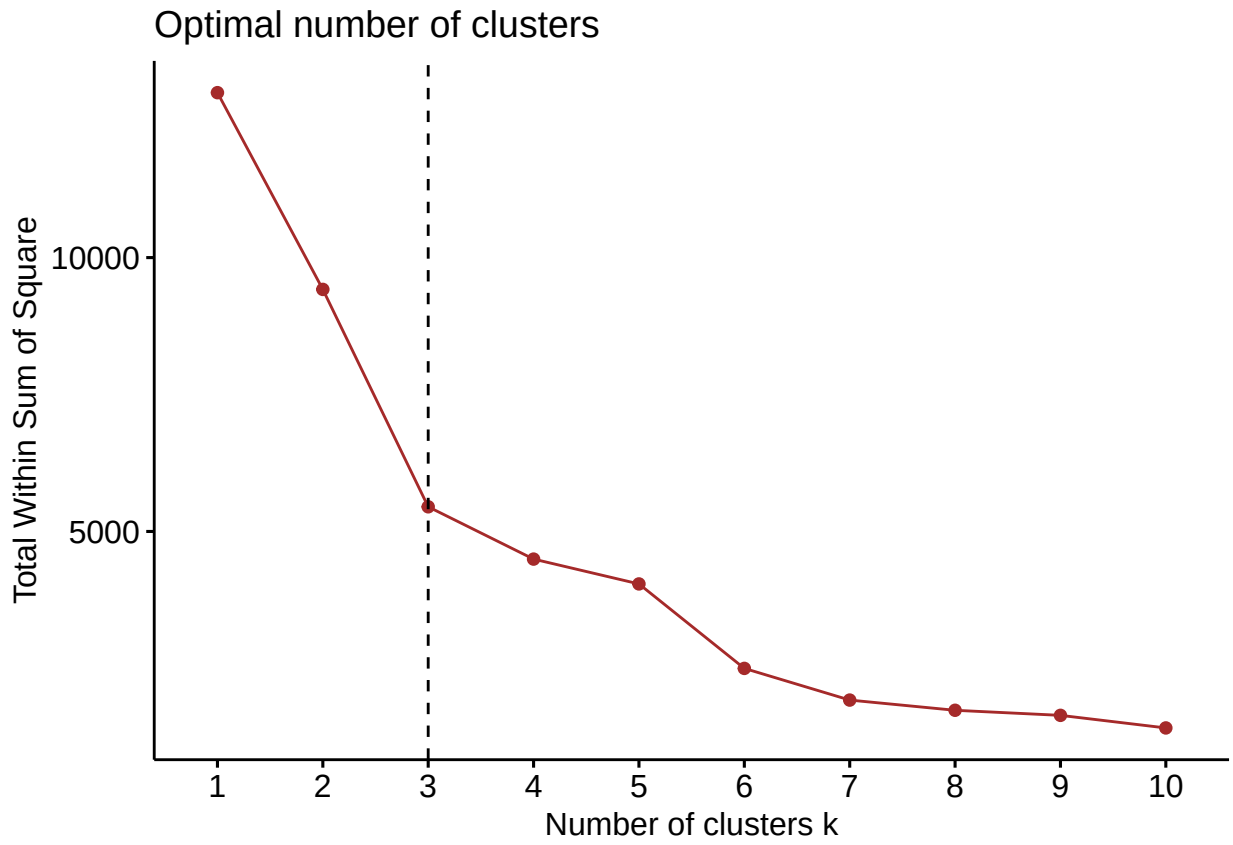



```
# Criação do gráfico de barras "Transacoes por hora do dia", com os rótulos "Hora do Dia" e "Transações
# Primeiro, é aplicado o group_by(dataset, hourOfDay) para agrupar as transações por hora do dia.
# Depois, é aplicado o summarise para contar o número de transações distintas, por hora do dia, utiliza
# Aplicando o ggplot, o gráfico é criado. Aplicando o geom_bar as barras do gráfico são desenhadas e o
# O objetivo deste gráfico é mostrar o número de transações distintas, por hora do dia, o que ajuda a e
suppressWarnings(ggplot(summarise(group_by(dataset, hourOfDay), transactions = n_distinct(InvoiceNo)), a
```

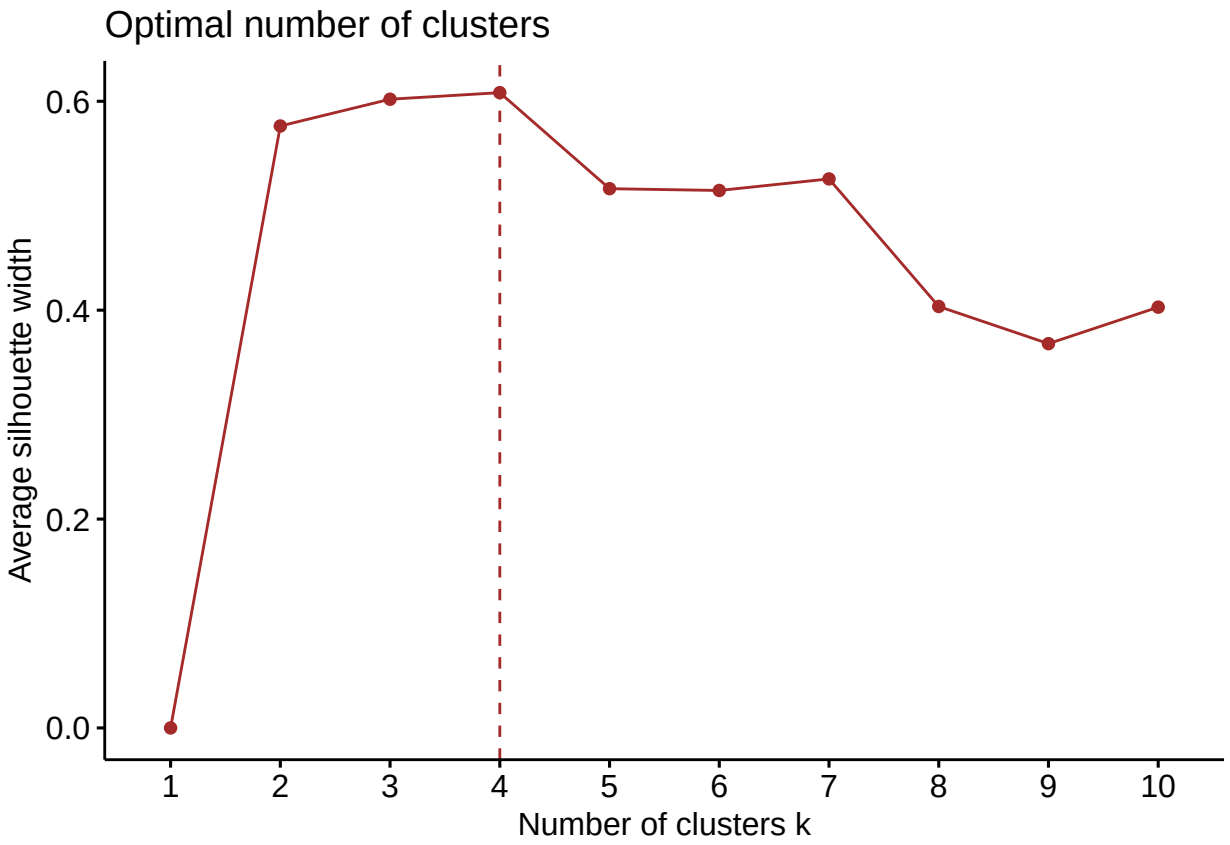


3.4 Cluster Analysis

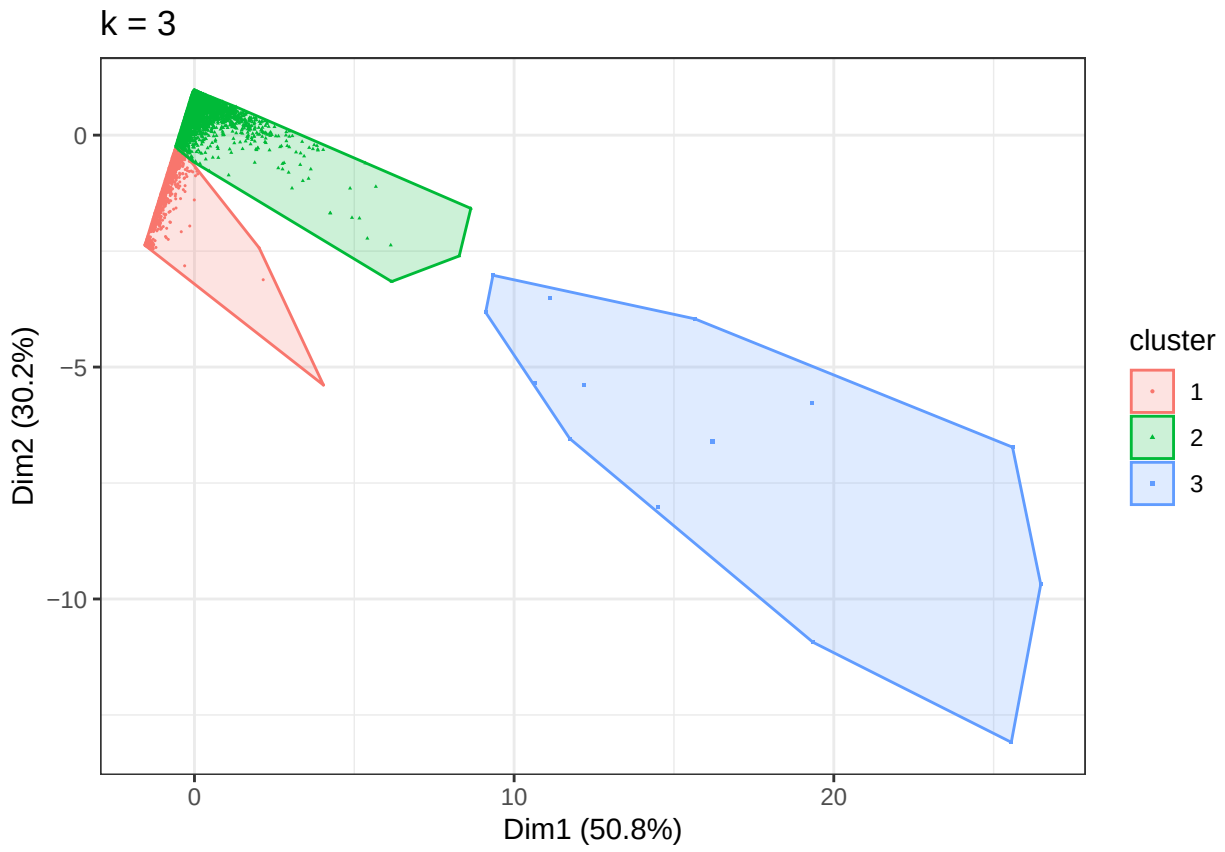
```
# Criação do dataframe "RFM", a partir de uma variável "RFM" já existente.
# De seguida, são definidos os nomes das linhas do dataframe, como o valor da coluna "CustomerID" do me
# Por fim, os valores do dataframe são escalados e o resultado é guardado num novo dataframe, chamado "
RFM <- data.frame(RFM)
row.names(RFM) <- RFM$CustomerID
RFM <- RFM[,-1]
RFM_scaled <- scale(RFM)
RFM_scaled <- data.frame(RFM_scaled)

# Utilização da função "fviz_nbclust", para visualizar o número ideal de clusters para o dataframe "RFM
# É adicionada uma linha vertical ao gráfico, na posição "x = 3", que é utilizada como referência para
fviz_nbclust(RFM_scaled, kmeans, method = "wss", linecolor = "Brown") + geom_vline(xintercept = 3, line
```



