INF0613 – Aprendizado de Máquina Não Supervisionado

Trabalho 3 - Técnicas de Agrupamento

Nicole Nogueira Silva Rodolfo Dalla Costa

O objetivo deste trabalho é exercitar o uso de algoritmos de agrupamento. Neste trabalho, vamos analisar diferentes atributos de carros com o objetivo de verificar se seus atributos são suficientes para indicar um valor de risco de seguro. O conjunto de dados já apresenta o risco calculado no campo `symboling` indicado na Tabela 1. Quanto mais próximo de 3, maior o risco. O conjunto de dados que deve ser usado está disponível na página do Moodle com o nome `imports-85.data`.

Atividade 0 – Configurando o ambiente

Antes de começar a implementação do seu trabalho configure o *workspace* e importe todos os pacotes e execute o preprocessamento da base:

```
# Adicione os pacotes usados neste trabalho:
library(dplyr)
library(caret)
library(NbClust)
library(factoextra)
library(dbscan)
library(knitr)
library(cluster)

# Configure ambiente de trabalho na mesma pasta
# onde colocou a base de dados:
#setwd("/Users/rodolfodc/Documents/mineracao-dados-complexos/homeworks/inf-0611-0612/inf0613/t3")
setwd("C:/Users/nicol/Documents/Mineração de dados/inf-0611-0612/inf0613/t2")
```

Atividade 1 – Análise e Preparação dos Dados

O conjunto de dados é composto por 205 amostras com 26 atributos cada descritos na Tabela 1. Os atributos são dos tipos `factor`, `integer` ou `numeric`. O objetivo desta etapa é a análise e preparação desses dados de forma a ser possível agrupá-los nas próximas atividades.

Implementações: Nos itens a seguir você implementará a leitura da base e aplicará tratamentos básicos.

a) *Tratamento de dados Incompletos:* Amostras incompletas deverão ser tratadas, e você deve escolher a forma que achar mais adequada. Considere como uma amostra incompleta uma linha na qual faltam dados em alguma das colunas selecionadas anteriormente. Note que, dados faltantes nas amostras podem causar uma conversão do tipo do atributo de todas as amostras e isso pode impactar no item b).

```
### Leitura da base
imports_85 <- read.table("imports-85.data",sep =",")

### Tratamento de dados faltantes
```

```r
#inspeção inicial
head(imports_85,2)
```

```
##   V1 V2          V3  V4  V5  V6          V7  V8    V9  V10 V11  V12  V13  V14
## 1  3  ? alfa-romero gas std two convertible rwd front 88.6 169 64.1 48.8 2548
## 2  3  ? alfa-romero gas std two convertible rwd front 88.6 169 64.1 48.8 2548
##    V15  V16 V17  V18  V19  V20 V21 V22  V23 V24 V25   V26
## 1 dohc four 130 mpfi 3.47 2.68   9 111 5000  21  27 13495
## 2 dohc four 130 mpfi 3.47 2.68   9 111 5000  21  27 16500
```

```r
#identificação de ? como dado faltante

### Removendo dados faltantes
imports_85[imports_85 == "?"] <- NA #substituindo por NA

#verificando se não está atrelado a uma categoria de seguro específica
faltantes <- imports_85[rowSums(is.na(imports_85)) != 0, ]

table(faltantes$V1) %>%
  kable(caption = "Risco do Seguro dos Dados faltantes", booktabs = T, linesep = "", digits = 2)
```

Table 1: Risco do Seguro dos Dados faltantes

Var1	Freq
-1	2
0	19
1	8
2	3
3	14

```r
#removendo os NAs
imports_85 <- na.omit(imports_85)

#transformando os atributos em suas devidas classes
imports_85 <- imports_85 %>%
  mutate(V2 = as.numeric(imports_85$V2),
         V19 = as.numeric(imports_85$V19),
         V20 = as.numeric(imports_85$V20),
         V22 = as.numeric(imports_85$V22),
         V23 = as.numeric(imports_85$V23),
         V26 = as.numeric(imports_85$V26))

#verificando a nova estrutura
str(imports_85)
```

```
## 'data.frame':    159 obs. of  26 variables:
##  $ V1 : int  2 2 1 1 2 0 0 0 2 1 ...
##  $ V2 : num  164 164 158 158 192 192 188 188 121 98 ...
##  $ V3 : chr  "audi" "audi" "audi" "audi" ...
##  $ V4 : chr  "gas" "gas" "gas" "gas" ...
##  $ V5 : chr  "std" "std" "std" "turbo" ...
##  $ V6 : chr  "four" "four" "four" "four" ...
##  $ V7 : chr  "sedan" "sedan" "sedan" "sedan" ...
```

```
##  $ V8 : chr  "fwd" "4wd" "fwd" "fwd" ...
##  $ V9 : chr  "front" "front" "front" "front" ...
##  $ V10: num  99.8 99.4 105.8 105.8 101.2 ...
##  $ V11: num  177 177 193 193 177 ...
##  $ V12: num  66.2 66.4 71.4 71.4 64.8 64.8 64.8 64.8 60.3 63.6 ...
##  $ V13: num  54.3 54.3 55.7 55.9 54.3 54.3 54.3 54.3 53.2 52 ...
##  $ V14: int  2337 2824 2844 3086 2395 2395 2710 2765 1488 1874 ...
##  $ V15: chr  "ohc" "ohc" "ohc" "ohc" ...
##  $ V16: chr  "four" "five" "five" "five" ...
##  $ V17: int  109 136 136 131 108 108 164 164 61 90 ...
##  $ V18: chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
##  $ V19: num  3.19 3.19 3.19 3.13 3.5 3.5 3.31 3.31 2.91 3.03 ...
##  $ V20: num  3.4 3.4 3.4 3.4 2.8 2.8 3.19 3.19 3.03 3.11 ...
##  $ V21: num  10 8 8.5 8.3 8.8 8.8 9 9 9.5 9.6 ...
##  $ V22: num  102 115 110 140 101 101 121 121 48 70 ...
##  $ V23: num  5500 5500 5500 5500 5800 5800 4250 4250 5100 5400 ...
##  $ V24: int  24 18 19 17 23 23 21 21 47 38 ...
##  $ V25: int  30 22 25 20 29 29 28 28 53 43 ...
##  $ V26: num  13950 17450 17710 23875 16430 ...
##  - attr(*, "na.action")= 'omit' Named int [1:46] 1 2 3 6 8 10 15 16 17 18 ...
##   ..- attr(*, "names")= chr [1:46] "1" "2" "3" "6" ...
```

b) *Seleção de Atributos:* Atributos não-numéricos não podem ser usados com as técnicas agrupamento vistas em aula. Portanto, você deve selecionar um conjunto de atributos numéricos que serão usados para o agrupamento. Além disso você deve analisar se os atributos não-numéricos são descritivos para a realização dos agrupamentos. Caso um dos atributos não numéricos seja necessário, use a técnica do *one hot encoding* para transformá-lo em numérico. **Não** aplique essa técnica nos atributos `symboling` e `make` para os agrupamentos subsequentes, eles não devem fazer parte do agrupamento.

```r
# Seleção de atributos
# Transformando os dados categóricos em numéricos através de one hot encoding
#v4
imports_85$v4gas <- as.numeric(imports_85$V4 == "gas")
imports_85$v4diesel <- as.numeric(imports_85$V4 == "diesel")
#v5
imports_85$v5std <- as.numeric(imports_85$V5 == "std")
imports_85$v5turbo <- as.numeric(imports_85$V5 == "turbo")
#v6
imports_85$v6four <- as.numeric(imports_85$V6 == "four")
imports_85$v6two <- as.numeric(imports_85$V6 == "two")
#v7
imports_85$v7convertible <- as.numeric(imports_85$V7 == "convertible")
imports_85$v7hardtop <- as.numeric(imports_85$V7 == "hardtop")
imports_85$v7hatchback <- as.numeric(imports_85$V7 == "hatchback")
imports_85$v7sedan <- as.numeric(imports_85$V7 == "sedan")
imports_85$v7wagon <- as.numeric(imports_85$V7 == "wagon")
#v8
imports_85$v84wd <- as.numeric(imports_85$V8 == "4wd")
imports_85$v8fwd <- as.numeric(imports_85$V8 == "fwd")
imports_85$v8rwd <- as.numeric(imports_85$V8 == "rwd")
#v9
imports_85$v9front <- as.numeric(imports_85$V9 == "front")
#v15
imports_85$v15dohc <- as.numeric(imports_85$V15 == "dohc")
imports_85$v15l <- as.numeric(imports_85$V15 == "l")
```

```r
imports_85$v15ohc <- as.numeric(imports_85$V15 == "ohc")
imports_85$v15ohcf <- as.numeric(imports_85$V15 == "ohcf")
imports_85$v15ohcv <- as.numeric(imports_85$V15 == "ohcv")
#v16
imports_85$V16eight <- as.numeric(imports_85$V16 == "eight")
imports_85$V16five <- as.numeric(imports_85$V16 == "five")
imports_85$V16four <- as.numeric(imports_85$V16 == "four")
imports_85$V16six <- as.numeric(imports_85$V16 == "six")
imports_85$V16three <- as.numeric(imports_85$V16 == "three")
#V18
imports_85$V181bbl <- as.numeric(imports_85$V18 == "1bbl")
imports_85$V182bbl <- as.numeric(imports_85$V18 == "2bbl")
imports_85$V18idi <- as.numeric(imports_85$V18 == "idi")
imports_85$V18mfi <- as.numeric(imports_85$V18 == "mfi")
imports_85$V18mpfi <- as.numeric(imports_85$V18 == "mpfi")
imports_85$V18spdi <- as.numeric(imports_85$V18 == "spdi")

#Matriz de correlação das variáveis
numericos <- imports_85[,c(2,c(10:14),17,19:40,42:57)] #removendo categoricos e o V9 - Front
matriz_corr <- cor(numericos)
#print(matriz_corr)

#Encontrando atributos fortemente correlacionados
altamente_corr <- findCorrelation(matriz_corr, cutoff =0.5)
print(altamente_corr)
```

```
##  [1]  6 15  4  3  7  2 14 11 13 29 28 41 37 16 17 42 18 32 24 20 21 34  9 35
```

```r
#Atributos relevantes
atributos <- names(numericos[,-altamente_corr])
imports_85sel <- imports_85 %>% select(atributos)
```

Análises

Após as implementações escreva uma análise da base de dados. Em especial, descreva o conjunto de dados inicial, relate como foi realizado o tratamento, liste quais os atributos escolhidos para manter na base e descreva a base de dados após os tratamentos listados. Explique todos os passos executados, mas sem copiar códigos na análise. Além disso justifique suas escolhas de tratamento nos dados faltantes e seleção de atributos.

Resposta:

O banco de dados é formado por informações relacionadas a características de carros. Entre os 26 atributos disponibilizados, 25 correspondem a fatores descritores dos carros inclusive a coluna *symboling*, representam o risco atrelado ao seguro do carro descrito. Ao realizar uma análise prévia, nota-se a existência de dados faltantes em 46 observações.

Antes de remover as observações, é necessário entender se os dados faltantes estavam atrelados à uma categoria específica de risco de seguro. Como observamos que isto não ocorria, decidimos remover esses registros com dados faltantes, resultando em uma base com 159 registros.

Após remover os dados faltantes, realizamos um tratamento das variáveis dado a presença de atributos categóricos. Assim, aplicamos a técnica de One-Hot-Encoding para conversão dos dados em variáveis numéricas, de forma a deixar o conjunto de dados mais preparado para a seleção. Com os dados transformados, a correlação entre todas as variáveis foi mensurada e a partir dos resultados da matriz, foi possível concluir que as variáveis, *V2, V13, V19, V21, V23, v5turbo, v7convertible, v7hardtop, v7sedan, v7wagon, v84wd, v15dohc,*

v15l, v15ohcf, V16five, V16six, V16three, V181bbl, V18mfi, V18mpfi eram suficientes para contribuir para a predição do risco do seguro.

Atividade 2 – Agrupamento com o K-*means*

Nesta atividade, você deverá agrupar os dados com o algoritmo K-*means* e utilizará duas métricas básicas para a escolha do melhor K: a soma de distâncias intra-cluster e o coeficiente de silhueta.

