INF0613 – Aprendizado de Máquina Não Supervisionado

Trabalho 3 - Técnicas de Agrupamento

Evandro Santos Rocha Laíssa Pacheco de Oliveira Rafael Dantas de Moura

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(datasets)
library(mlbench)
library(caret)
library(ggplot2)

library(cluster)
library(factoextra)
library(NbClust)

library(dbscan)

# Configure ambiente de trabalho na mesma pasta
# onde colocou a base de dados:
# setwd("")
```

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).

```r
# Leitura da base
#carros <- read.table(file.choose(), sep=",")
carros <- read.table("imports-85.data", sep=",")
summary(carros)
```

```
##        V1              V2                V3                V4
##  Min.   :-2.000   Length:205         Length:205         Length:205
##  1st Qu.: 0.000   Class :character   Class :character   Class :character
##  Median : 1.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 0.834
##  3rd Qu.: 2.000
##  Max.   : 3.000
##       V5                V6                V7                V8
##  Length:205         Length:205         Length:205         Length:205
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       V9                V10              V11            V12             V13
##  Length:205         Min.   : 86.6   Min.   :141   Min.   :60.3   Min.   :47.8
##  Class :character   1st Qu.: 94.5   1st Qu.:166   1st Qu.:64.1   1st Qu.:52.0
##  Mode  :character   Median : 97.0   Median :173   Median :65.5   Median :54.1
##                     Mean   : 98.8   Mean   :174   Mean   :65.9   Mean   :53.7
##                     3rd Qu.:102.4   3rd Qu.:183   3rd Qu.:66.9   3rd Qu.:55.5
##                     Max.   :120.9   Max.   :208   Max.   :72.3   Max.   :59.8
##       V14            V15                V16                V17
##  Min.   :1488   Length:205         Length:205         Min.   : 61
##  1st Qu.:2145   Class :character   Class :character   1st Qu.: 97
##  Median :2414   Mode  :character   Mode  :character   Median :120
##  Mean   :2556                                         Mean   :127
##  3rd Qu.:2935                                         3rd Qu.:141
##  Max.   :4066                                         Max.   :326
##      V18                V19                V20                V21
##  Length:205         Length:205         Length:205         Min.   : 7.0
##  Class :character   Class :character   Class :character   1st Qu.: 8.6
##  Mode  :character   Mode  :character   Mode  :character   Median : 9.0
##                                                           Mean   :10.1
##                                                           3rd Qu.: 9.4
##                                                           Max.   :23.0
##      V22                V23                V24            V25
##  Length:205         Length:205         Min.   :13.0   Min.   :16.0
##  Class :character   Class :character   1st Qu.:19.0   1st Qu.:25.0
##  Mode  :character   Mode  :character   Median :24.0   Median :30.0
##                                        Mean   :25.2   Mean   :30.8
##                                        3rd Qu.:30.0   3rd Qu.:34.0
##                                        Max.   :49.0   Max.   :54.0
##      V26
##  Length:205
##  Class :character
##  Mode  :character
##
##
```

##

```
dim(carros)
```

```
## [1] 205  26
```

```
#[1] 205  26

# Tratamento de dados faltantes
any(is.na(carros)) # verifica se há algum valor NA no conjunto de dados
```

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

```
# FALSE

# Verificando quais são os valores não numéricos de cada campo
# que deveria vir como numérico no summary (de acordo com o enunciado)
carros[is.na(as.numeric(carros$V2)),]$V2      # possui 41 valores iguais "?" ---> retirar a feature
```

```
##  [1] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
## [20] "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?" "?"
## [39] "?" "?" "?"
```

```
carros[is.na(as.numeric(carros$V19)),]$V19
```

```
## [1] "?" "?" "?" "?"
```

```
carros[is.na(as.numeric(carros$V20)),]$V20
```

```
## [1] "?" "?" "?" "?"
```

```
carros[is.na(as.numeric(carros$V22)),]$V22
```

```
## [1] "?" "?"
```

```
carros[is.na(as.numeric(carros$V23)),]$V23
```

```
## [1] "?" "?"
```

```
carros[is.na(as.numeric(carros$V26)),]$V26
```

```
## [1] "?" "?" "?" "?"
```

```r
# Encontrando valores iguais a "?" e substituindo por NA
carros$V2[which(carros$V2 == "?")] <- NA
carros$V19[which(carros$V19 == "?")] <- NA
carros$V20[which(carros$V20 == "?")] <- NA
carros$V22[which(carros$V22 == "?")] <- NA
carros$V23[which(carros$V23 == "?")] <- NA
carros$V26[which(carros$V26 == "?")] <- NA

# Conversão do tipo dos atributos
carros$V2 <- as.numeric(carros$V2)
carros$V19 <- as.numeric(carros$V19)
carros$V20 <- as.numeric(carros$V20)
carros$V22 <- as.numeric(carros$V22)
carros$V23 <- as.numeric(carros$V23)
carros$V26 <- as.numeric(carros$V26)