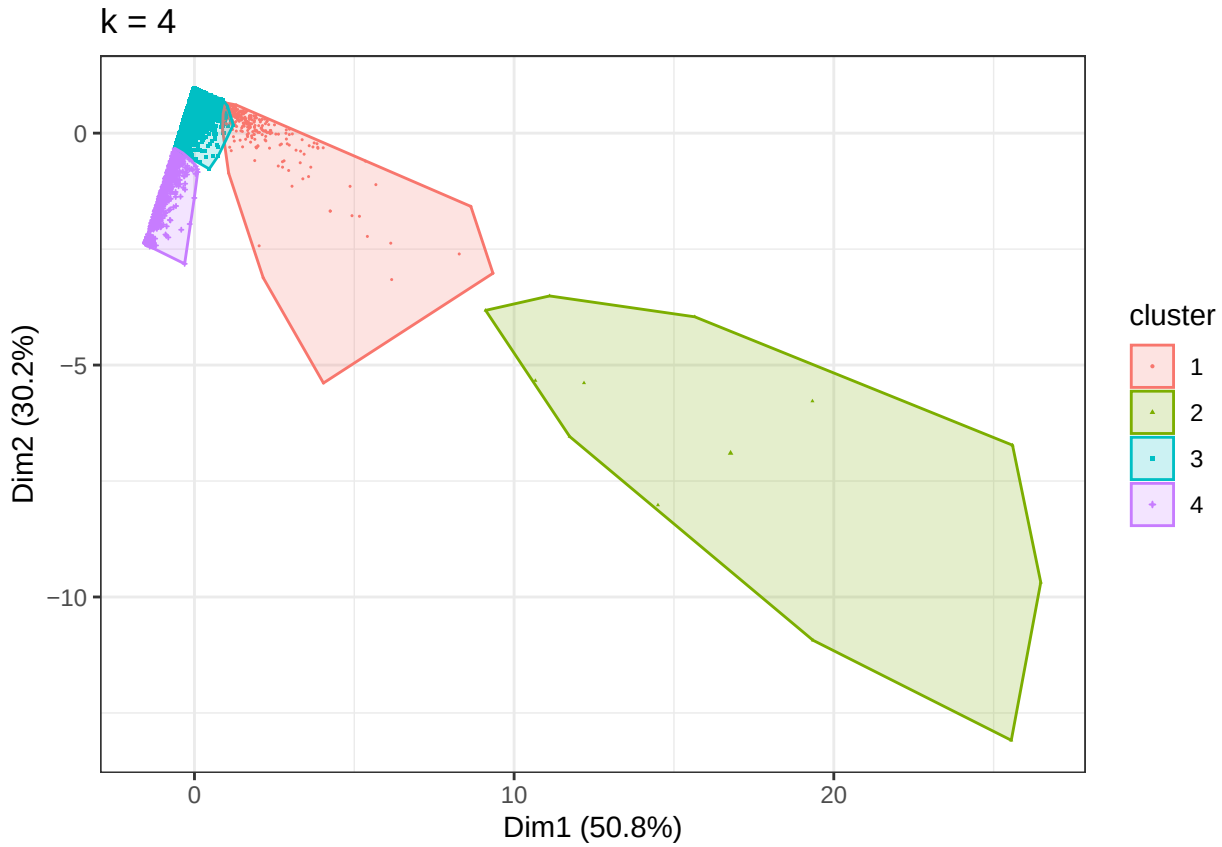

```
# Utilização da função "fviz_nbclust", para visualizar o grupo de dados, juntamente com o algoritmo "km
# O método de avaliação utilizado é o "silhouette".
fviz_nbclust(RFM_scaled, kmeans, method = "silhouette", linecolor = "Brown")
```




```r
# Utilização da função "kmeans" para agrupar os dados "RFM_scaled" em 3 clusters. O número de vezes que
k3 <- kmeans(RFM_scaled, centers = 3, nstart = 25)
# Utilização da função "kmeans" para agrupar os dados "RFM_scaled" em 2 clusters. O número de vezes que
k2 <- kmeans(RFM_scaled, centers = 2, nstart = 25)
# Utilização da função "kmeans" para agrupar os dados "RFM_scaled" em 4 clusters. O número de vezes que
k4 <- kmeans(RFM_scaled, centers = 4, nstart = 25)

# Utilização da função "fviz_cluster" para visualizar o grupo de dados armazenado na variável "k3". É u
# O dataframe utilizado é o "RFM_scaled" e o tamanho dos pontos é definido como 0,2. Além disso, é adic
fviz_cluster(k3, geom = "point", data = RFM_scaled, pointsize = 0.2) + ggtitle("k = 3") + theme_bw()
```

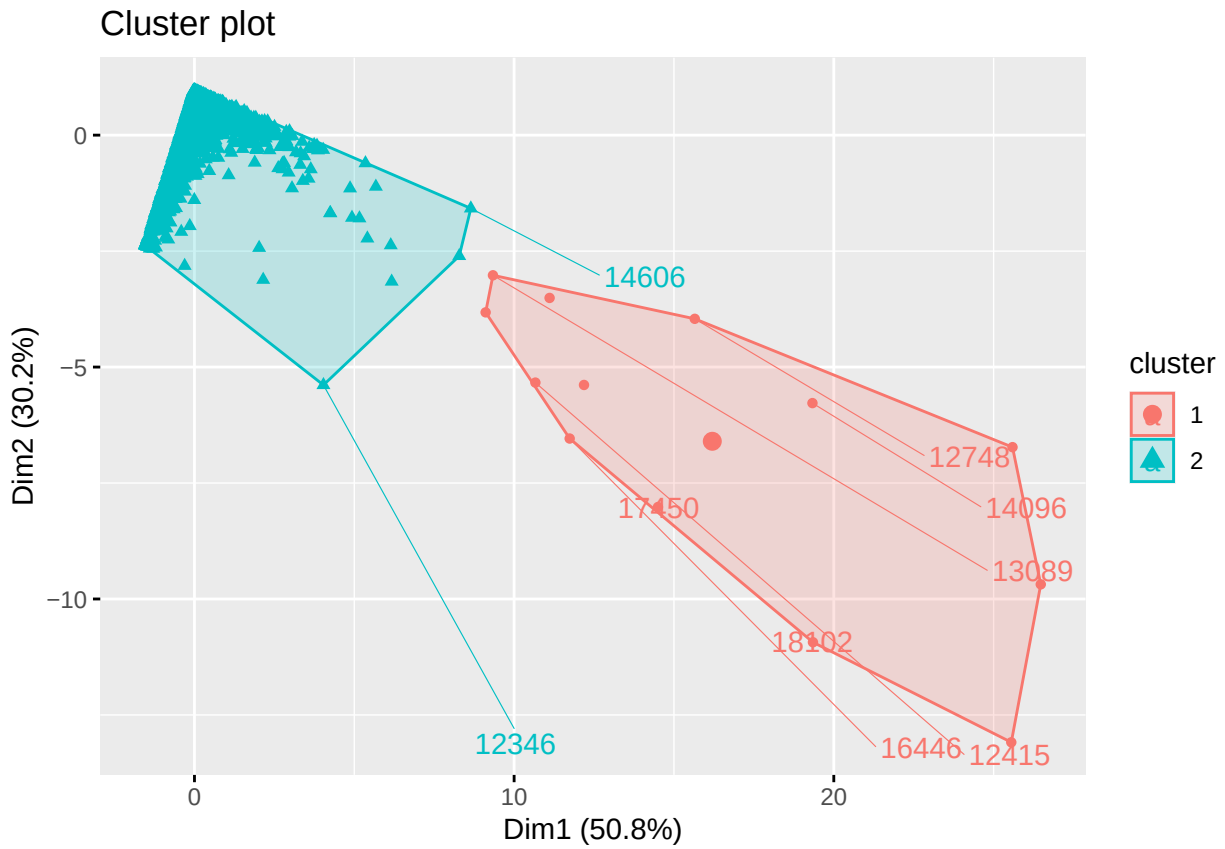



```
# Utilização da função "fviz_cluster" para visualizar o grupo de dados armazenado na variável "k4". É u
# O dataframe utilizado é o "RFM_scaled" e o tamanho dos pontos é definido como 0,2. Além disso, é adic
fviz_cluster(k4, geom = "point", data = RFM_scaled, pointsize = 0.2) + ggtitle("k = 4") + theme_bw()
```