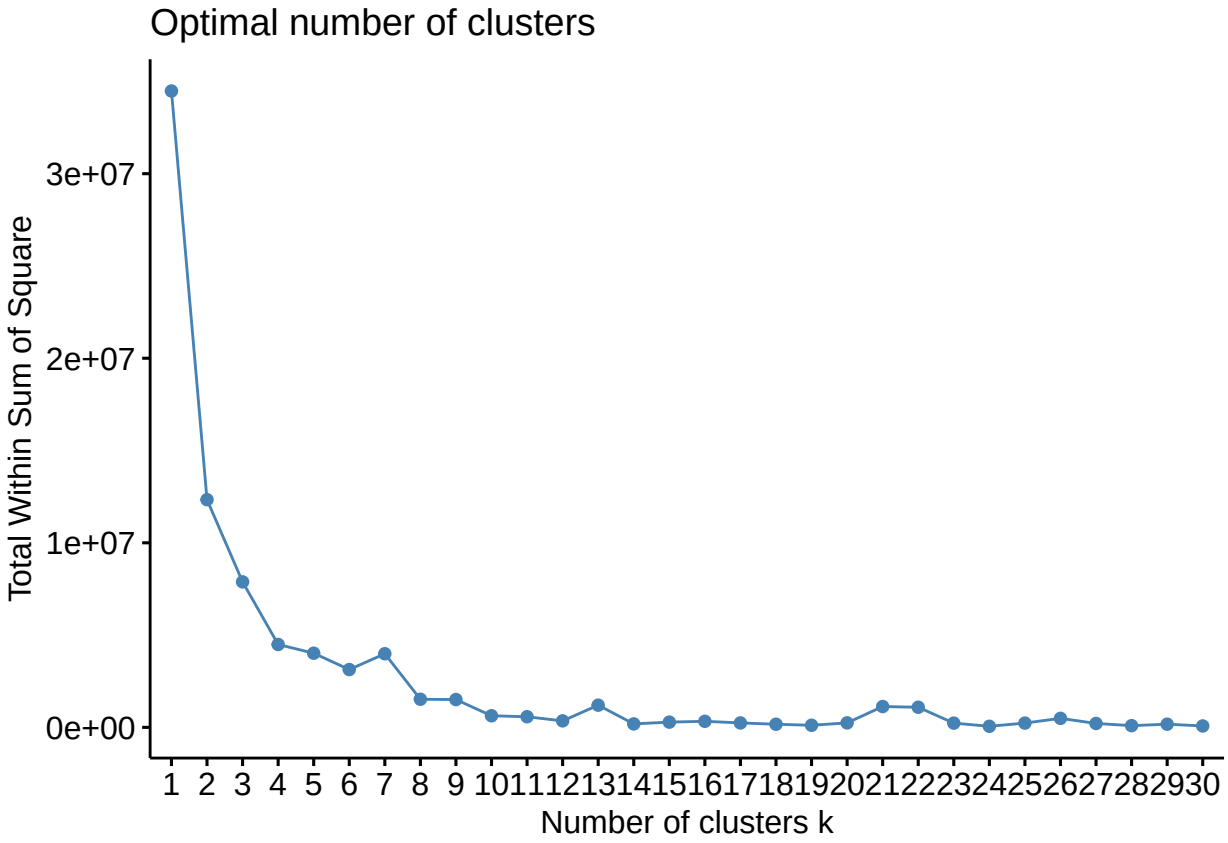
Implementações: Nos itens a seguir você implementará a geração de gráficos para a análise das distâncias intra-cluster e do coeficiente de silhueta. Em seguida, você implementará o agrupamento dos dados processados na atividade anterior com o algoritmo K-*means* utilizando o valor de K escolhido.

a) *Gráfico Elbow Curve*: Construa um gráfico com a soma das distâncias intra-cluster para K variando de 2 a 30.

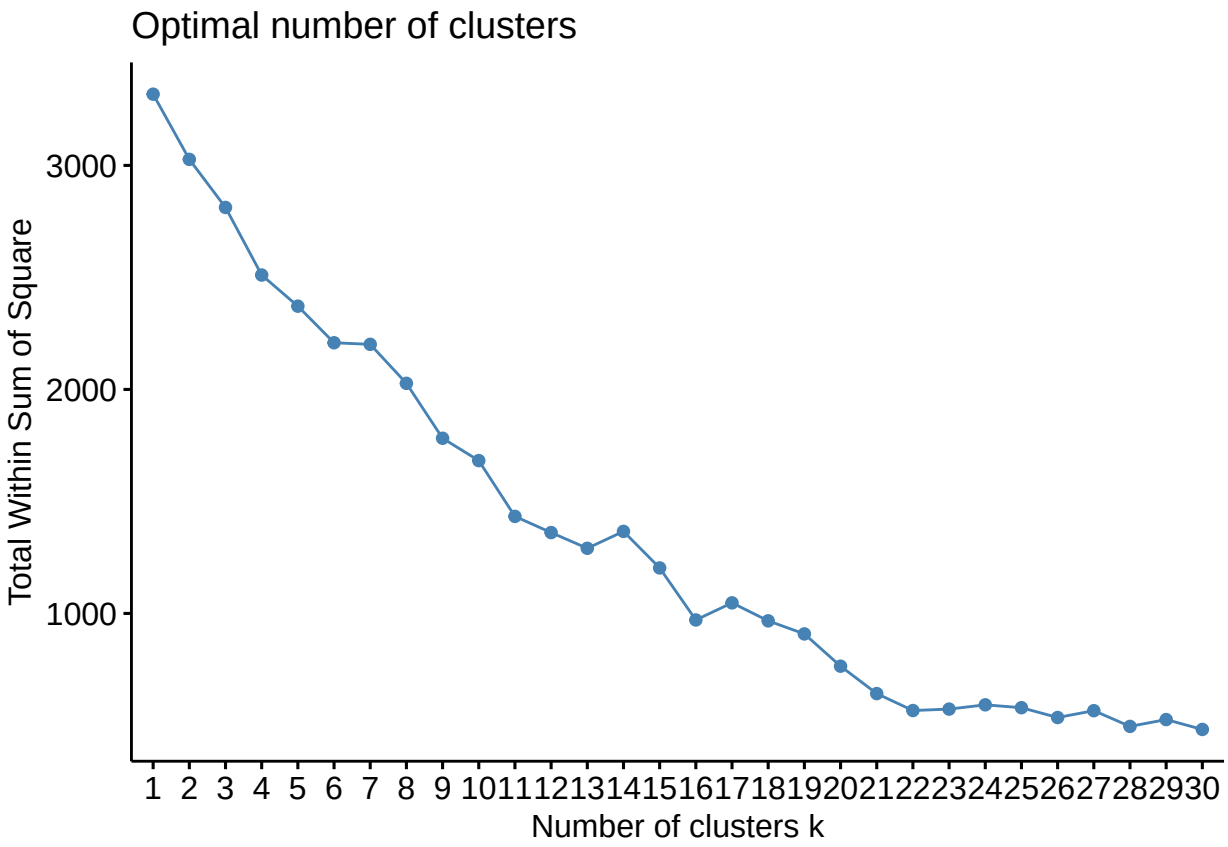
```
# Construindo um gráfico com as distâncias intra-cluster
set.seed(13)

imports_scaled <- scale(imports_85sel)

fviz_nbclust(imports_85sel, kmeans,k.max = 30, method ="wss")
```

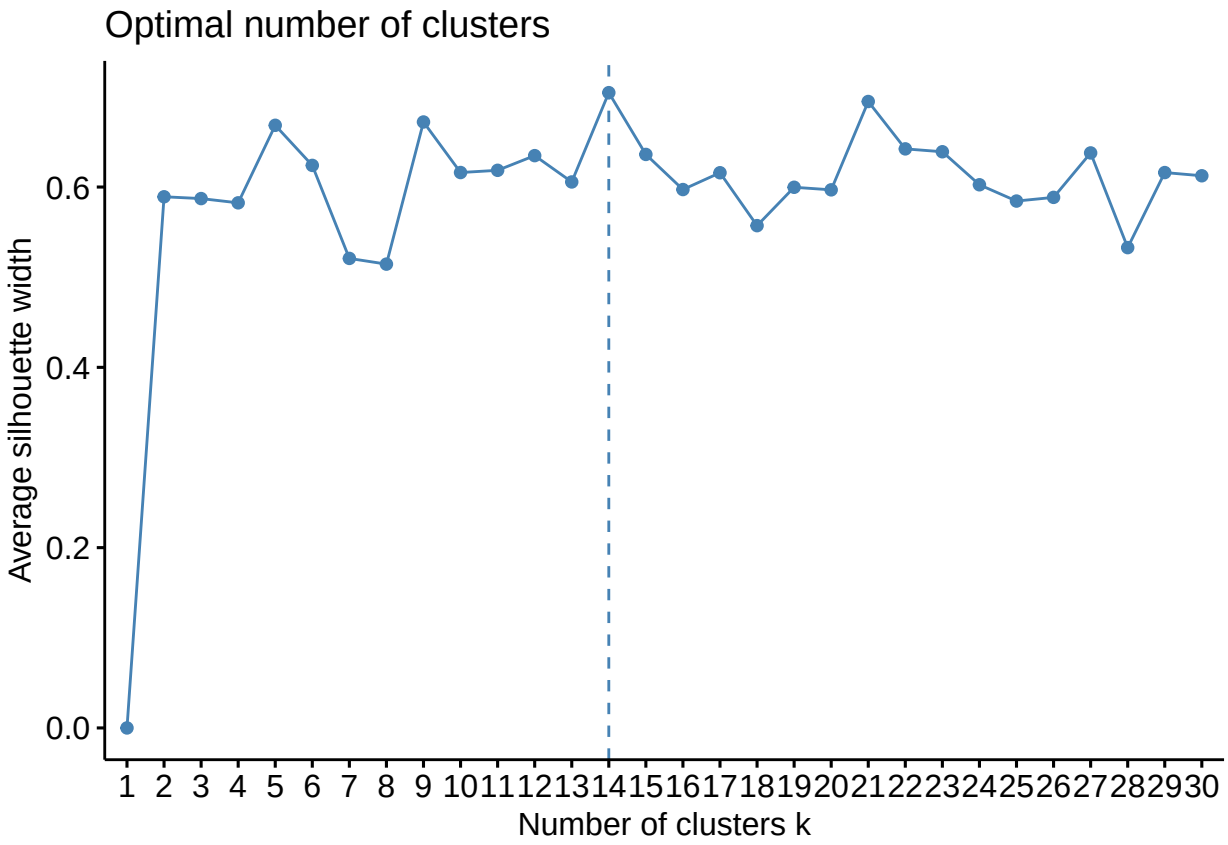


```
fviz_nbclust(imports_scaled, kmeans,k.max = 30, method ="wss")
```

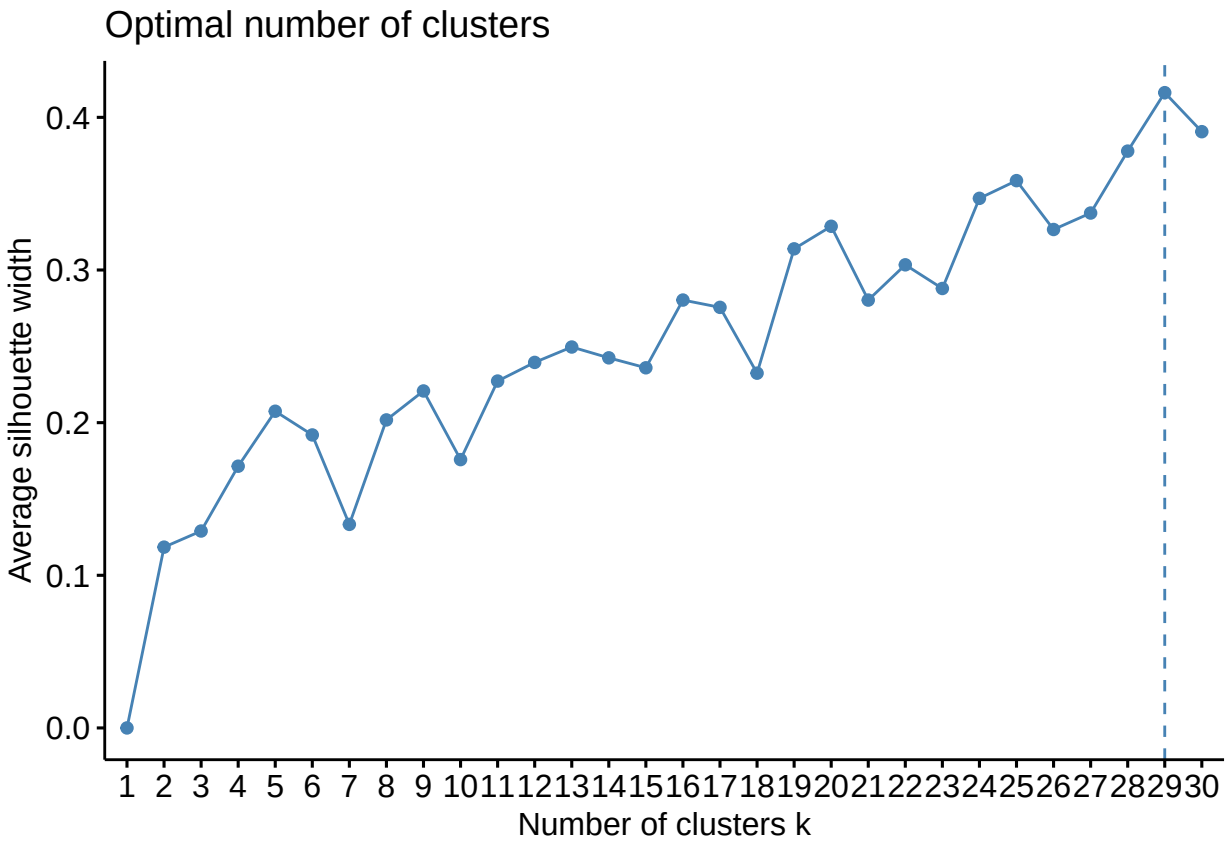


b) *Gráfico da Silhueta:* Construa um gráfico com o valor da silhueta para K variando de 2 a 30.

```
# Construindo um gráfico com os valores da silhueta
fviz_nbclust(imports_85sel, kmeans, method ="silhouette", k.max = 30)
```




```
fviz_nbclust(imports_scaled, kmeans, method ="silhouette", k.max = 30)
```