# Verificando se todos os campos numéricos estão ok
summary(carros)
```

```
##        V1                V2           V3                 V4
##  Min.   :-2.000   Min.   : 65   Length:205         Length:205
##  1st Qu.: 0.000   1st Qu.: 94   Class :character   Class :character
##  Median : 1.000   Median :115   Mode  :character   Mode  :character
##  Mean   : 0.834   Mean   :122
##  3rd Qu.: 2.000   3rd Qu.:150
##  Max.   : 3.000   Max.   :256
##                   NA's   :41
##       V5                 V6                 V7                 V8
##  Length:205         Length:205         Length:205         Length:205
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##       V9                 V10              V11            V12            V13
##  Length:205         Min.   : 86.6   Min.   :141   Min.   :60.3   Min.   :47.8
##  Class :character   1st Qu.: 94.5   1st Qu.:166   1st Qu.:64.1   1st Qu.:52.0
##  Mode  :character   Median : 97.0   Median :173   Median :65.5   Median :54.1
##                     Mean   : 98.8   Mean   :174   Mean   :65.9   Mean   :53.7
##                     3rd Qu.:102.4   3rd Qu.:183   3rd Qu.:66.9   3rd Qu.:55.5
##                     Max.   :120.9   Max.   :208   Max.   :72.3   Max.   :59.8
##
##       V14            V15                V16                 V17
##  Min.   :1488   Length:205         Length:205         Min.   : 61
##  1st Qu.:2145   Class :character   Class :character   1st Qu.: 97
##  Median :2414   Mode  :character   Mode  :character   Median :120
##  Mean   :2556                                         Mean   :127
##  3rd Qu.:2935                                         3rd Qu.:141
##  Max.   :4066                                         Max.   :326
##
##      V18                V19             V20            V21            V22
##  Length:205         Min.   :2.54   Min.   :2.07   Min.   : 7.0   Min.   : 48
```

```
##  Class :character   1st Qu.:3.15   1st Qu.:3.11   1st Qu.: 8.6   1st Qu.: 70
##  Mode  :character   Median :3.31   Median :3.29   Median : 9.0   Median : 95
##                     Mean   :3.33   Mean   :3.26   Mean   :10.1   Mean   :104
##                     3rd Qu.:3.59   3rd Qu.:3.41   3rd Qu.: 9.4   3rd Qu.:116
##                     Max.   :3.94   Max.   :4.17   Max.   :23.0   Max.   :288
##                     NA's   :4      NA's   :4                     NA's   :2
##       V23            V24            V25            V26
##  Min.   :4150   Min.   :13.0   Min.   :16.0   Min.   : 5118
##  1st Qu.:4800   1st Qu.:19.0   1st Qu.:25.0   1st Qu.: 7775
##  Median :5200   Median :24.0   Median :30.0   Median :10295
##  Mean   :5125   Mean   :25.2   Mean   :30.8   Mean   :13207
##  3rd Qu.:5500   3rd Qu.:30.0   3rd Qu.:34.0   3rd Qu.:16500
##  Max.   :6600   Max.   :49.0   Max.   :54.0   Max.   :45400
##  NA's   :2                                    NA's   :4
```

```
# Atributo V2 possui muitos NAs, então resolvemos retirá-lo
carros$V2 <- NULL

# Tratamento de NA: será atribuída a média
carros$V19[is.na(carros$V19)] <- mean(carros$V19, na.rm = TRUE)
carros$V20[is.na(carros$V20)] <- mean(carros$V20, na.rm = TRUE)
carros$V22[is.na(carros$V22)] <- mean(carros$V22, na.rm = TRUE)
carros$V23[is.na(carros$V23)] <- mean(carros$V23, na.rm = TRUE)
carros$V26[is.na(carros$V26)] <- mean(carros$V26, na.rm = TRUE)

# Verificando se sobraram NAs
any(is.na(carros))
```

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

```
# Checando o summary novamente
summary(carros)
```

```
##        V1              V3                 V4                 V5
##  Min.   :-2.000   Length:205         Length:205         Length:205
##  1st Qu.: 0.000   Class :character   Class :character   Class :character
##  Median : 1.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 0.834
##  3rd Qu.: 2.000
##  Max.   : 3.000
##       V6                 V7                 V8                 V9
##  Length:205         Length:205         Length:205         Length:205
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       V10             V11            V12            V13            V14
##  Min.   : 86.6   Min.   :141   Min.   :60.3   Min.   :47.8   Min.   :1488
##  1st Qu.: 94.5   1st Qu.:166   1st Qu.:64.1   1st Qu.:52.0   1st Qu.:2145
##  Median : 97.0   Median :173   Median :65.5   Median :54.1   Median :2414
##  Mean   : 98.8   Mean   :174   Mean   :65.9   Mean   :53.7   Mean   :2556
##  3rd Qu.:102.4   3rd Qu.:183   3rd Qu.:66.9   3rd Qu.:55.5   3rd Qu.:2935
```