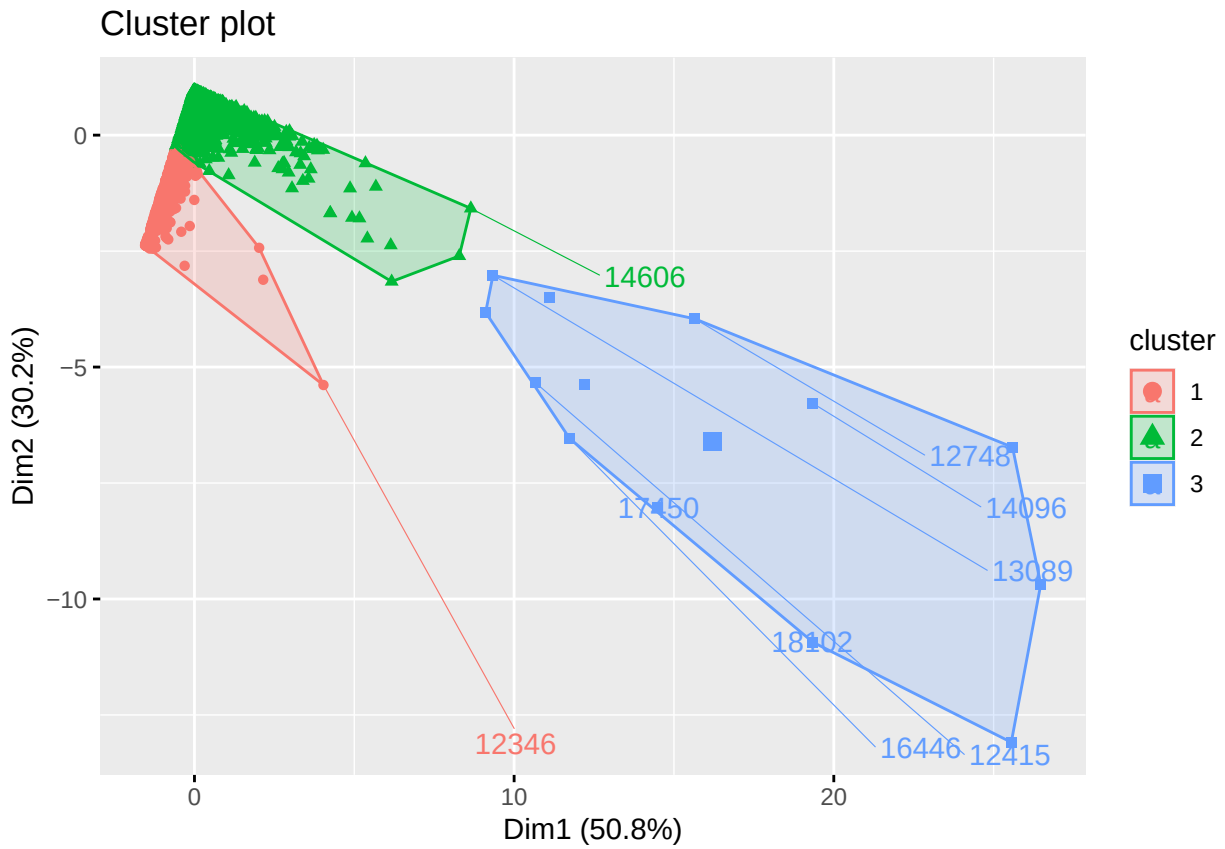

```
# Utilização da função "fviz_cluster" para visualizar o grupo de dados guardado na variável "k2", utili
# O tipo de elipse utilizado para representar os clusters é o "convex", e o argumento "repel" está defi
fviz_cluster(k2, data=RFM_scaled, ellipse.type = "convex", repel = T)
```

```
## Warning: ggrepel: 4329 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



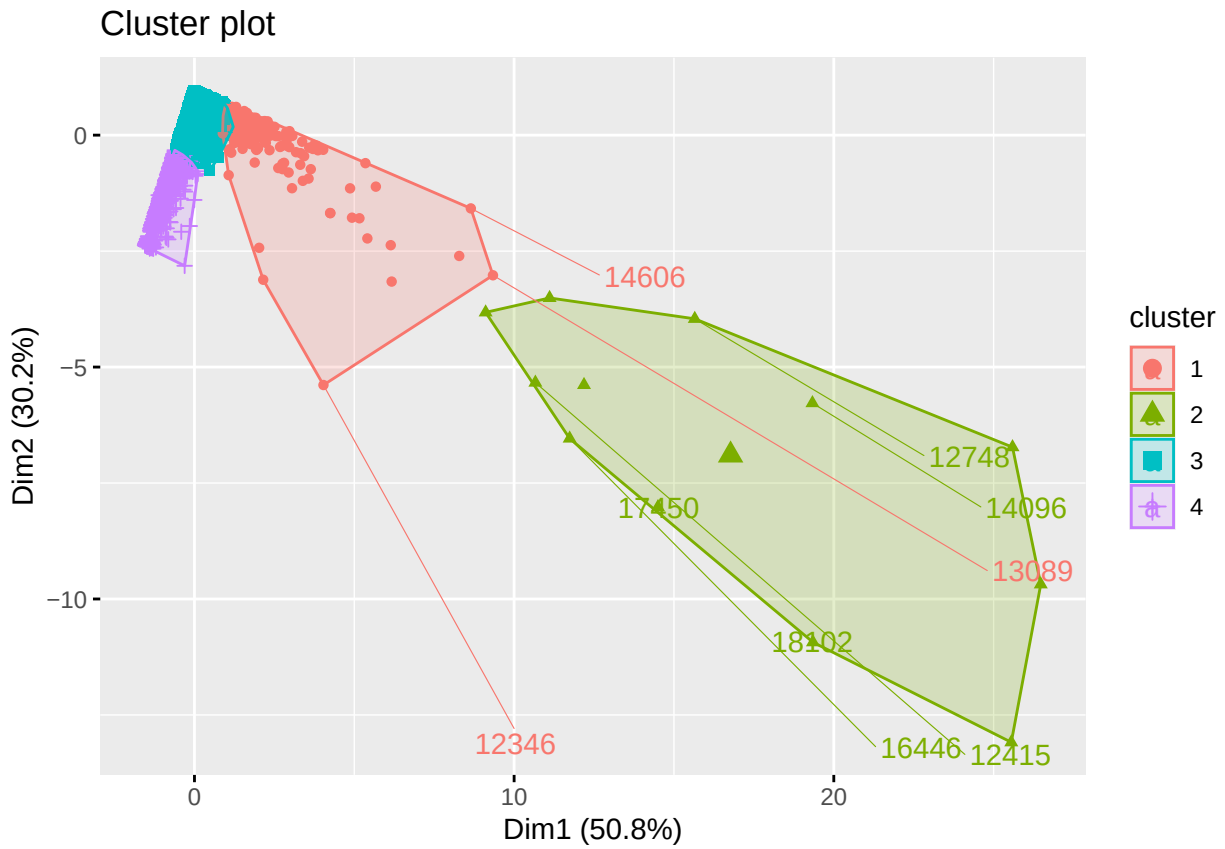

```
# Utilização da função "fviz_cluster" para visualizar o grupo de dados guardado na variável "k3", utili
# O tipo de elipse utilizado para representar os clusters é o "convex", e o argumento "repel" está defi
fviz_cluster(k3, data=RFM_scaled, ellipse.type = "convex", repel = T)
```

```
## Warning: ggrepel: 4329 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



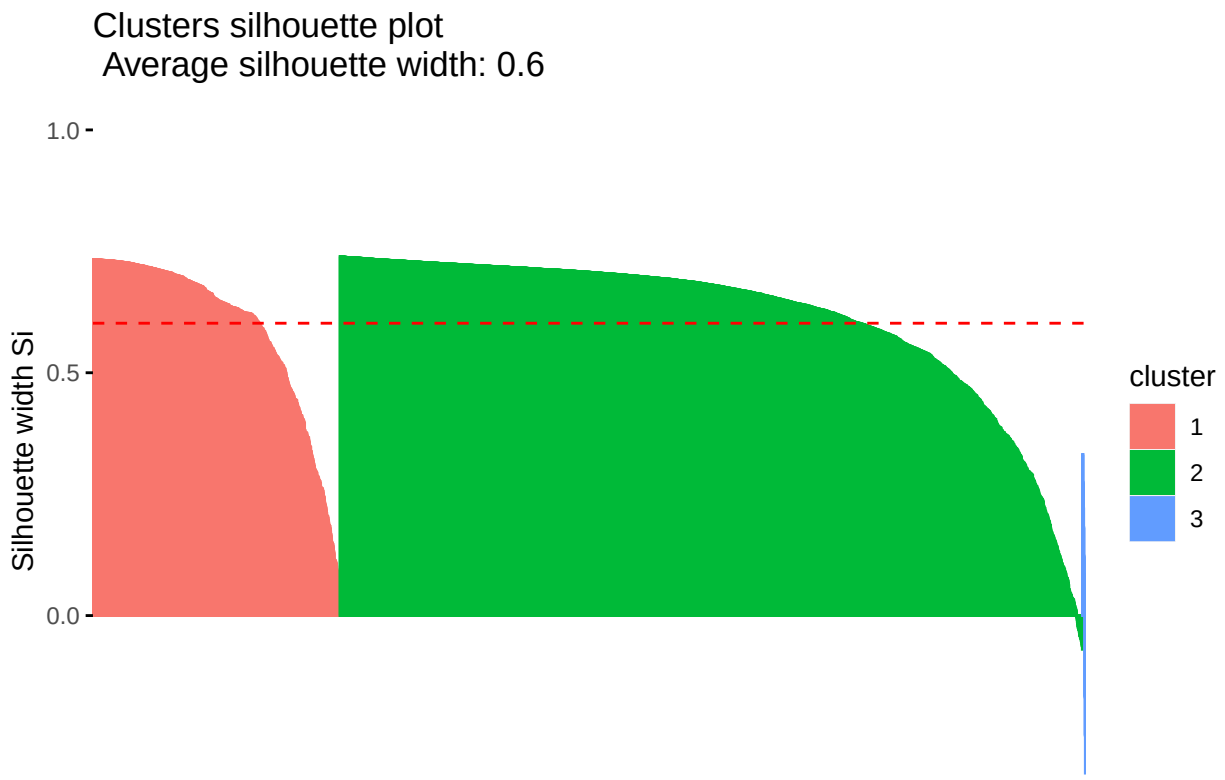

```
# Utilização da função "fviz_cluster" para visualizar o grupo de dados guardado na variável "k4", utili
# O tipo de elipse utilizado para representar os clusters é o "convex", e o argumento "repel" está defi
fviz_cluster(k4, data=RFM_scaled, ellipse.type = "convex", repel = T)
```

```
## Warning: ggrepel: 4329 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



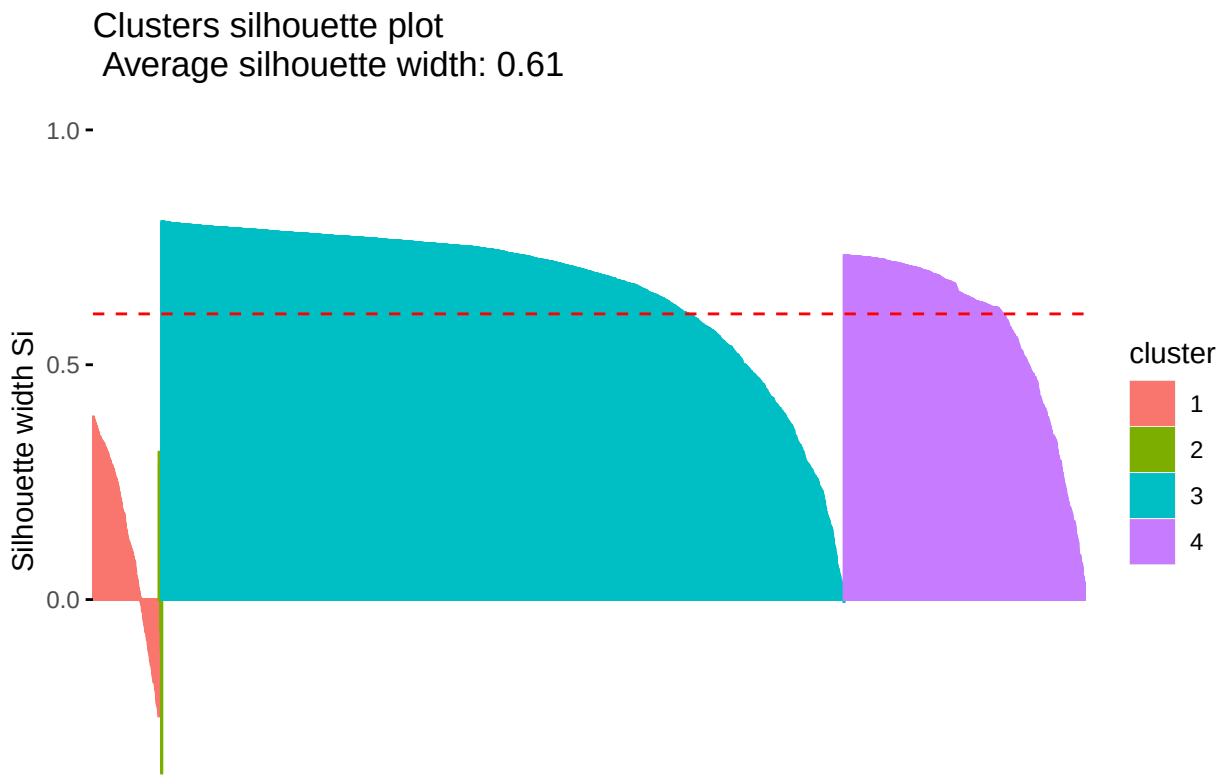