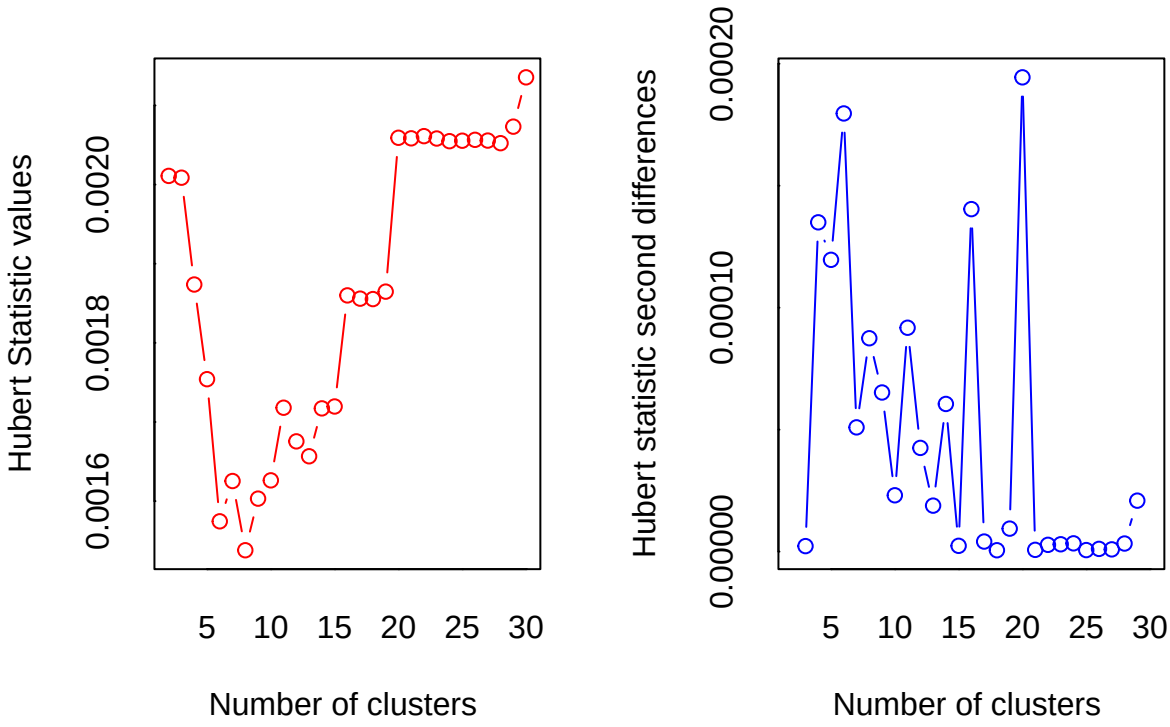


c) *Escolha do K*: Avalie os gráficos gerados nos itens anteriores e escolha o melhor valor de K com base nas informações desses gráficos e na sua análise. Se desejar, use também a função `NbClust` para ajudar nas análises. Com o valor de K definido, utilize o rótulo obtido para cada amostra, indicando o grupo ao qual ela pertence, para gerar um gráfico de dispersão (atribuindo cores diferentes para cada grupo).

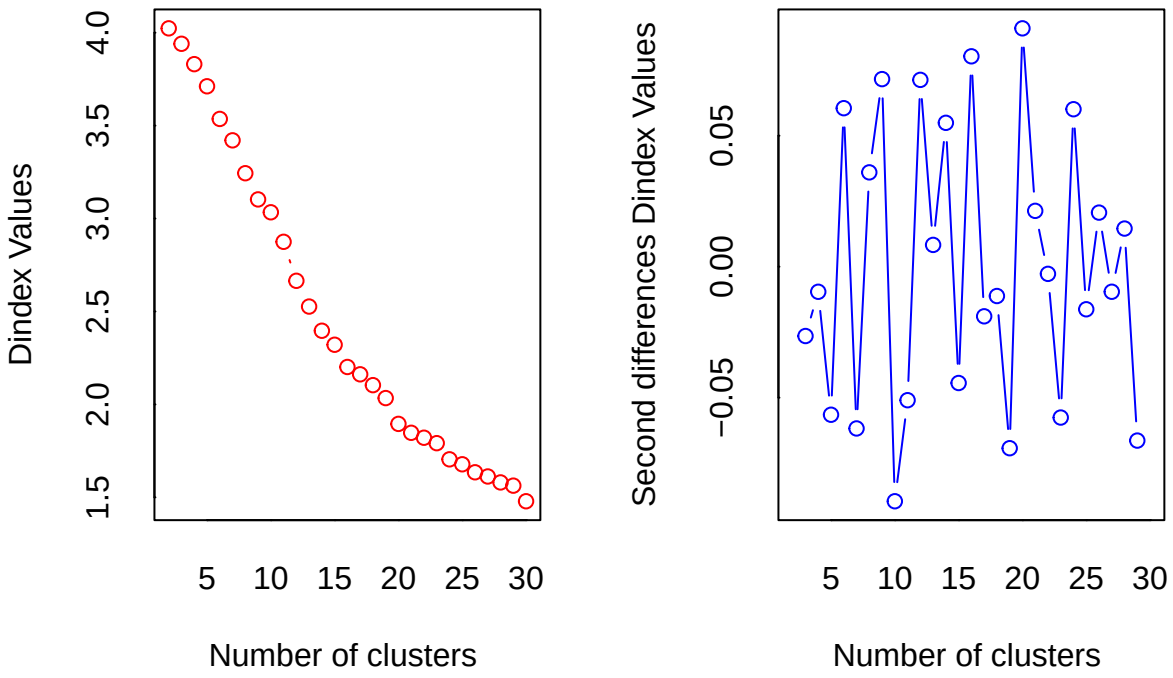
```
# Aplicando o k-means com o k escolhido

set.seed(137)

nb <- NbClust(imports_scaled, distance ="euclidean", min.nc =2, max.nc =30, method ="complete", index =
```



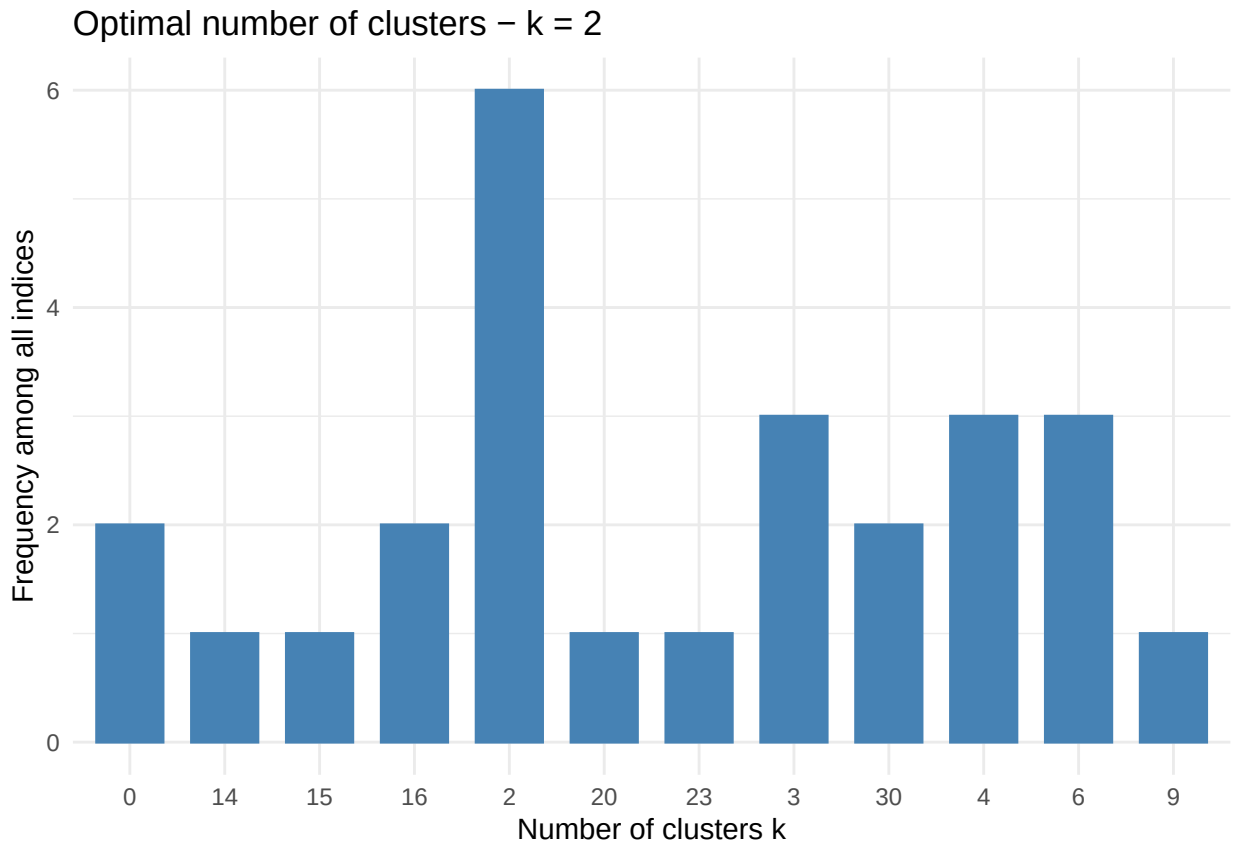

```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```



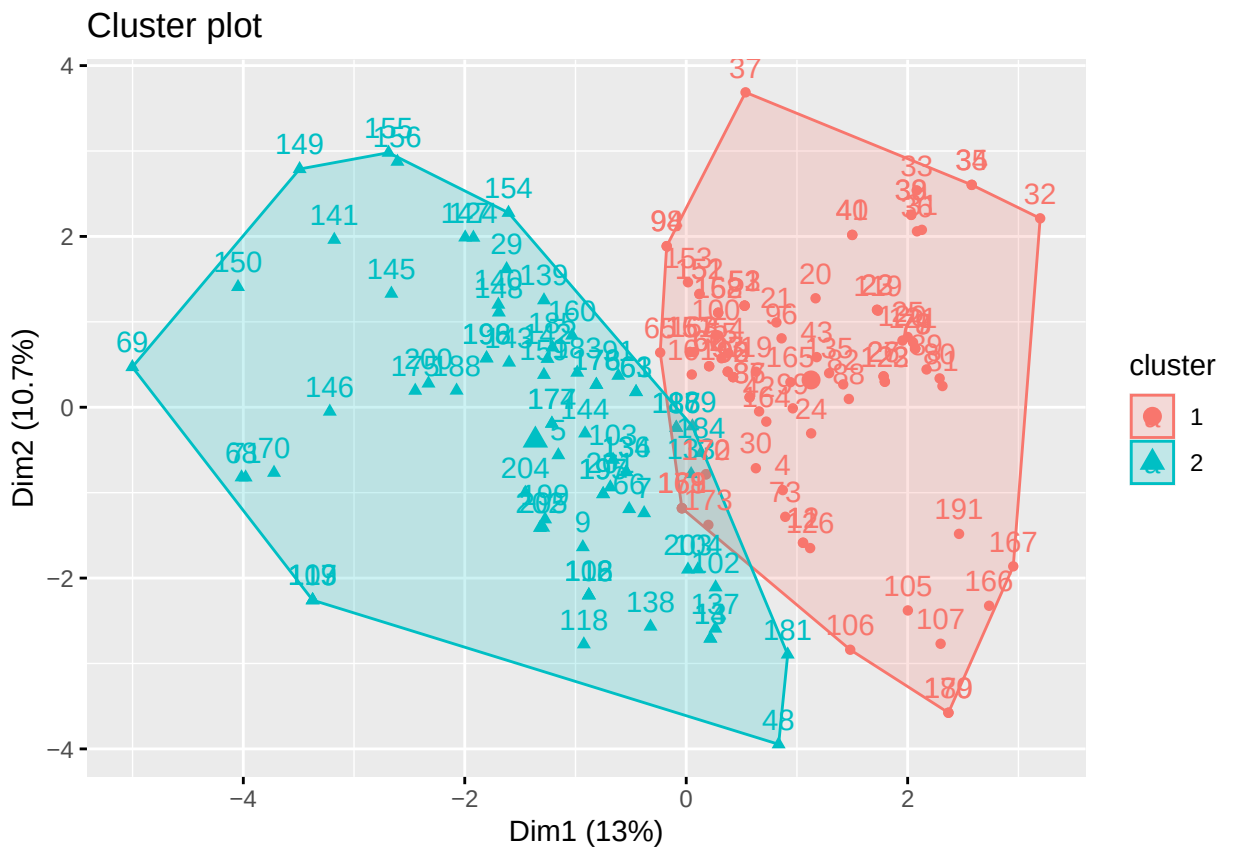