```
##  Max.   :120.9   Max.   :208   Max.   :72.3   Max.   :59.8   Max.   :4066
##      V15                V16                V17          V18
##  Length:205         Length:205         Min.   : 61   Length:205
##  Class :character   Class :character   1st Qu.: 97   Class :character
##  Mode  :character   Mode  :character   Median :120   Mode  :character
##                                        Mean   :127
##                                        3rd Qu.:141
##                                        Max.   :326
##       V19            V20            V21            V22           V23
##  Min.   :2.54   Min.   :2.07   Min.   : 7.0   Min.   : 48   Min.   :4150
##  1st Qu.:3.15   1st Qu.:3.11   1st Qu.: 8.6   1st Qu.: 70   1st Qu.:4800
##  Median :3.31   Median :3.29   Median : 9.0   Median : 95   Median :5200
##  Mean   :3.33   Mean   :3.26   Mean   :10.1   Mean   :104   Mean   :5125
##  3rd Qu.:3.58   3rd Qu.:3.41   3rd Qu.: 9.4   3rd Qu.:116   3rd Qu.:5500
##  Max.   :3.94   Max.   :4.17   Max.   :23.0   Max.   :288   Max.   :6600
##       V24            V25            V26
##  Min.   :13.0   Min.   :16.0   Min.   : 5118
##  1st Qu.:19.0   1st Qu.:25.0   1st Qu.: 7788
##  Median :24.0   Median :30.0   Median :10595
##  Mean   :25.2   Mean   :30.8   Mean   :13207
##  3rd Qu.:30.0   3rd Qu.:34.0   3rd Qu.:16500
##  Max.   :49.0   Max.   :54.0   Max.   :45400
```

```
dim(carros)
```

```
## [1] 205  25
```

```
#[1] 205  25
```

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.

```
####################################
# Transformação em one-hot-encoding
####################################

#fuel-type
unique(carros$V4)
```

```
## [1] "gas"    "diesel"
```

```
#[1] "gas"    "diesel"
carros$V4_gas    <- as.numeric(carros$V4 == "gas")
carros$V4_diesel <- as.numeric(carros$V4 == "diesel")
carros$V4 <- NULL

#aspiration
unique(carros$V5)
```

```
## [1] "std"   "turbo"
```

```
#[1] "std"   "turbo"
carros$V5_std   <- as.numeric(carros$V5 == "std")
carros$V5_turbo <- as.numeric(carros$V5 == "turbo")
carros$V5 <- NULL

#num-of-doors
unique(carros$V6)
```

```
## [1] "two"  "four" "?"
```

```
#table(carros$V6)
#   ? four  two
#   2  114   89
#[1] "two"  "four" "?"
carros$V6_two  <- as.numeric(carros$V6 == "two")
carros$V6_four <- as.numeric(carros$V6 == "four" | carros$V6 == "?")
carros$V6 <- NULL

#body-style
unique(carros$V7)
```

```
## [1] "convertible" "hatchback"   "sedan"       "wagon"       "hardtop"
```

```
#[1] "convertible" "hatchback"   "sedan"       "wagon"       "hardtop"
carros$V7_convertible <- as.numeric(carros$V7 == "convertible")
carros$V7_hatchback   <- as.numeric(carros$V7 == "hatchback")
carros$V7_sedan       <- as.numeric(carros$V7 == "sedan")
carros$V7_wagon       <- as.numeric(carros$V7 == "wagon")
carros$V7_hardtop     <- as.numeric(carros$V7 == "hardtop")
carros$V7 <- NULL

#drive-wheels
unique(carros$V8)
```

```
## [1] "rwd" "fwd" "4wd"
```

```
#[1] "rwd" "fwd" "4wd"
carros$V8_rwd <- as.numeric(carros$V8 == "rwd")
carros$V8_fwd <- as.numeric(carros$V8 == "fwd")
carros$V8_4wd <- as.numeric(carros$V8 == "4wd")
carros$V8 <- NULL

#engine-location
unique(carros$V9)
```

```
## [1] "front" "rear"
```

```
#[1] "front" "rear"
carros$V9_front <- as.numeric(carros$V9 == "front")
carros$V9_rear  <- as.numeric(carros$V9 == "rear")
carros$V9 <- NULL