```
# Utilização da função "silhouette", para calcular a medida da silhueta para cada ponto do dado, compar
# Utilização da função "fviz_silhouette" para visualizar esses valores de silhueta. O resultado é uma v

# Avaliar e visualizar o quão bem os dados do dataframe "RFM_scaled" estão agrupados no modelo armazena
sil <- silhouette(k3$cluster, dist(RFM_scaled))
fviz_silhouette(sil)
```

```
##   cluster size ave.sil.width
## 1       1 1080          0.59
## 2       2 3245          0.61
## 3       3   13          0.09
```



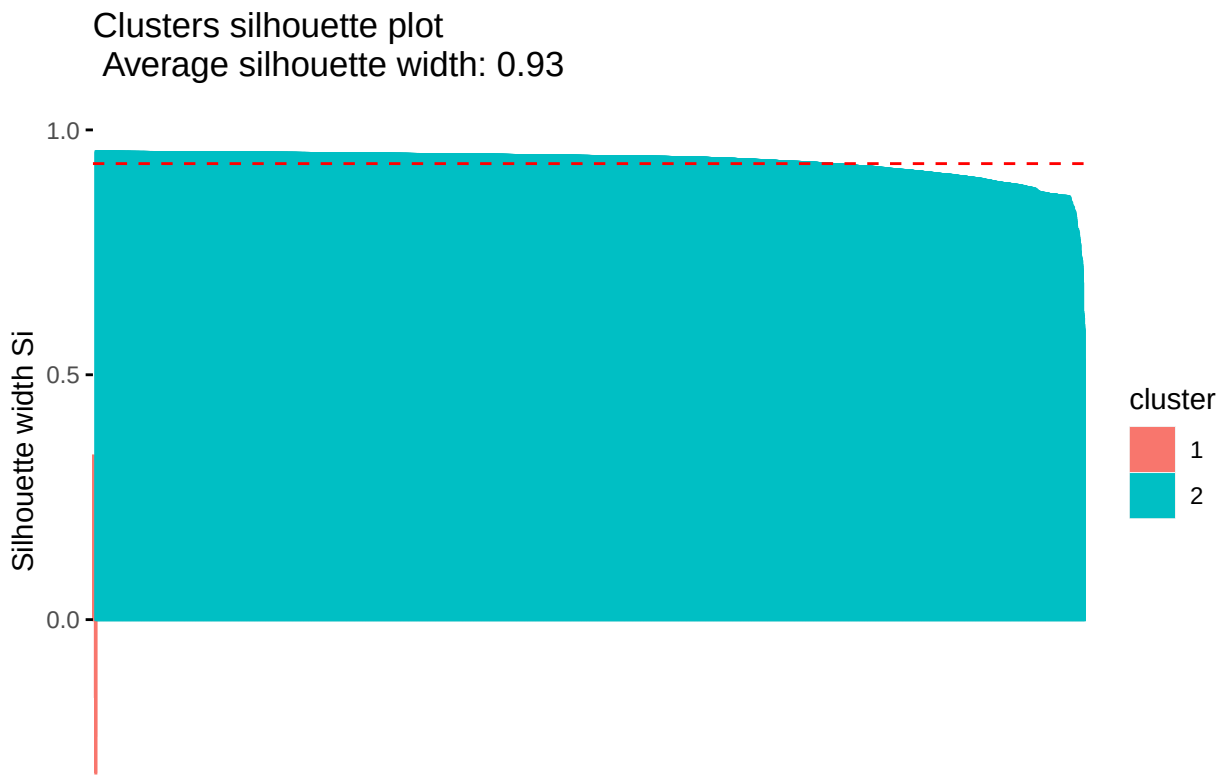

```
# Avaliar e visualizar o quão bem os dados do dataframe "RFM_scaled" estão agrupados no modelo armazena
sil <- silhouette(k4$cluster, dist(RFM_scaled))
fviz_silhouette(sil)
```

```
##   cluster size ave.sil.width
## 1       1  289          0.13
## 2       2   12          0.04
## 3       3 2982          0.67
## 4       4 1055          0.58
```

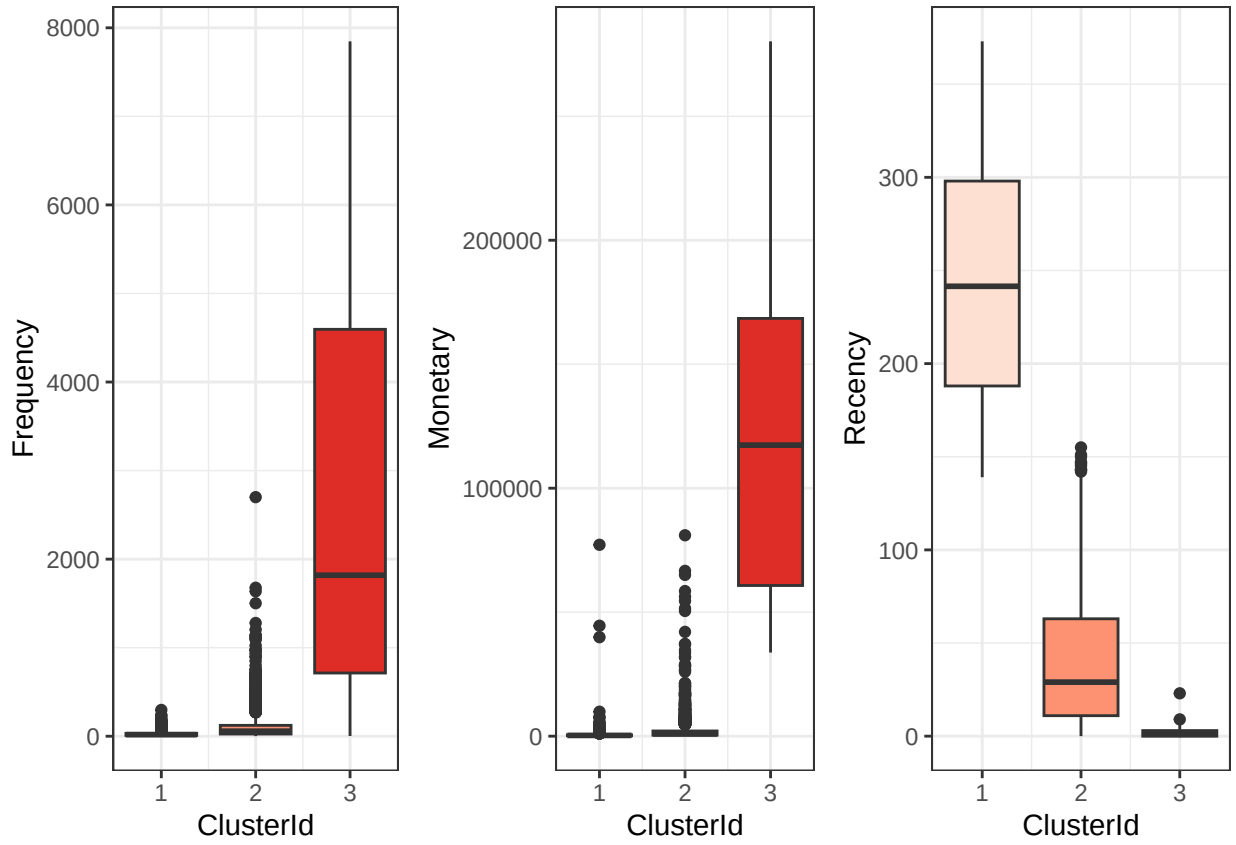



```
# Avaliar e visualizar o quão bem os dados do dataframe "RFM_scaled" estão agrupados no modelo armazena
sil <- silhouette(k2$cluster, dist(RFM_scaled))
fviz_silhouette(sil)
```

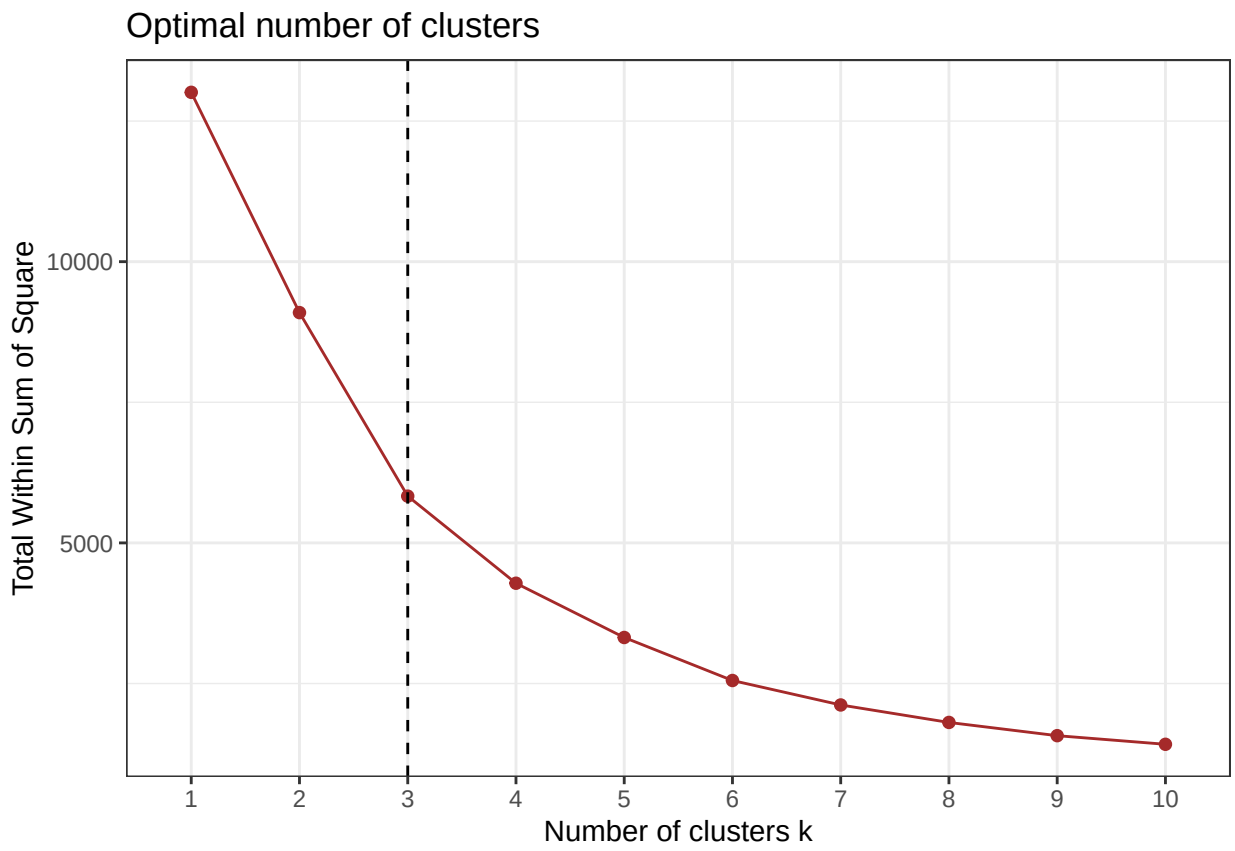
```
##   cluster size ave.sil.width
## 1       1   13          0.09
## 2       2 4325          0.93
```



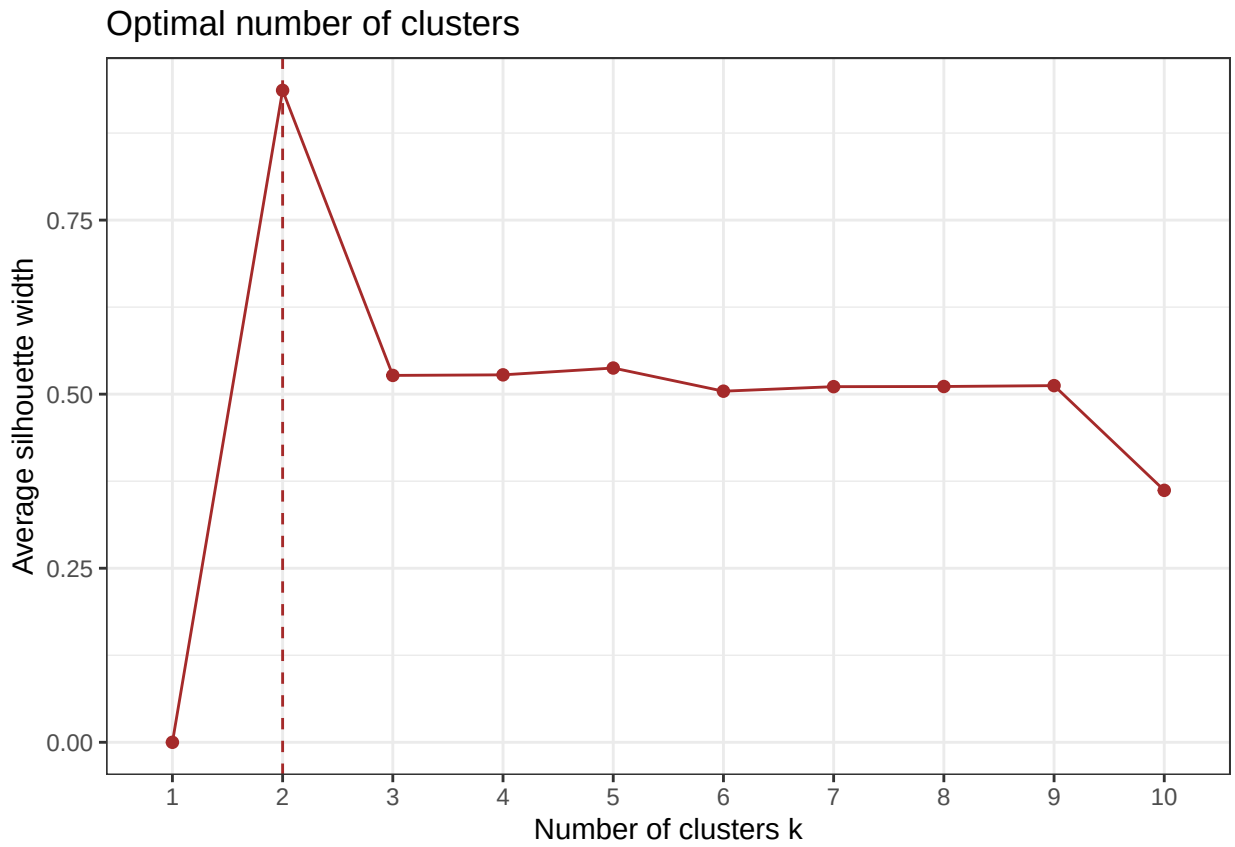