```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 6 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 3 proposed 4 as the best number of clusters
## * 3 proposed 6 as the best number of clusters
## * 1 proposed 9 as the best number of clusters
## * 1 proposed 14 as the best number of clusters
## * 1 proposed 15 as the best number of clusters
## * 2 proposed 16 as the best number of clusters
## * 1 proposed 20 as the best number of clusters
## * 1 proposed 23 as the best number of clusters
## * 2 proposed 30 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
#
fviz_nbclust(nb) + theme_minimal()
```

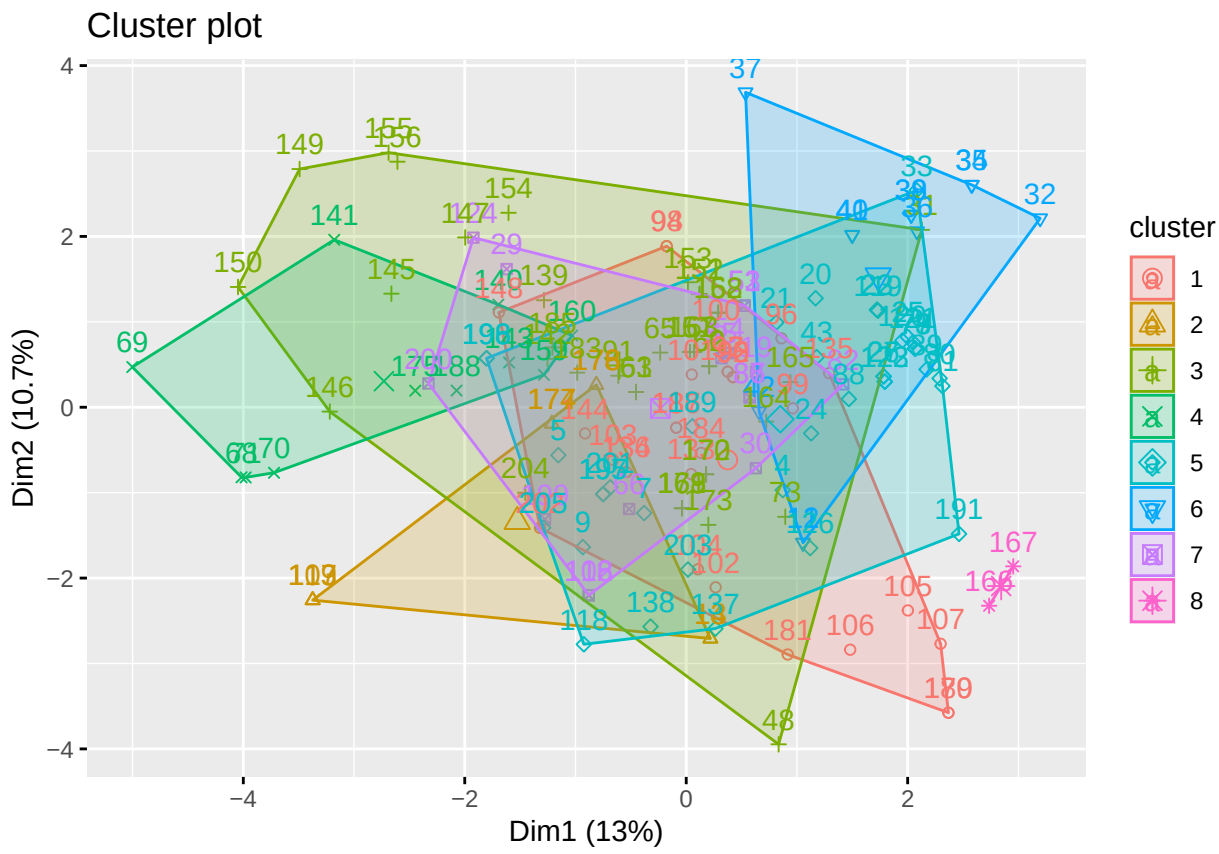
```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 6 proposed  2 as the best number of clusters
## * 3 proposed  3 as the best number of clusters
## * 3 proposed  4 as the best number of clusters
## * 3 proposed  6 as the best number of clusters
## * 1 proposed  9 as the best number of clusters
## * 1 proposed  14 as the best number of clusters
## * 1 proposed  15 as the best number of clusters
## * 2 proposed  16 as the best number of clusters
## * 1 proposed  20 as the best number of clusters
## * 1 proposed  23 as the best number of clusters
## * 2 proposed  30 as the best number of clusters
##
## Conclusion
## ==========================
## * According to the majority rule, the best number of clusters is  2 .
```



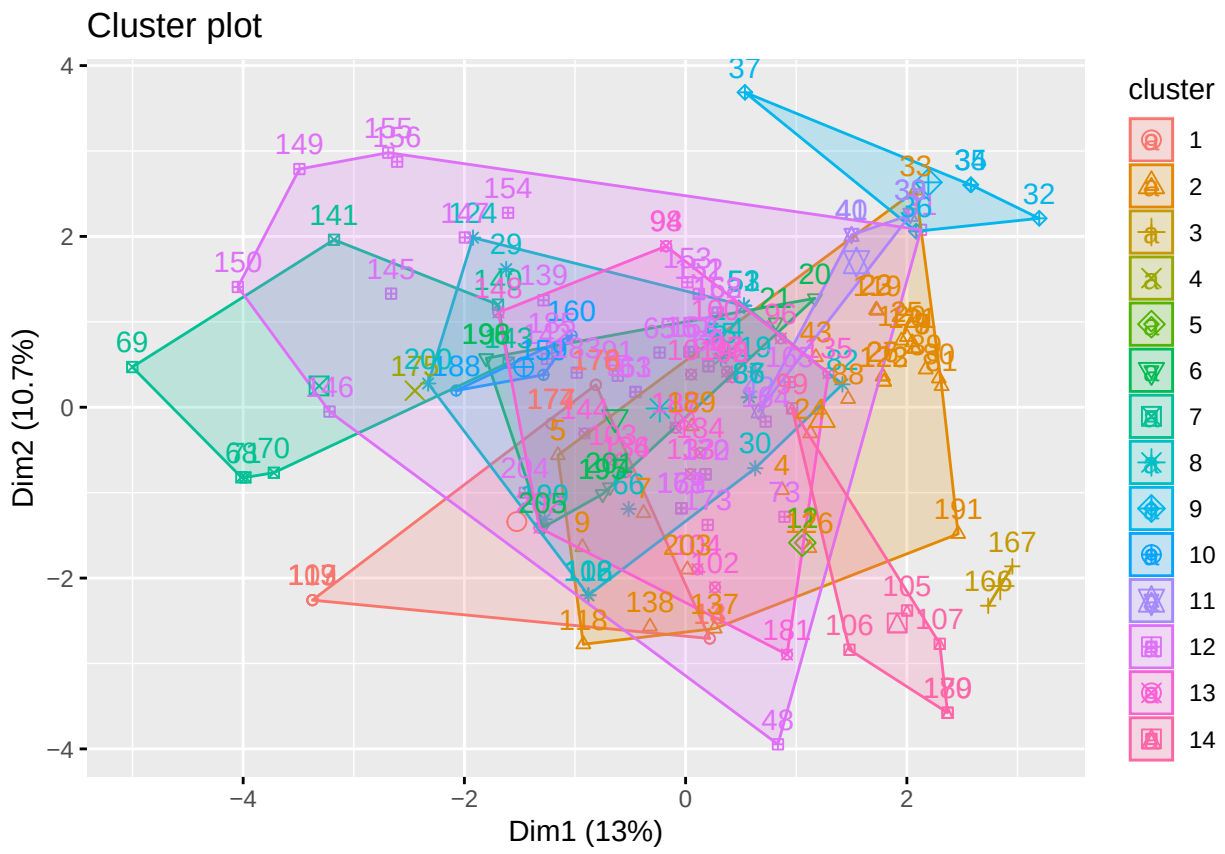

```
# Normalizada
final <- kmeans(imports_scaled, centers = 2,  nstart=37)
# Construindo um gráfico de dispersão
fviz_cluster(final, data =imports_scaled)
```



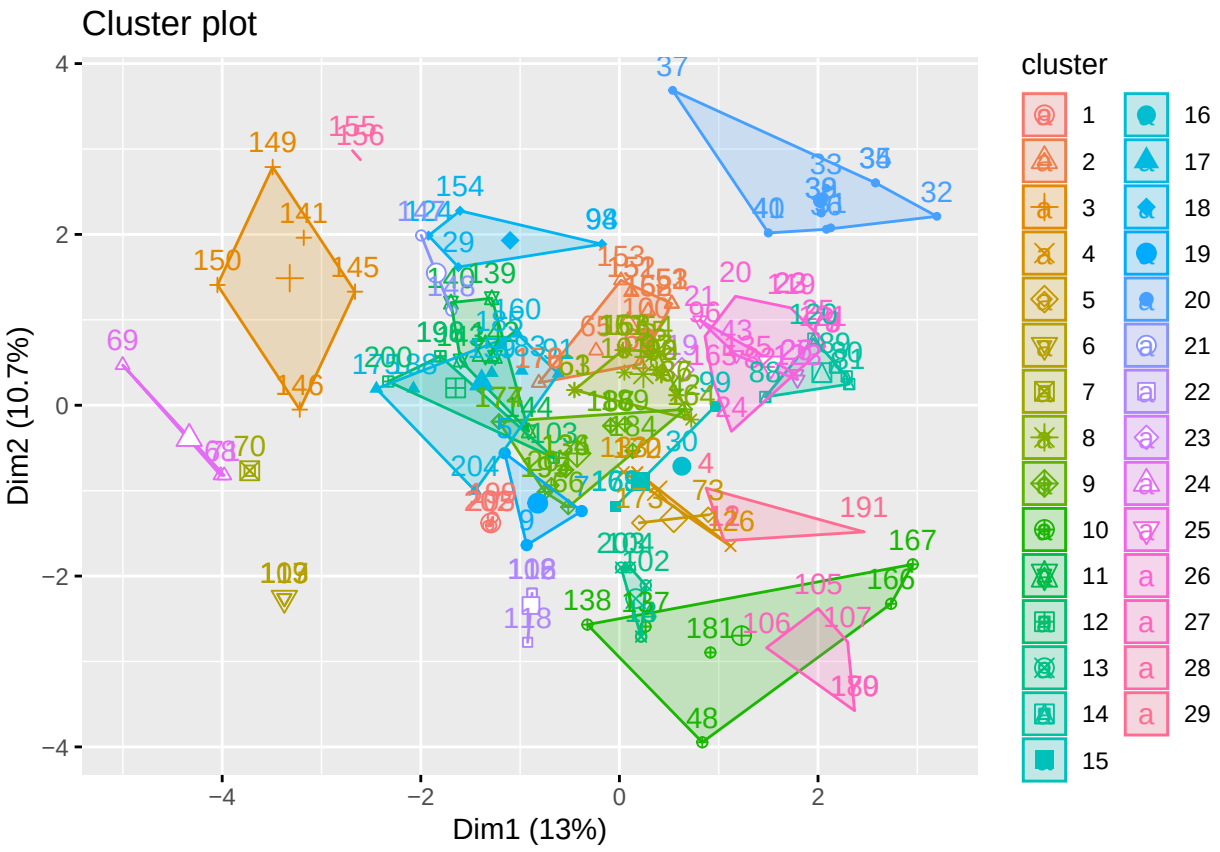