```r
# Criação de uma nova tabela com os dados do dataframe "RFM". É adicionada uma coluna "ClusterId" com o

# A função "cbind" é utilizada para combinar as colunas do dataframe "RFM" e os identificadores de clus
res <- cbind(RFM, ClusterId = k3$cluster)
# A função "as.data.frame" é utilizada para converter a tabela combinada para um dataframe. A saída é g
res <- as.data.frame(res)
# Utilização da biblioteca "ggplot2" para criar um gráfico, que compara as três diferentes colunas abai
# O argumento "group" é utilizado para agrupar os dados, de acordo com o "ClusterId", e o argumento "fi
# A função "geom_boxplot" é utilizada para criar o gráfico, em formato de caixa, e o argumento "show.le
# A função "scale_fill_brewer" é utilizada para definir o conjunto de cores "Reds" para os clusters.

# A função "ggplot" é utilizada para especificar o dataframe "res" como fonte de dados, e mapear as col
a <- ggplot(res, aes(x = ClusterId, y = Frequency, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_bw() + scale_fill_brewer(palette = "Reds")
# A função "ggplot" é utilizada para especificar o dataframe "res" como fonte de dados, e mapear as col
b <- ggplot(res, aes(x = ClusterId, y = Monetary, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_bw() + scale_fill_brewer(palette = "Reds")
# A função "ggplot" é utilizada para especificar o dataframe "res" como fonte de dados, e mapear as col
c <- ggplot(res, aes(x = ClusterId, y = Recency, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_bw() + scale_fill_brewer(palette = "Reds")
# Utilização da função "grid.arrange" que organiza os gráficos "a", "b" e "c" numa tabela com 3 colunas
grid.arrange(a,b,c, ncol = 3)
```




```
# Utilização da função "fviz_nbclust" para visualizar o grupo de dados do dataframe "RFM_scaled", e uti
# É adicionada uma linha vertical com interceção em 3, utilizando a função "geom_vline" e o tipo de lin
fviz_nbclust(RFM_scaled, FUN = hcut, method = "wss", linecolor = "Brown") + geom_vline(xintercept = 3, l
```




```
# Utilização da função "fviz_nbclust" para visualizar o grupo de dados do dataframe "RFM_scaled", e uti
fviz_nbclust(RFM_scaled, FUN = hcut, method = "silhouette", linecolor = "Brown") + theme_bw()
```



Hierarchical Clustering

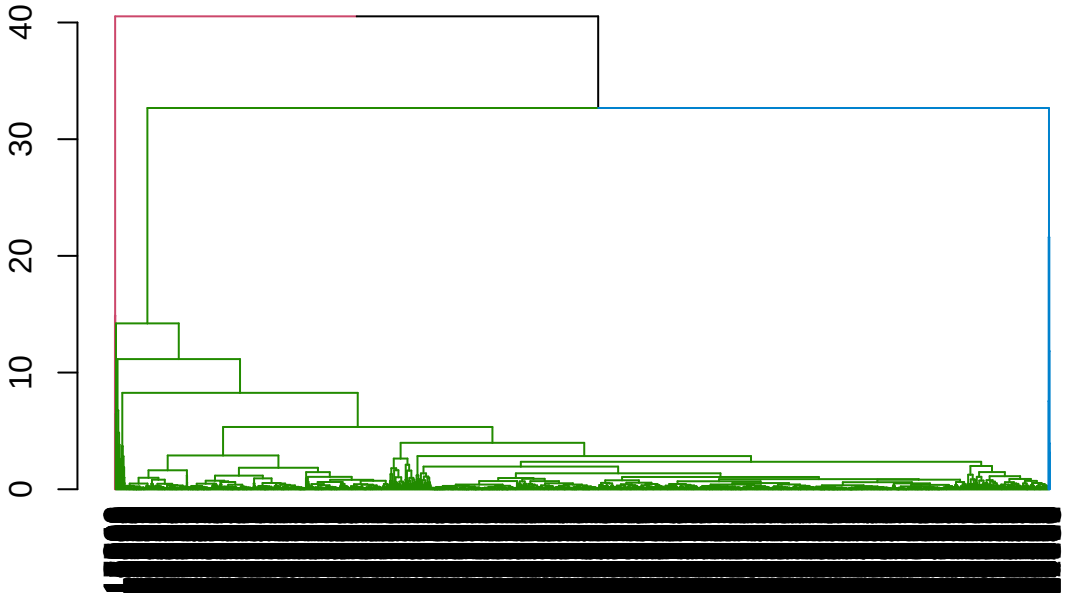
```
#HC aglomerante com hclust. Primeiro calculamos os valores dissimilares com dist e depois alimentamos e

euclidian_dist <- dist(RFM_scaled, method = "euclidean")


hc1 <- hclust(euclidian_dist, method = "single" )

hc2 <- hclust(euclidian_dist, method = "complete" )

hc3 <- hclust(euclidian_dist, method = "ward.D2" )

hc4 <- hclust(euclidian_dist, method = "average" )

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


ac <- function(x) {
  agnes(RFM_scaled, method = x)$ac
}
map_dbl(m, ac)
```

```
##   average    single  complete      ward
## 0.9976872 0.9955580 0.9984517 0.9992894
```

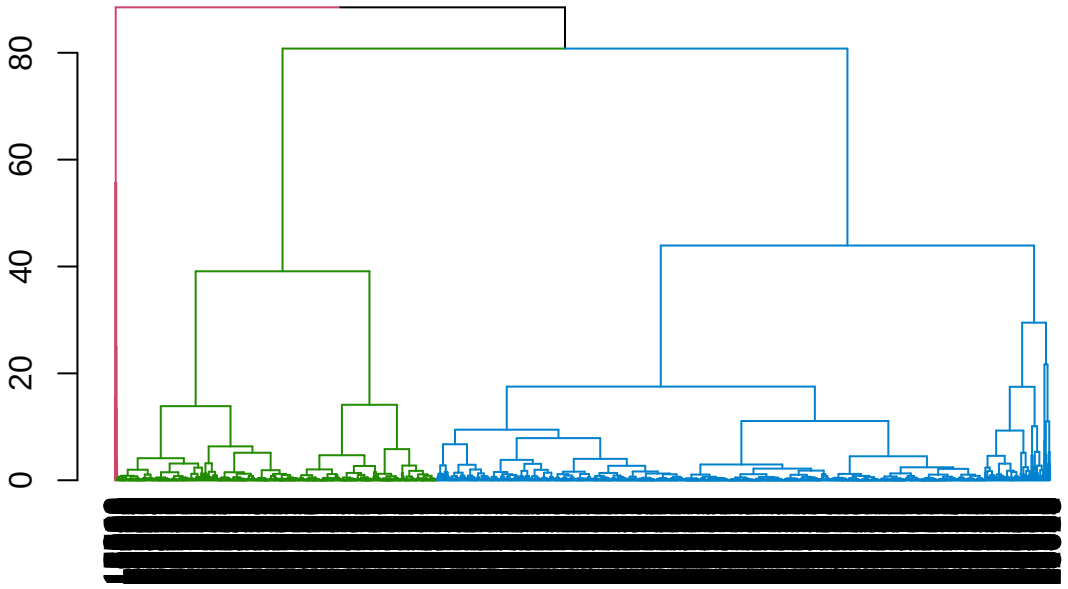
```
# Utilização da biblioteca "dendextend" para transformar o objeto "hc2" num dendrograma.
# De seguida, atribui-se as cores para as ramificações do dendrograma, com base em 3 grupos (k = 3), ut
# Por fim, realiza-se o dendrograma com as cores atribuídas.
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.16.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