```
#Nao normalizada
# k = 8
final <- kmeans(imports_85sel, centers = 8,  nstart=37)
# Construindo um gráfico de dispersão
fviz_cluster(final, data =imports_85sel)
```




```
# k = 14
final <- kmeans(imports_85sel, centers = 14,  nstart=37)
# Construindo um gráfico de dispersão
fviz_cluster(final, data =imports_85sel)
```




```
final <- kmeans(imports_scaled, centers = 29,  nstart=37)
# Construindo um gráfico de dispersão
fviz_cluster(final, data =imports_85sel)
```



Análises

Descreva cada um dos gráficos gerados nos itens acima e analise-os. Inclua na sua análise as informações mais importantes que podemos retirar desses gráficos. Discuta sobre a escolha do valor K e sobre a apresentação dos dados no gráfico de dispersão.

Resposta: Primeiramente foi gerado os graficos elbow e silhueta, para identificar possiveis valores otimos de K, observa-se que os graficos foram gerados em cima da base nao normalizada e da base normalizada. Para a base nao normalizaad, o grafico elbow estima um valor otimo para K de aproximadamente 8, ja de acordo com o grafico de silhueta, o valor otimo para K seria 14. Para a base normalizada, o grafico elbow nao apresentou a caracterizacao esperada, mas, quando analisado em conjunto com o silhueta, percebe-se que o nuemro ideal sugerido para K eh 29. A funcao DbCluster foi utilizada para que pudesse ser obtido uma outra perspectiva de valor otimo para K, a mesma retornou que o mais indicado eh o valor 2 e soh pode ser executada na base normalizada. A aplicacao do K-means foi realizada para ambas as bases, normalizada e nao normalizada, seguindo os valores indicados como otimos para cada caso, percebe-se desse modo que a clusterizacao para o dado normalizado obteve uma perspectiva interessante, mas ainda sim apresenta sobreposicao dos clusters, ja para a base nao normalizada, observa-se que tanto para k = 8 quanto k = 14, o agrupamento se demonstrou confuso. Por fim, ao observar o comportamento dos dados normalizados nos graficos Elbow e Silhueta e o desempenho da clusterizacao para ambos, estima-se que o algoritmo k-means nao seja o mais adequado para que seja realizado um agrupamento da base.

Atividade 3 – Agrupamento com o *DBscan*

Nesta atividade, você deverá agrupar os dados com o algoritmo *DBscan*. Para isso será necessário experimentar com diferentes valores de *eps* e *minPts*.

a) *Ajuste de Parâmetros:* Experimente com valores diferentes para os parâmetros *eps* e *minPts*. Verifique o impacto dos diferentes valores nos agrupamentos.

```
set.seed(137)

# Experimento com valores de eps e minPts
db <- dbscan(imports_scaled, eps =0.15 , minPts = 2)
 print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 2
## The clustering contains 31 cluster(s) and 89 noise points.
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## 89  2  2  3  2  2  2  2  3  2  2  2  3  2  3  2  2  3  3  2  2  2  3  2  3  2
## 26 27 28 29 30 31
##  2  2  2  2  2  2
##
## Available fields: cluster, eps, minPts
```

```
# Experimento com valores de eps e minPts
db <- dbscan(imports_scaled, eps =0.15 , minPts =3)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 3
## The clustering contains 8 cluster(s) and 135 noise points.
##
##   0   1   2   3   4   5   6   7   8
## 135   3   3   3   3   3   3   3   3
##
## Available fields: cluster, eps, minPts
```

```
db <- dbscan(imports_scaled, eps =0.15 , minPts =4)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 4
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```
db <- dbscan(imports_scaled, eps =0.15 , minPts =5)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 5
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```r
# Experimento com valores de eps e minPts
db <- dbscan(imports_scaled, eps =0.2 , minPts =15)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.2, minPts = 15
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```r
# Experimento com valores de eps e minPts
db <- dbscan(imports_scaled, eps =0.3 , minPts =50)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.3, minPts = 50
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```r
db <- dbscan(imports_85sel, eps =0.15 , minPts = 2)
 print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 2
## The clustering contains 31 cluster(s) and 91 noise points.
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## 91  2  2  2  2  2  2  2  3  2  2  2  3  2  3  2  2  3  3  2  2  2  2  2  3  2
## 26 27 28 29 30 31
##  2  2  2  2  2  2
##
## Available fields: cluster, eps, minPts
```

```r
# Experimento com valores de eps e minPts
db <- dbscan(imports_85sel, eps =0.15 , minPts =3)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 3
## The clustering contains 6 cluster(s) and 141 noise points.
##
##   0   1   2   3   4   5   6
## 141   3   3   3   3   3   3
##
## Available fields: cluster, eps, minPts
```

```r
db <- dbscan(imports_85sel, eps =0.15 , minPts =4)
print(db)
```

```
## DBSCAN clustering for 159 objects.
```

```
## Parameters: eps = 0.15, minPts = 4
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```
db <- dbscan(imports_85sel, eps =0.15 , minPts =5)
print(db)
```

```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.15, minPts = 5
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```
# Experimento com valores de eps e minPts
db <- dbscan(imports_85sel, eps =0.2 , minPts =15)
print(db)
```

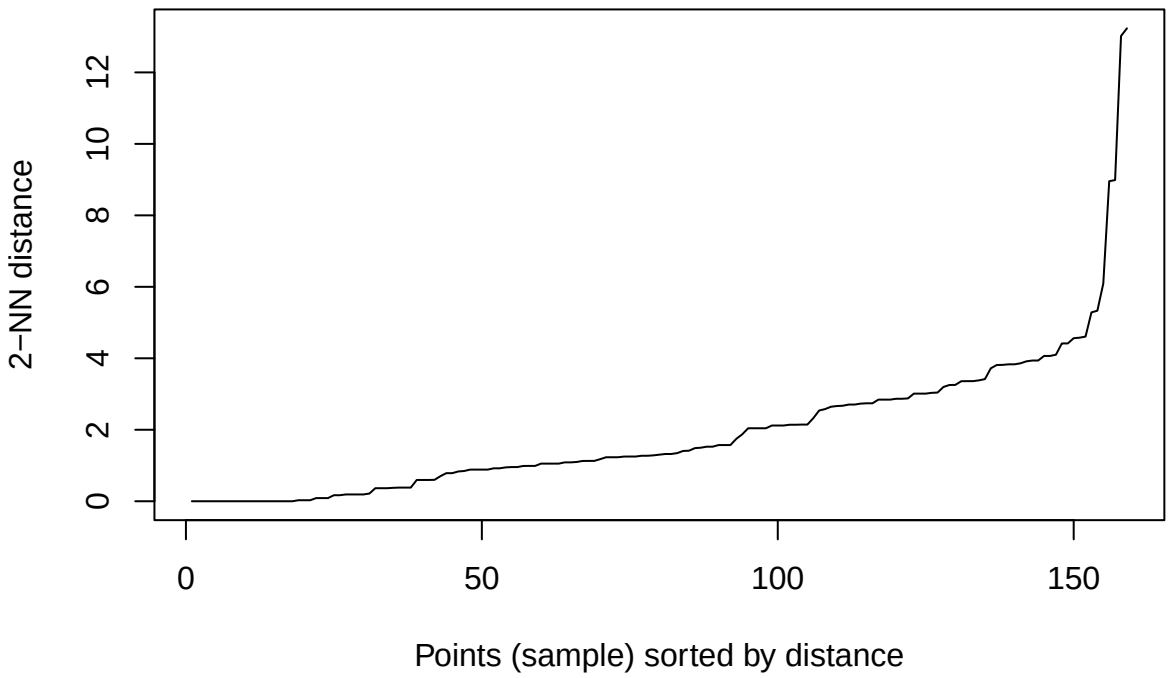
```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.2, minPts = 15
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

```
# Experimento com valores de eps e minPts
db <- dbscan(imports_85sel, eps =0.3 , minPts =50)
print(db)
```

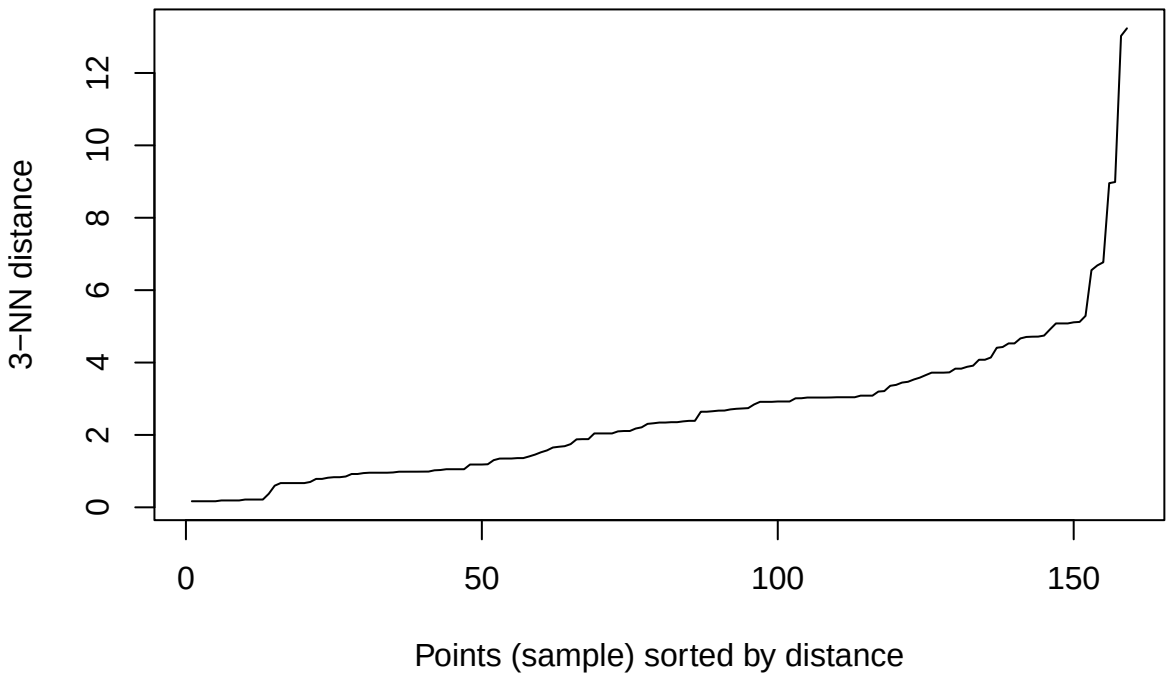
```
## DBSCAN clustering for 159 objects.
## Parameters: eps = 0.3, minPts = 50
## The clustering contains 0 cluster(s) and 159 noise points.
##
##   0
## 159
##
## Available fields: cluster, eps, minPts
```

b) *Determinando Ruídos:* Escolha o valor de *minPts* que obteve o melhor resultado no item anterior e use a função `kNNdistplot` do pacote `dbscan` para determinar o melhor valor de *eps* para esse valor de *minPts*. Lembre-se que o objetivo não é remover todos os ruídos.

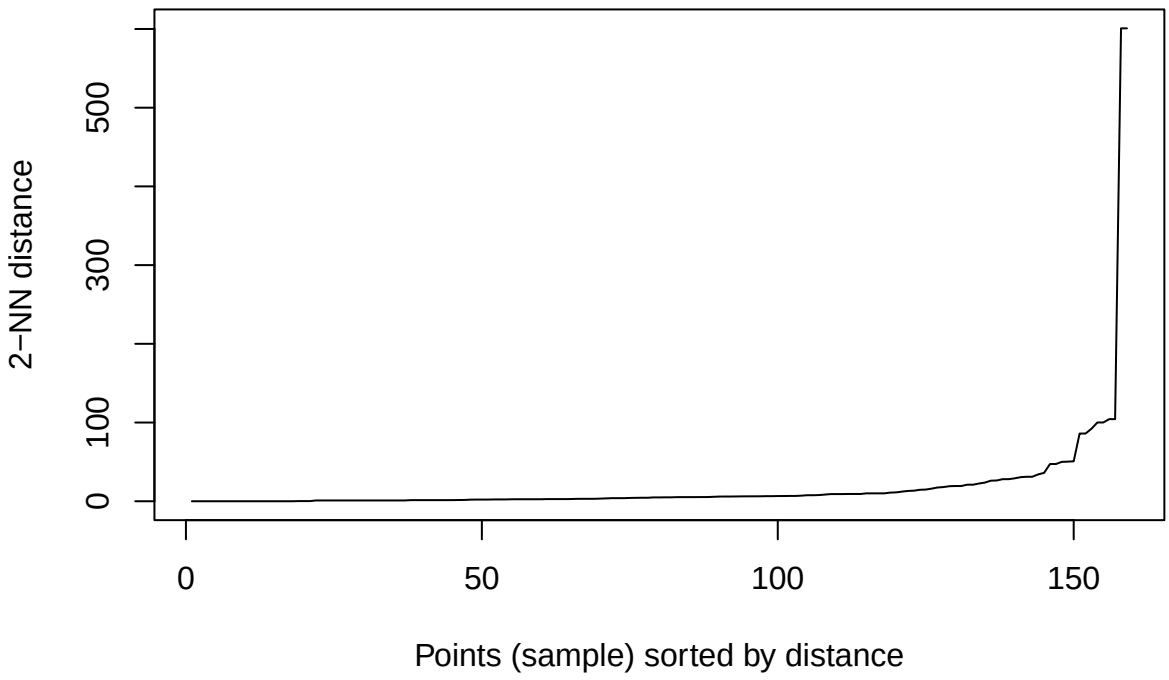
```
# Encontrando o melhor eps com o kNNdistplot
kNNdistplot(imports_scaled, k = 2)
```



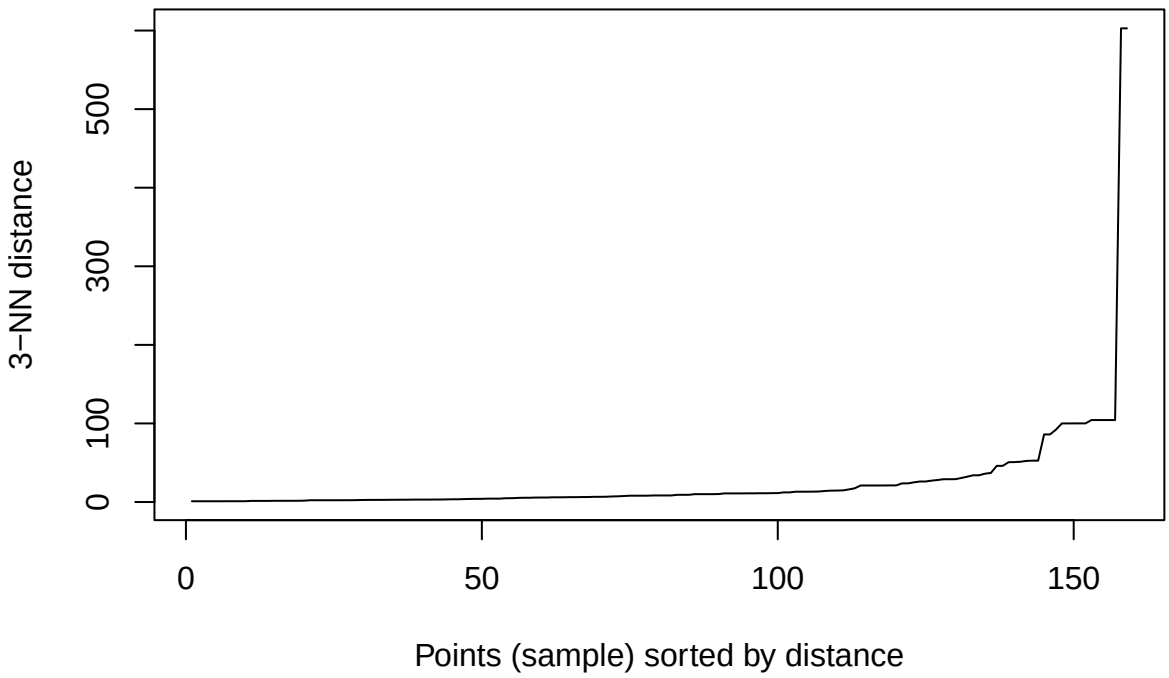

```
kNNdistplot(imports_scaled, k = 3)
```




```
kNNdistplot(imports_85sel, k = 2)
```



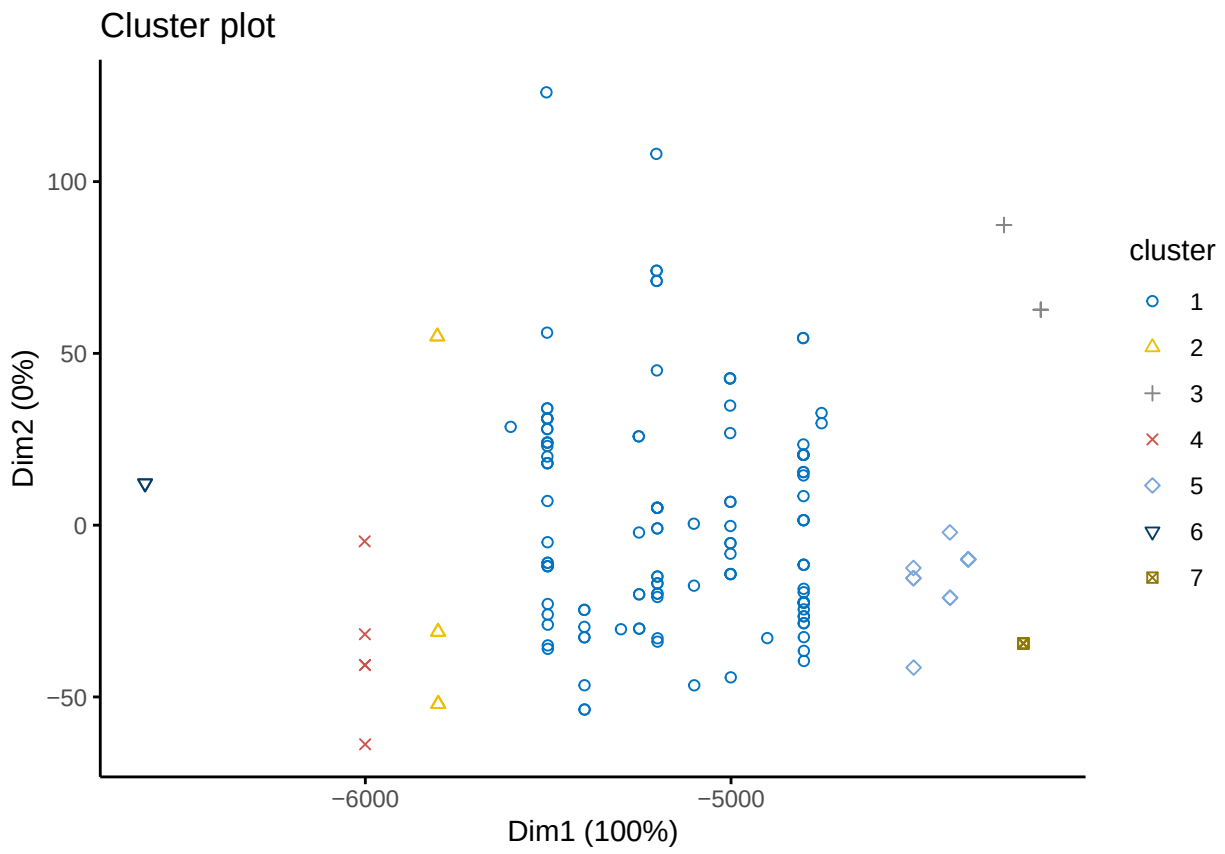

```
kNNdistplot(imports_85sel, k = 3)
```



c) *Visualizando os Grupos:* Após a escolha dos parâmetros *eps* e *minPts*, utilize o rótulo obtido para cada amostra, indicando o grupo ao qual ela pertence, para gerar um gráfico de dispersão (atribuindo cores diferentes para cada grupo).

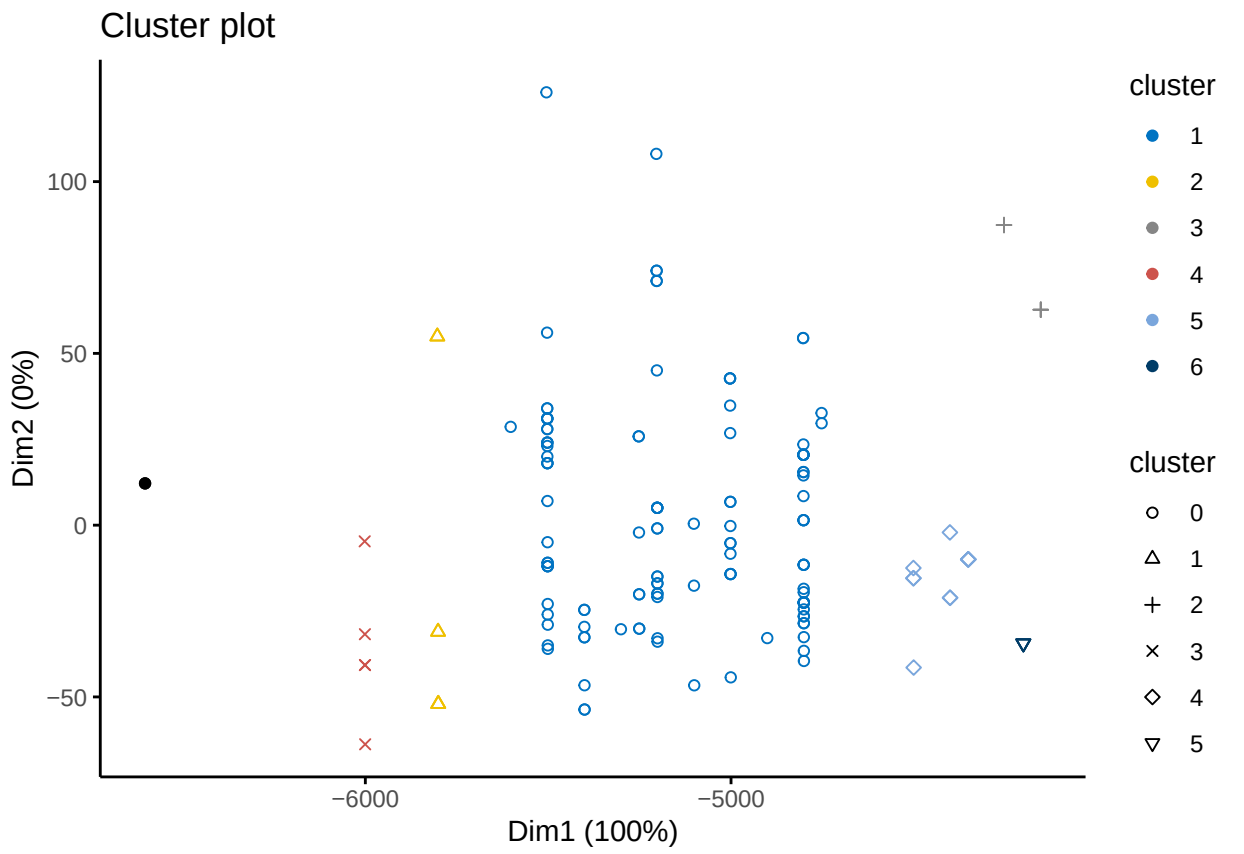
```
#Aplicando DBScan para a base nao normalizada (nn)
db_k2_nn <-dbscan(imports_85sel, eps = 107 , minPts = 2)

# Construindo um gráfico de dispersão
fviz_cluster(db_k2_nn, data = imports_85sel , stand = FALSE, ellipse = FALSE , show.clust.cent = FALSE,
geom ="point", palette ="jco", ggtheme = theme_classic())
```