## The following object is masked from 'package:data.table':
##
##     set

## The following object is masked from 'package:stats':
##
##     cutree
```

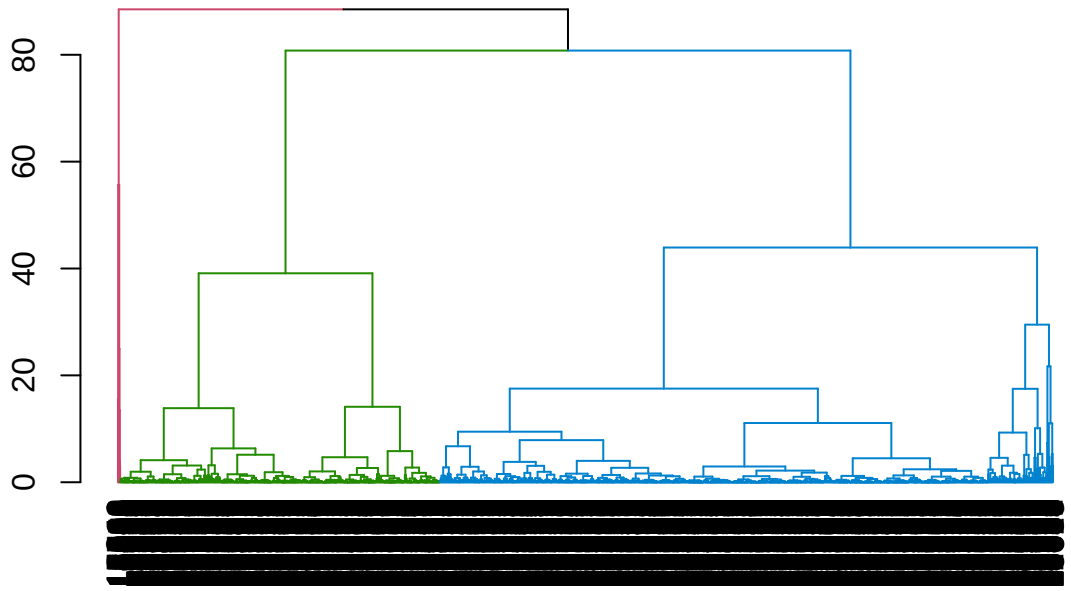
```
hc2 <- as.dendrogram(hc2)
cd = color_branches(hc2,k = 3)
plot(cd)
```



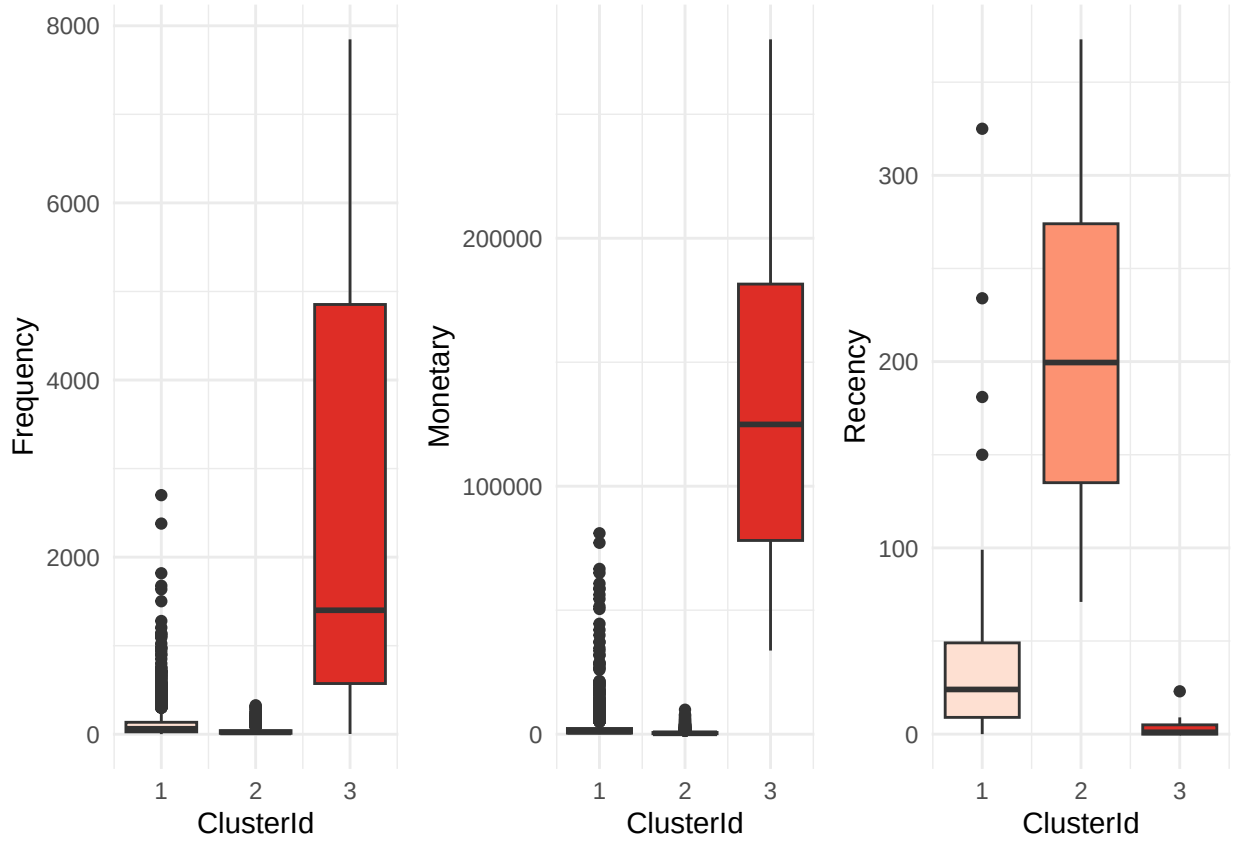

```
# Utilização da biblioteca "dendextend" para transformar o objeto "hc3" num dendrograma.
# De seguida, atribui-se as cores para as ramificações do dendrograma, com base em 3 grupos (k = 3), ut
# Por fim, realiza-se o dendrograma com as cores atribuídas.
hc3 <- as.dendrogram(hc3)
cd = color_branches(hc3,k = 3)
plot(cd)
```




```
# Utilização da biblioteca "dendextend" para transformar o objeto "hc3" num dendrograma.
# De seguida, atribui-se as cores para as ramificações do dendrograma, com base em 3 grupos (k = 3), ut
# Por fim, realiza-se o dendrograma com as cores atribuídas.
hc3 <- as.dendrogram(hc3)
cd = color_branches(hc3,k = 3)
plot(cd)
```

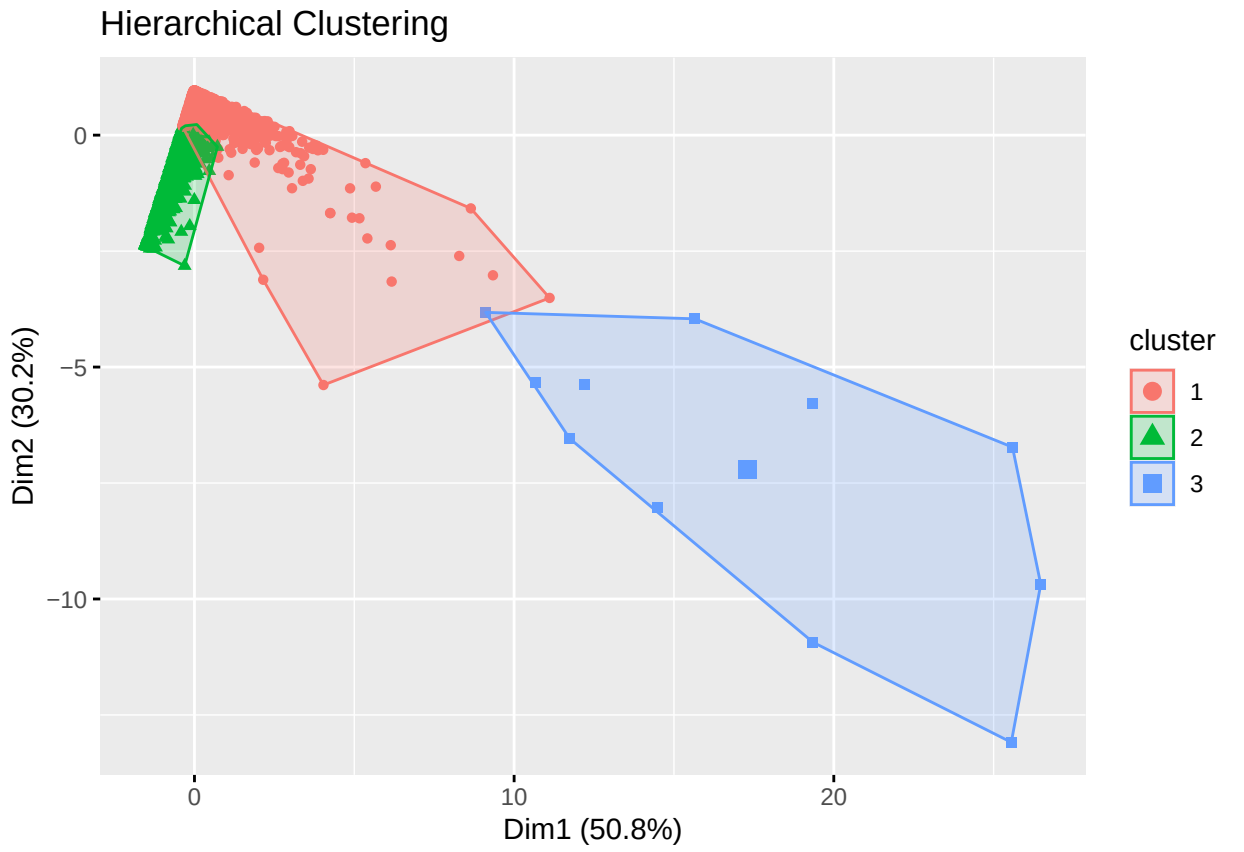



```
# À função "ward.clust" é atribuído o valor de "cutree" do agrupamento hierárquico (hc3), para 3 grupos
# De seguida, as colunas "RFM" e "ClusterId" são adicionadas ao "res1", sendo convertidas para um dataf
ward.clust = cutree(hc3,k = 3)
res1 <- cbind(RFM, ClusterId = ward.clust)
res1 <- as.data.frame(res1)
# Criação de um gráfico, utilizando a biblioteca "ggplot2", e utilizando os dados do dataframe "res1" e
# São agrupados os dados pelo "ClusterId" e as caixas do gráfico são preenchidas com o conjunto de core
# É utilizado um tema simples e não é exibida legenda.
a <- ggplot(res1, aes(x = ClusterId, y = Frequency, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_minimal() + scale_fill_brewer(palette = "Reds")
# Criação de um gráfico, utilizando a biblioteca "ggplot2", e utilizando os dados do dataframe "res1" e
# São agrupados os dados pelo "ClusterId" e as caixas do gráfico são preenchidas com o conjunto de core
# É utilizado um tema simples e não é exibida legenda.
b <- ggplot(res1, aes(x = ClusterId, y = Monetary, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_minimal() + scale_fill_brewer(palette = "Reds")
# Criação de um gráfico, utilizando a biblioteca "ggplot2", e utilizando os dados do dataframe "res1" e
# São agrupados os dados pelo "ClusterId" e as caixas do gráfico são preenchidas com o conjunto de core
# É utilizado um tema simples e não é exibida legenda.
c <- ggplot(res1, aes(x = ClusterId, y = Recency, group = ClusterId, fill = as.factor(ClusterId))) +
  geom_boxplot(show.legend = FALSE) + theme_minimal() + scale_fill_brewer(palette = "Reds")
# Utilização da função "grid.arrange" que organiza os gráficos "a", "b" e "c" numa tabela com 3 colunas
grid.arrange(a,b,c, ncol = 3)
```