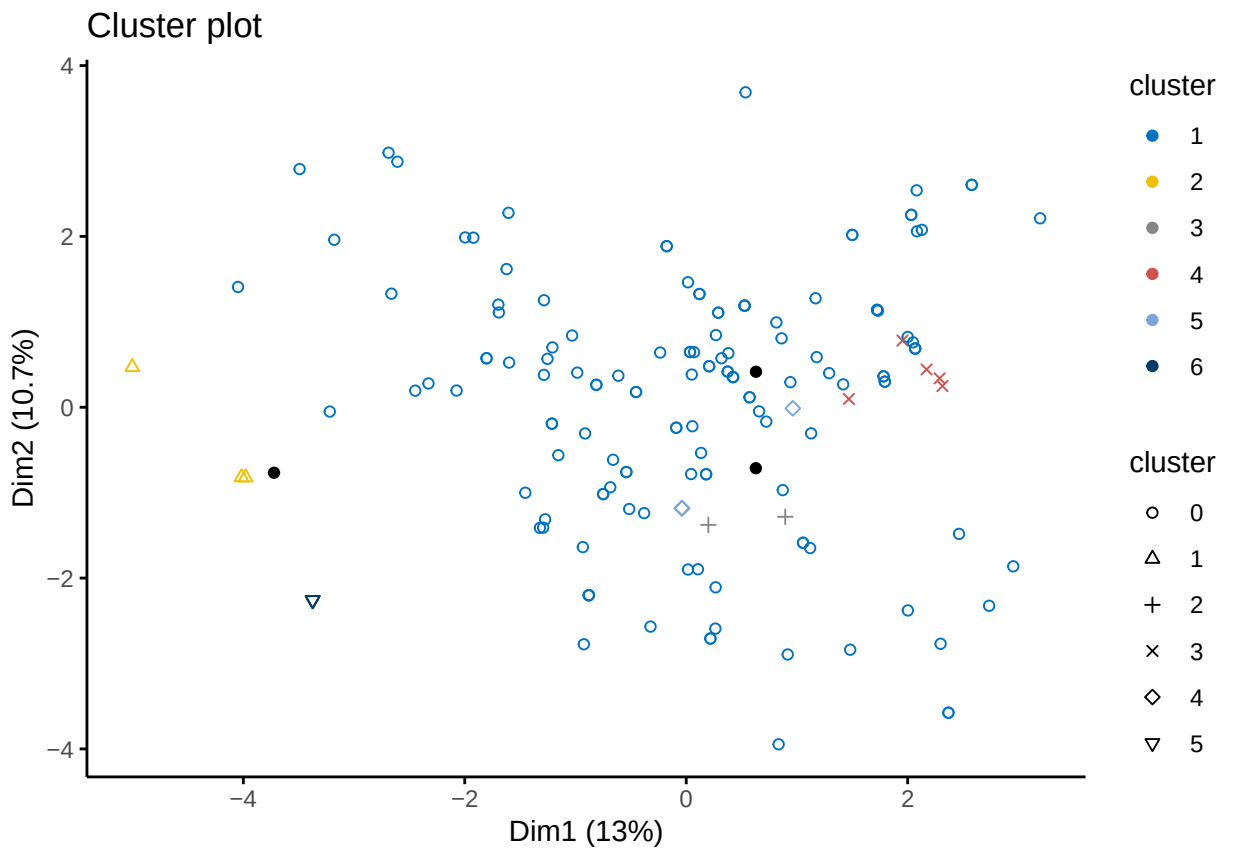

```
# Aplicando o DBscan com os parâmetros escolhidos
db_k3_nn <-dbscan(imports_85sel, eps = 107 , minPts = 3)

# Construindo um gráfico de dispersão
fviz_cluster(db_k3_nn, data = imports_85sel , stand = FALSE, ellipse = FALSE , show.clust.cent = FALSE,
geom ="point", palette ="jco", ggtheme = theme_classic())
```



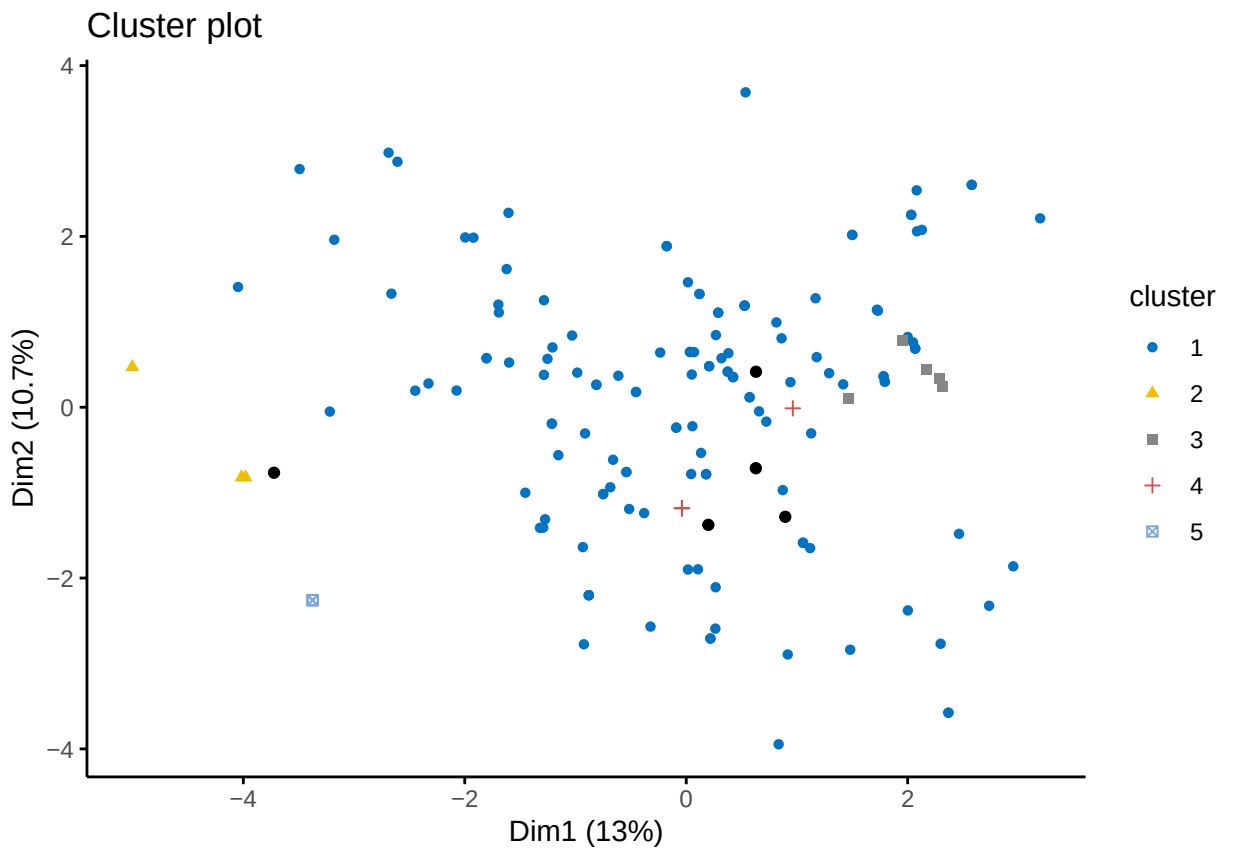

```
# Aplicando o DBscan para a base normalizada
db_k2 <-dbscan(imports_scaled, eps = 5 , minPts = 2)

# Construindo um gráfico de dispersão
fviz_cluster(db_k2, data = imports_scaled , stand = FALSE, ellipse = FALSE , show.clust.cent = FALSE,
geom ="point", palette ="jco", ggtheme = theme_classic())
```




```
# Aplicando o DBscan com os parâmetros escolhidos
db_k3 <-dbscan(imports_scaled, eps = 5 , minPts = 3)

# Construindo um gráfico de dispersão
fviz_cluster(db_k3, data = imports_scaled , stand = FALSE, ellipse = FALSE , show.clust.cent = FALSE,
geom ="point", palette ="jco", ggtheme = theme_classic())
```



Análises

Descreva os experimentos feitos para a escolha dos parâmetros *eps* e *minPts*. Inclua na sua análise as informações mais importantes que podemos retirar dos gráficos gerados. Justifique a escolha dos valores dos parâmetros e analise a apresentação dos dados no gráfico de dispersão.

Resposta: As analises para o DBScan foram realizadas para as bases normalizada e nao normalizada. Inicialmente o algorimto DBScan foi previamente executado com valores arbitrariamente escolhidos, a partir dos resultados, constatou-se que, para ambas as bases, os valores de K mais promissores eram 3 e 2, sendo que para k = 2 foram encontrados 31 grupos nas duas bases, e para k = 3, foram encontrados 6 grupos para a nao normalizada e 8 para a normalizada, ou seja, minPts pode assumir o valor 2 ou 3 para ambos os casos. A partir disso, utilizou-se ou algoritmo kNNdistplot para que fosse possivel identificar os possiveis valores mais adequados para eps, a partir disso constatou-se que para os dados normalizados, o valor ideal seria em torno de 5 e para os nao normalizados, o valor seria em torno de 107. Portanto: - Normalizados: - minPts = 2 ou 3 - eps = 5 - Nao Normalizado: - minpts = 2 ou 3 - eps = 107 A partir desses parametros, aplicou-se novamente o algoritmo DBScan e os resultados foram colocados em grafico. A partir disso, percebeu-se que o resultado que visualmente indica ser o mais adequado, eh o DBScan aplicado aos dados nao normalizados com minPts = 2 e eps = 107, pois nele eh possivel verificar o agrupamento adequadamente separado.

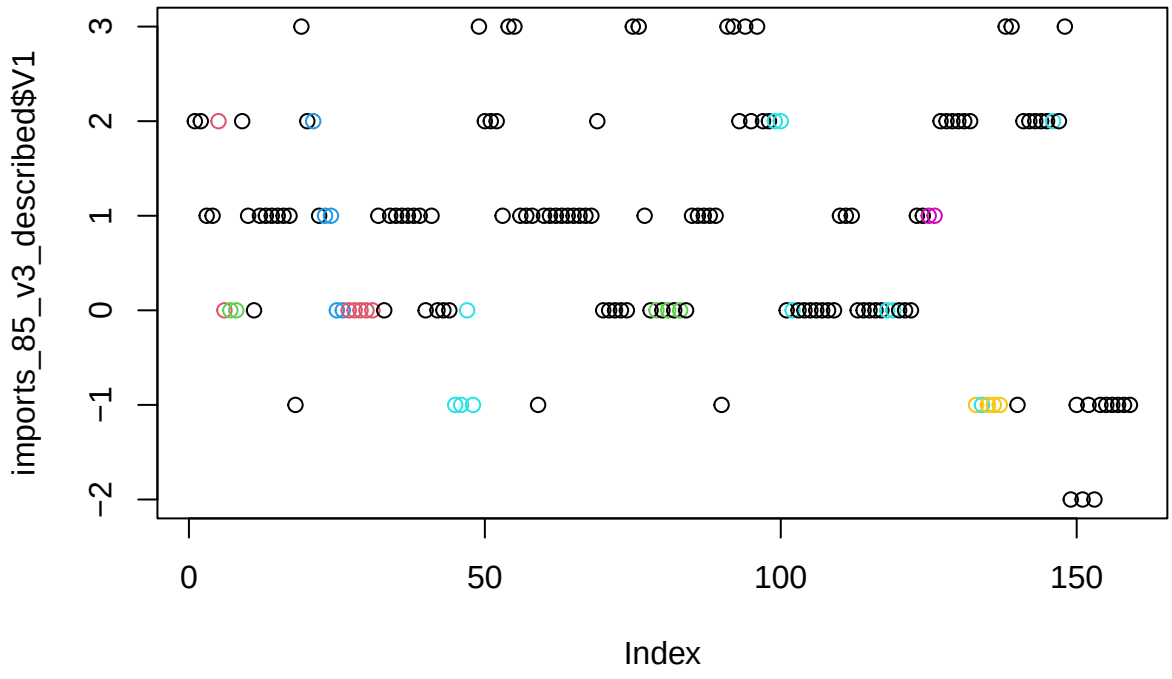
Atividade 4 – Comparando os Algoritmos

```
imports_85_v3_described <- imports_85
imports_85_v3_described$V3[imports_85$V3 == "audi"] <- 1
imports_85_v3_described$V3[imports_85$V3 == "bmw"] <- 2
imports_85_v3_described$V3[imports_85$V3 == "chevrolet"] <- 3
```

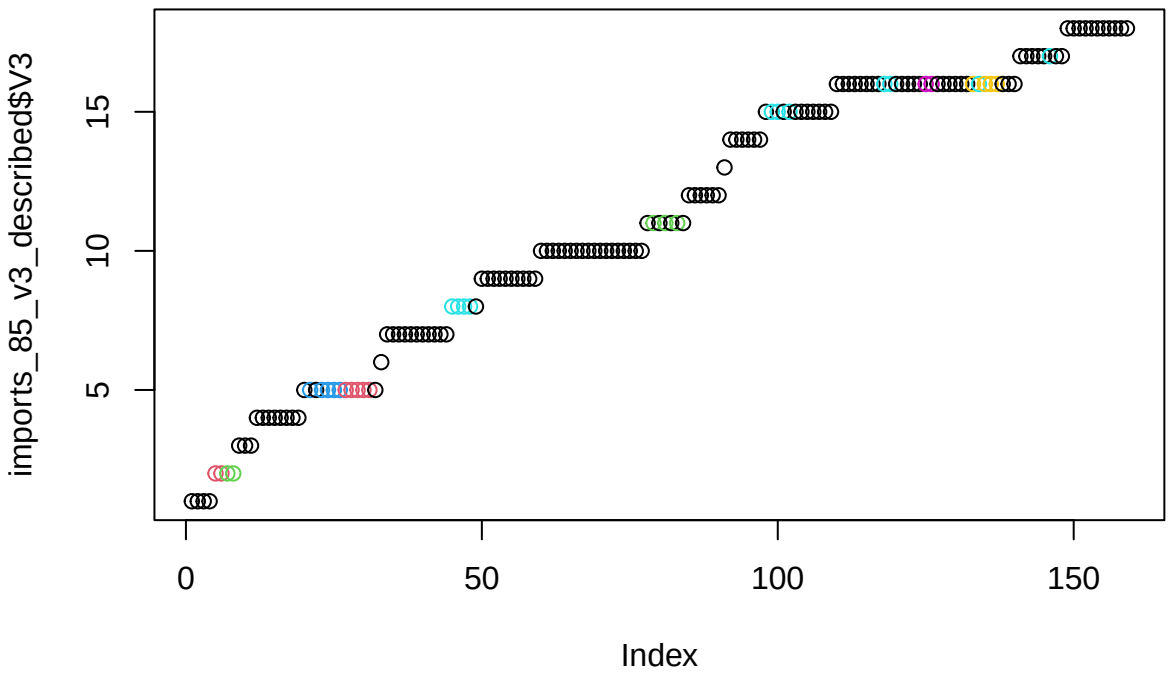
```
imports_85_v3_described$V3[imports_85$V3 == "dodge"] <- 4
imports_85_v3_described$V3[imports_85$V3 == "honda"] <- 5
imports_85_v3_described$V3[imports_85$V3 == "jaguar"] <- 6
imports_85_v3_described$V3[imports_85$V3 == "mazda"] <- 7
imports_85_v3_described$V3[imports_85$V3 == "mercedes-benz"] <- 8
imports_85_v3_described$V3[imports_85$V3 == "mitsubishi"] <- 9
imports_85_v3_described$V3[imports_85$V3 == "nissan"] <- 10
imports_85_v3_described$V3[imports_85$V3 == "peugot"] <- 11
imports_85_v3_described$V3[imports_85$V3 == "plymouth"] <- 12
imports_85_v3_described$V3[imports_85$V3 == "porsche"] <- 13
imports_85_v3_described$V3[imports_85$V3 == "saab"] <- 14
imports_85_v3_described$V3[imports_85$V3 == "subaru"] <- 15
imports_85_v3_described$V3[imports_85$V3 == "toyota"] <- 16
imports_85_v3_described$V3[imports_85$V3 == "volkswagen"] <- 17
imports_85_v3_described$V3[imports_85$V3 == "volvo"] <- 18

imports_clistered <- cbind(imports_85, db_k2_nn$cluster)
imports_clistered$cluster <- imports_clistered$`db_k2_nn$cluster`

plot(imports_85_v3_described$V1, col=db_k2_nn$cluster)
```

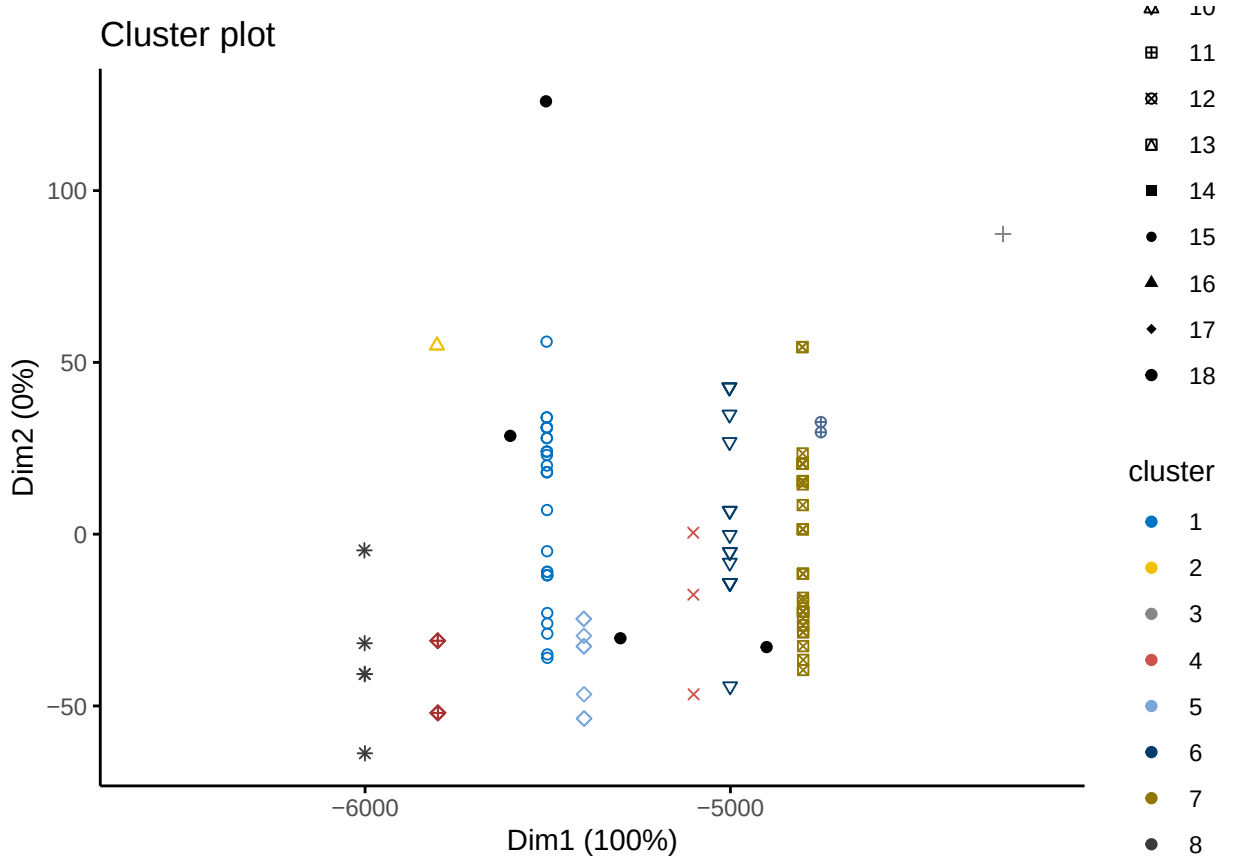



```
plot(imports_85_v3_described$V3, col=db_k2_nn$cluster)
```

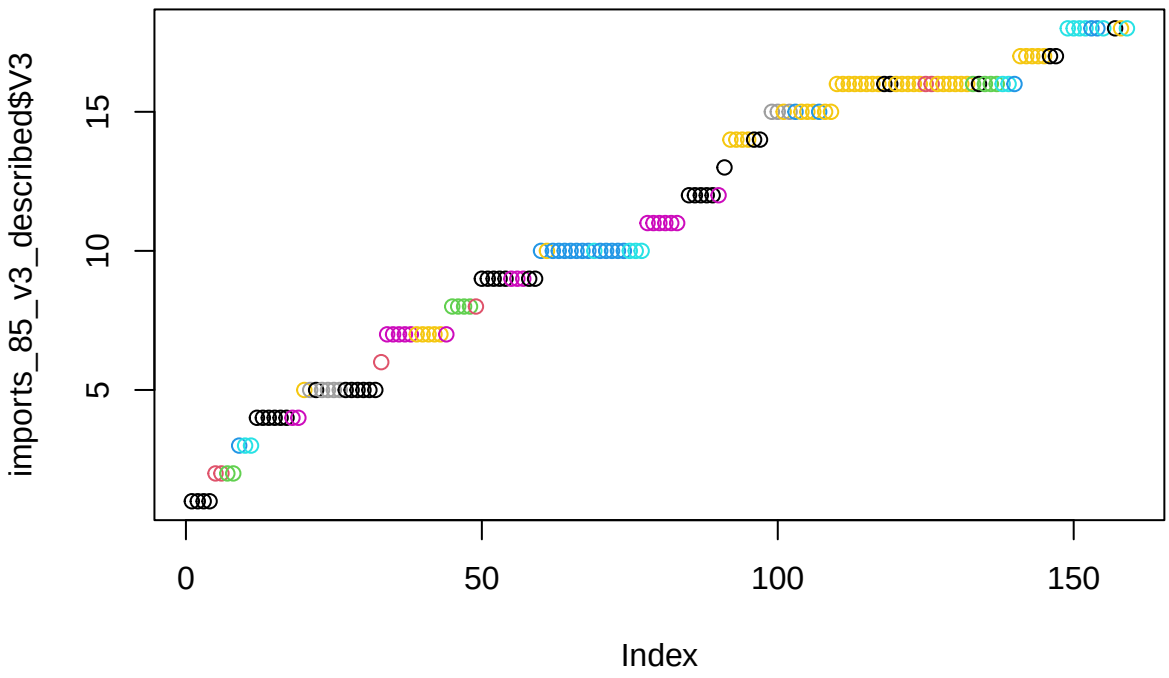



```
db_k18_nn <-dbscan(imports_85sel, eps = 37 , minPts = 2)

# Construindo um gráfico de dispersão
fviz_cluster(db_k18_nn, data = imports_85sel , stand = FALSE, ellipse = FALSE , show.clust.cent = FALSE
geom ="point", palette ="jco", ggtheme = theme_classic())
```




```
plot(imports_85_v3_described$V3, col=db_k18_nn$cluster)
```



Com base nas atividades anteriores, faça uma conclusão dos seus experimentos respondendo às seguintes perguntas:

a) Qual dos métodos apresentou melhores resultados? Justifique.

b) Quantos agrupamentos foram obtidos?

c) Analisando o campo `symboling` e o grupo designado para cada amostra, os agrupamentos conseguiram separar os níveis de risco?

d) Analisando o campo `make` que contém as marcas dos carros, os agrupamentos conseguiram separar as marcas?

Respostas:

a) O metodo que apresentou os melhores resulados foi o DBSCAN, o resultado visual do agrupamento com minPts = 2 e eps = 107 demonstrou uma adequada clusterizacao com 7 agrupamentos.

b) No caso do algoritmo com melhor desempenho, foi obtido 7 agrupamentos.

c) Considerando que na base fornecida, haviam apenas 6 categorias para o campo Symboling, pode-se dizer que sim, pois o algoritmo foi capaz de gerar a mesma quantidade de grupos. No entanto na base estao presentes apenas 6 valores (-2 1 0 -1 3 -2), ou seja, o setimos grupo eh provavelmente correspondente a outliers, o que, de certo modo, ainda se encaixa na solucao. Porem o grafico indica que os clusters nao estao, de fato, separando por Symboling, ja que pode-se encontrar uma mistura de cores para algumas categorias, ainda assim, percebe-se que a categoria -1 por exemplo eh a unica que contem a cor amarela, ou que a categoria 0 eh a unica que contem a cor vermelha. mas aidna sim, a cor preta eh a mais presente e domina os extremos (-2 e 3). Portanto, para a calssificacao adequada do Symboling, seria necessario observar mais profundamente as features, ou propor outro algoritmo.

d) O agrupamento nao foi capaz de separar os carros por marca adequadamente, apesar de haver 18 marcas, e nenhum dos agrupamentos com valores apontados como ideais tendeu a k = 18. Porem, a

partir da tentativa e erro, foi possivel obter 18 clusters com minPts = 2 e eps = 37 para o algoritmo DBScan. O modelo de agrupamento foi entao utilizado para produzir o ultimo grafico, que indica que eh possivel, de certo modo agrupar por montadora do carro, no entanto, o algoritmo ainda sim aparenta possuir uma aciracia baixa.