```
# Utilização da função "fviz_cluster" para visualizar os resultados do grupo "K-means", com 3 clusters
# O parametro "geom" é configurado como "point" para exibir cada ponto de dados num gráfico de dispersã
# O título do gráfico é definido como "K-means Clustering", usando a função "ggtitle".
fviz_cluster(k3, data = RFM_scaled, geom = "point") + ggtitle("K-means Clustering")
```


```
# Visualização dos clusters gerados pelo algoritmo de agrupamento hierárquico "ward", utilizando os dad
# A função "fviz_cluster" é utilizada para exibir os clusters como pontos.
# O título do gráfico é "Hierarchical Clustering".
fviz_cluster(list(data = RFM_scaled, cluster = ward.clust), geom = "point") + ggtitle("Hierarchical Clus
```




```
#K-means
# Agrupamento dos dados "res" pelo id de clusters "res$ClusterId", e calculando a média dos valores par
# A função "aggregate" é usada para agrupar os dados e calcular a média, e a "by" é usada para especifi
aggregate(res,by = list(res$ClusterId),FUN = mean)
```

```
##   Group.1  Frequency    Monetary    Recency ClusterId
## 1       1   27.78796    637.3185 246.308333         1
## 2       2  103.08906   2028.8339  40.377196         2
## 3       3 2565.30769 126118.3100   3.692308         3
```

```
#Hierarchical
# Agrupamento dos dados "res1" pelo id de clusters "res1$ClusterId", e calculando a média dos valores p
# A função "aggregate" é usada para agrupar os dados e calcular a média, e a "by" é usada para especifi
aggregate(res1,by = list(res1$ClusterId),FUN = mean)
```

```
##   Group.1  Frequency    Monetary    Recency ClusterId
## 1       1  112.33978   2277.6918  30.464298         1
## 2       2   33.25472    617.2412 209.183962         2
## 3       3 2650.18182 138176.7545   4.181818         3
```

3.5 Association Rules

```
# O comando "str(dataset)" exibe as informações estruturais sobre o dataset.
# De seguida, é criada uma variável chamada "united_kingdom", que é uma cópia do dataset, e utiliza o p
# A função "as.datetime" é utilizada para formatar a coluna de data, de acordo com o parametro '%Y-%m-%
str(dataset)
```

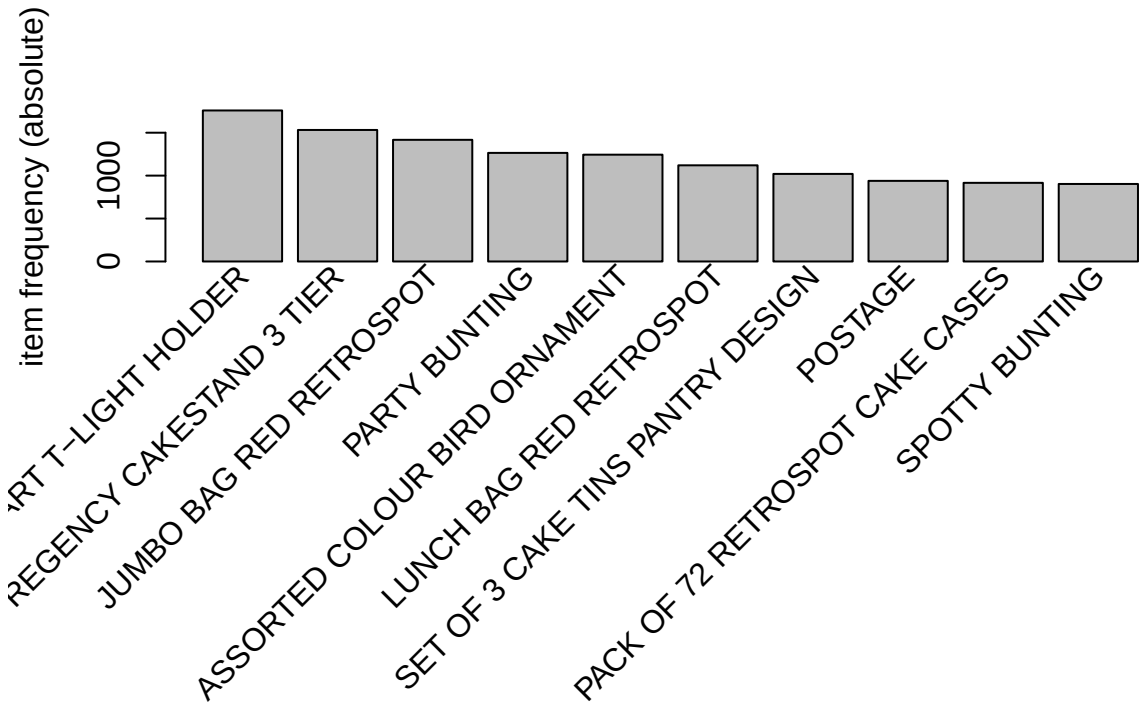
```
## tibble [397,884 x 16] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : chr [1:397884] "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr [1:397884] "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr [1:397884] "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPI
##  $ Quantity   : num [1:397884] 6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: POSIXct[1:397884], format: "2010-12-01 08:26:00" "2010-12-01 08:26:00" ...
##  $ UnitPrice  : num [1:397884] 2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : num [1:397884] 17850 17850 17850 17850 17850 ...
##  $ Country    : Factor w/ 37 levels "Australia","Austria",..: 35 35 35 35 35 35 35 35 35 35 ...
##  $ date       : Named chr [1:397884] "2010-12-01" "2010-12-01" "2010-12-01" "2010-12-01" ...
##   ..- attr(*, "names")= chr [1:397884] "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26
##  $ time       : Named chr [1:397884] "08:26:00" "08:26:00" "08:26:00" "08:26:00" ...
##   ..- attr(*, "names")= chr [1:397884] "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26
##  $ month      : Factor w/ 12 levels "01","02","03",..: 12 12 12 12 12 12 12 12 12 12 ...
##   ..- attr(*, "names")= chr [1:397884] "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26
##  $ year       : Factor w/ 2 levels "2010","2011": 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:397884] "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26
##  $ hourOfDay  : Named chr [1:397884] "08" "08" "08" "08" ...
##   ..- attr(*, "names")= chr [1:397884] "2010-12-01 08:26:00" "2010-12-01 08:26:00" "2010-12-01 08:26
##  $ TotalSales : num [1:397884] 15.3 20.3 22 20.3 20.3 ...
##  $ dayOfWeek  : Factor w/ 6 levels "1","2","3","4",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Diff       : 'difftime' num [1:397884] 373 373 373 373 ...
##   ..- attr(*, "units")= chr "days"
##  - attr(*, "na.action")= 'omit' Named int [1:135080] 623 1444 1445 1446 1447 1448 1449 1450 1451 145
##   ..- attr(*, "names")= chr [1:135080] "623" "1444" "1445" "1446" ...
```

```
united_kingdom <- dataset %>%
 mutate(InvoiceDate = as.datetime(InvoiceDate, '%Y-%m-%d %H:%M:%S'))

# Criação de uma nova variável chamada "invoiced_items" que contém informações sobre os itens faturados
# 1 - Utilização do pipe operator %>% para aplicar as funções "group_by", "select" e "distinct" para ag
# 2 - Utilização da função "setDT" para converter o conjunto de dados num objeto do tipo data.table.
# 3 - Utilização da função "dcast" para transformar os dados numa tabela de contagem, com linhas como a
# 4 - Utilização do "select" para remover a coluna "InvoiceNo".
# Por fim, temos uma tabela com linhas como o número de fatura e as colunas como as descrições de itens
invoiced_items <- dcast(setDT(united_kingdom %>% group_by(InvoiceNo) %>% select(InvoiceNo, Description)
 select(!InvoiceNo)
```

```
## Using 'Description' as value column. Use 'value.var' to override
```

```
# O conteúdo da variável "invoiced_items" é salvo num ficheiro CSV chamado "invoiced_items.csv".
# O ficheiro CSV não tem cabeçalho, aspas e os valores em falta aparecem como uma string vazia, graças
write.csv(invoiced_items, 'invoiced_items.csv', quote = FALSE, row.names = FALSE, col.names = FALSE, na
# Leitura do ficheiro 'invoiced_items.csv', sendo o mesmo guardado numa variável chamada 'transaction'.
# O formato especificado é 'basket' e o separador de colunas é ','.
suppressWarnings(transaction <- read.transactions('invoiced_items.csv', format = 'basket', sep = ','))
```

```
# A função "itemFrequencyPlot" gera um gráfico de frequência de itens a partir de uma transação de dados
# O parâmetro "transaction" é a transação de dados que será utilizada para gerar o gráfico.
# O parâmetro "topN" especifica o número de itens mais frequentes a serem exibidos no gráfico.
# O parâmetro "type" especifica o tipo de frequência a ser exibida no gráfico, que pode ser "absolute"
itemFrequencyPlot(transaction, topN = 10, type = 'absolute')
```




```
# A função "apriori" é utilizada para encontrar regras de associação nos dados da transação.
# O parâmetro "transaction" é a transação de dados que será utilizada para encontrar as regras de assoc
# O parâmetro "parameter" é uma lista de parâmetros adicionais que são utilizados para especificar os c
# O suporte é especificado com "supp" e é definido como 0.01, a confiança é especificada com "conf" e é
# As regras encontradas são guardadas na variável "rules".
rules <- apriori(transaction, parameter = list(supp = 0.01, conf = 0.85, maxlen = 3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##       3  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

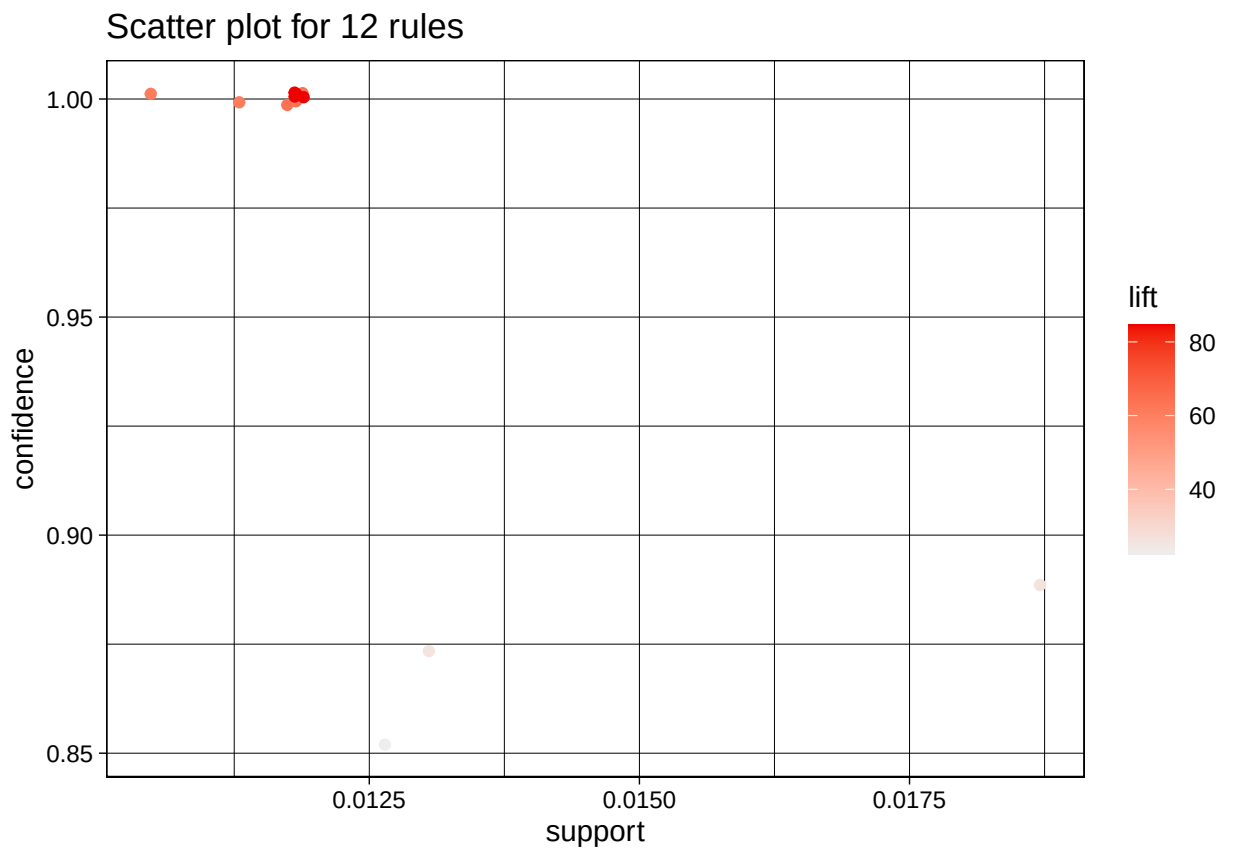
```
## Absolute minimum support count: 185
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[8431 item(s), 18533 transaction(s)] done [0.11s].
## sorting and recoding items ... [502 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3
```

```
## Warning in apriori(transaction, parameter = list(supp = 0.01, conf = 0.85, :
## Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
```

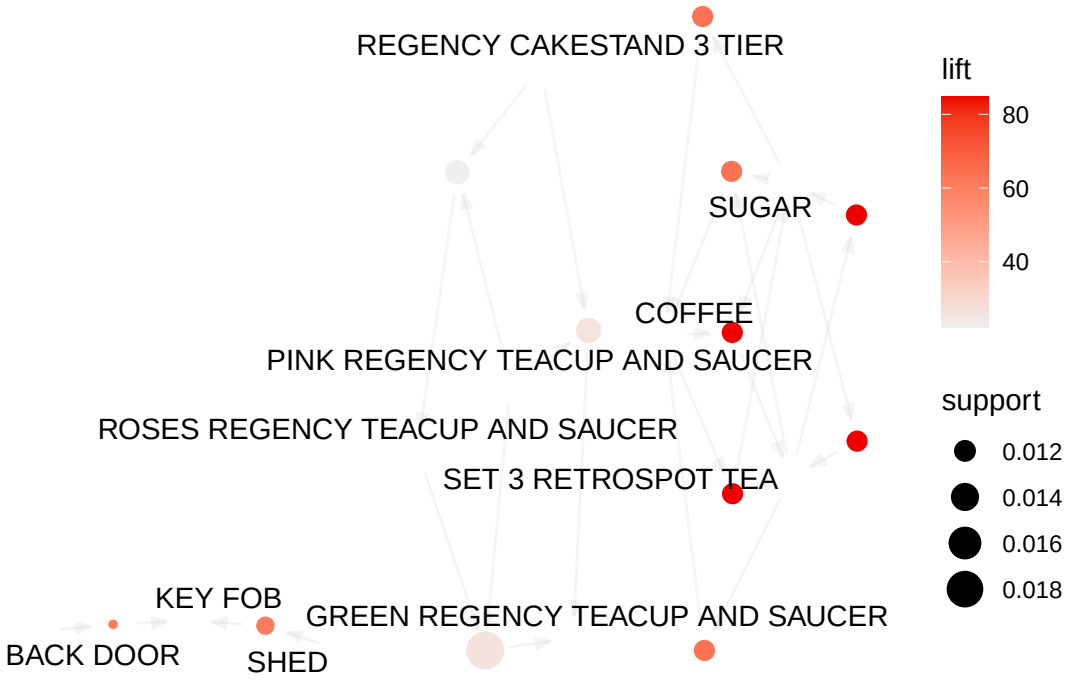
```
##  done [0.01s].
## writing ... [12 rule(s)] done [0.00s].
## creating S4 object  ... done [0.01s].
```

```
# A função "sort" é utilizada para classificar as regras de associação encontradas anteriormente.
# O parâmetro "rules" é a lista de regras de associação encontradas anteriormente.
# O parâmetro "by" é usado para especificar como as regras devem ser classificadas. Neste caso é "confi
# O parâmetro "decreasing" é usado para especificar se a classificação deve ser em ordem decrescente ou
rules <- sort(rules, by = 'confidence', decreasing = TRUE)
# Gráfico do parâmetro "rules".
plot(rules)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



```
# Realização do parâmetro "rules", em formato de gráfico.
plot(rules, method = 'graph')
```




```
inspect(sort(rules))
```

```
##      lhs                                  rhs                                support confidence   
## [1]  {PINK REGENCY TEACUP AND SAUCER,
##       ROSES REGENCY TEACUP AND SAUCER} => {GREEN REGENCY TEACUP AND SAUCER} 0.01872336  0.8897436 0.
## [2]  {PINK REGENCY TEACUP AND SAUCER,
##       REGENCY CAKESTAND 3 TIER}        => {GREEN REGENCY TEACUP AND SAUCER} 0.01300383  0.8731884 0.
## [3]  {PINK REGENCY TEACUP AND SAUCER,
##       REGENCY CAKESTAND 3 TIER}        => {ROSES REGENCY TEACUP AND SAUCER} 0.01268008  0.8514493 0.
## [4]  {SET 3 RETROSPOT TEA}             => {SUGAR}                           0.01181676  1.0000000 0.
## [5]  {SUGAR}                           => {SET 3 RETROSPOT TEA}             0.01181676  1.0000000 0.
## [6]  {SET 3 RETROSPOT TEA}             => {COFFEE}                          0.01181676  1.0000000 0.
## [7]  {SUGAR}                           => {COFFEE}                          0.01181676  1.0000000 0.
## [8]  {SET 3 RETROSPOT TEA,
##       SUGAR}                           => {COFFEE}                          0.01181676  1.0000000 0.
## [9]  {COFFEE,
##       SET 3 RETROSPOT TEA}             => {SUGAR}                           0.01181676  1.0000000 0.
## [10] {COFFEE,
##       SUGAR}                           => {SET 3 RETROSPOT TEA}             0.01181676  1.0000000 0.
## [11] {SHED}                            => {KEY FOB}                         0.01122322  1.0000000 0.
## [12] {BACK DOOR}                       => {KEY FOB}                         0.01041386  1.0000000 0.
```

4 Conclusões

Os clientes do Cluster 3 são os clientes com elevado volume de transacções, são compradores frequentes, e compradores que voltam recorrentemente em comparação com outros clientes, por isso são os mais importantes do ponto de vista comercial. Os clientes do Cluster 2 são os clientes com volume médio de transacções, em comparação com os melhores e os clientes fracos, sendo que são em termos de quantidade mais do que os melhores diria que a grosso modo tanto os clientes do Cluster 2 bem como os clientes do Cluster 3 serão os mais importantes para o negócio. Os clientes do Cluster 1 são os clientes com menor quantidade de transacções, são compradores pouco frequentes, e são compradores que voltam menos recorrentemente, portanto, menos importantes do ponto de vista comercial.

Hierárquico (3 Clusters) Os clientes do Cluster 3 são os clientes com elevado volume de transacções, são compradores frequentes, e compradores que voltam recorrentemente em comparação com outros clientes, por isso os mais importantes do ponto de vista comercial. Os clientes do Cluster 1 são os clientes com volume médio de transacções, em comparação com os melhores e os clientes fracos, sendo que são em termos de quantidade mais do que os melhores, diria que de grosso modo os clientes do Cluster 1 bem como os clientes do cluster 3 serão os mais importantes para o negócio. Os clientes do Cluster 2 são os clientes com menor quantidade de transacções, são compradores pouco frequentes, que voltam menos recorrentemente, portanto, menos importantes do ponto de vista comercial.

Sintetisando nas conclusões iniciais dos Clusters, achou-se que os Clientes estariam divididos entre Clientes Bons, Médios e Maus. No entanto após análise profunda o que é um mau cliente para o negócio? Qualquer pessoa que efetue uma compra no negócio é para nós importante e por consequência sendo o nosso principal objetivo vender o máximo de artigos possível, é lógico que para nós. Após uma análise mais profunda percebeu-se que a segmentação realizada anteriormente deixava de fazer qualquer sentido. Decidiu-se então padronizá-los por tipo de clientes e chegámos à conclusão de que o cluster 3 são as nossas “whales”, ou seja, os nossos melhores clientes aqueles que compram mais, com maior frequência e maior taxa de regresso à nossa loja. Atribuímos a estes um nível de importância 3 com a tentativa de os manter, não os esquecendo nunca, mas uma vez que já se encontram fidelizados, elaboramos campanhas mais esporádicas. Nos Clusters 2 no K-Means e 1 no Hierárquico de baixa taxa de regresso à loja, mas de alto valor monetário, notou-se que já foram clientes valiosos, mas desde então pararam. Chamámos-lhes de “Antigos”. Temos de promover arduamente o seu regresso utilizando possivelmente campanhas de loyalty e exclusivas para esta segmentação, obviamente não cometendo desigualdade relativamente ao setor anterior. Atribuímos a estes um nível de importância 1, tornando-se os mais importantes dos 3 Clusters. Quanto aos últimos Clusters 1 no K-Means e 2 no Hierárquico, chamaram-se de “Novos” pois são normalmente pessoas com alta taxa de regresso e baixa frequência. É um padrão de quem ainda se encontra a “avaliar” o terreno e ainda não está fidelizado, no entanto, achamos que um acompanhamento direcionado pode convertê-los em clientes repetidos, ou até quem sabe em “whales”, considerou-se que estes últimos teriam um nível de importância 2. Os níveis 1, 2 e 3 são uma escala de importância sendo 1 o mais importante e o 3 o menos importante.

Relativamente á associação criámos um novo ficheiro csv filtrado por País para podermos perceber pelo mesmos quais são as associações de produtos efetuadas.

Os 5 primeiros resultados da tabela com um IC de 85% ou superior são os seguintes:

Se comprar {PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY TEACUP AND SAUCER} então compra => {GREEN REGENCY TEACUP AND SAUCER} Se comprar {PINK REGENCY TEACUP AND SAUCER, REGENCY CAKESTAND 3 TIER} então compra => {GREEN REGENCY TEACUP AND SAUCER} Se comprar {PINK REGENCY TEACUP AND SAUCER, REGENCY CAKESTAND 3 TIER} então compra => {ROSES REGENCY TEACUP AND SAUCER} Se comprar {SET 3 RETROSPOT TEA} então compra => {SUGAR} Se comprar {SUGAR} então compra => {SET 3 RETROSPOT TEA}

Podemos utilizar esta associação para reforçar a compra junto dos clientes, relembrando que estes produtos comprados em conjunto terão vantagens monetárias para os mesmos, reforçando assim a fidelização e a compra.