#engine-type
unique(carros$V15)
```

```
## [1] "dohc"  "ohcv"  "ohc"   "l"     "rotor" "ohcf"  "dohcv"
```

```
#[1] "dohc"  "ohcv"  "ohc"   "l"     "rotor" "ohcf"  "dohcv"
carros$V15_dohc  <- as.numeric(carros$V15 == "dohc")
carros$V15_ohcv  <- as.numeric(carros$V15 == "ohcv")
carros$V15_ohc   <- as.numeric(carros$V15 == "ohc")
carros$V15_l     <- as.numeric(carros$V15 == "l")
carros$V15_rotor <- as.numeric(carros$V15 == "rotor")
carros$V15_ohcf  <- as.numeric(carros$V15 == "ohcf")
carros$V15_dohcv <- as.numeric(carros$V15 == "dohcv")
carros$V15 <- NULL

#num-of-cylinders
unique(carros$V16)
```

```
## [1] "four"   "six"    "five"   "three"  "twelve" "two"    "eight"
```

```
#[1] "four"   "six"    "five"   "three"  "twelve" "two"    "eight"
carros$V16_four   <- as.numeric(carros$V16 == "four")
carros$V16_six    <- as.numeric(carros$V16 == "six")
carros$V16_five   <- as.numeric(carros$V16 == "five")
carros$V16_three  <- as.numeric(carros$V16 == "three")
carros$V16_twelve <- as.numeric(carros$V16 == "twelve")
carros$V16_two    <- as.numeric(carros$V16 == "two")
carros$V16_eight  <- as.numeric(carros$V16 == "eight")
carros$V16 <- NULL

#fuel-system
unique(carros$V18)
```

```
## [1] "mpfi" "2bbl" "mfi"  "1bbl" "spfi" "4bbl" "idi"  "spdi"
```

```
#[1] "mpfi" "2bbl" "mfi"  "1bbl" "spfi" "4bbl" "idi"  "spdi"
carros$V18_mpfi <- as.numeric(carros$V18 == "mpfi")
carros$V18_2bbl <- as.numeric(carros$V18 == "2bbl")
carros$V18_mfi  <- as.numeric(carros$V18 == "mfi")
carros$V18_1bbl <- as.numeric(carros$V18 == "1bbl")
carros$V18_spfi <- as.numeric(carros$V18 == "spfi")
carros$V18_4bbl <- as.numeric(carros$V18 == "4bbl")
carros$V18_idi  <- as.numeric(carros$V18 == "idi")
carros$V18_spdi <- as.numeric(carros$V18 == "spdi")
carros$V18 <- NULL
```

```
# Verificação de duplicados
dim(carros)
```

```
## [1] 205  54
```

```
#[1] 205  54
dim(unique(carros))
```

```
## [1] 205  54
```

```
#[1] 205  54
# RESULTADO: não há observações duplicadas

##########################
# Seleção de atributos
##########################

carros_atributos <- carros[,3:length(carros)]

#Agora vem a seleção dos atributos
matriz_corr <- cor(carros_atributos)
#print(matriz_corr)

altamente_corr <- findCorrelation(matriz_corr, cutoff = 0.5)
print(altamente_corr)
```

```
##  [1]  5  2 10  3  1 13  6 12 14 26 27 45 38 15 16 51  4 22 19 20 17 29  8 35 50
```

```
#[1]  5  2 10  3  1 13  6 12 14 26 27 45 38 15 16 51  4 22 19 20 17 29  8 35 50

# atributos que estão altamente correlacionados e que podem ser excluídos
names(carros_atributos[altamente_corr])
```

```
##  [1] "V14"          "V11"          "V22"          "V12"          "V10"
##  [6] "V25"          "V17"          "V24"          "V26"          "V8_rwd"
## [11] "V8_fwd"       "V18_mpfi"     "V16_four"     "V4_gas"       "V4_diesel"
## [16] "V18_idi"      "V13"          "V7_hatchback" "V6_two"       "V6_four"
## [21] "V5_std"       "V9_front"     "V20"          "V15_rotor"    "V18_4bbl"
```

```
# [1] "V14"          "V11"          "V22"          "V12"          "V10"          "V25"
# [7] "V17"          "V24"          "V26"          "V8_rwd"       "V8_fwd"       "V18_mpfi"
#[13] "V16_four"     "V4_gas"       "V4_diesel"    "V18_idi"      "V13"          "V7_hatchback"
#[19] "V6_two"       "V6_four"      "V5_std"       "V9_front"     "V20"          "V15_rotor"
#[25] "V18_4bbl"

dim(carros)
```

```
## [1] 205  54
```

```
#[1] 205  54

#Conjunto de atributos
setdiff(names(carros), names(carros_atributos[altamente_corr]))
```

```
##  [1] "V1"            "V3"            "V19"           "V21"
##  [5] "V23"           "V5_turbo"      "V7_convertible" "V7_sedan"
##  [9] "V7_wagon"      "V7_hardtop"    "V8_4wd"        "V9_rear"
## [13] "V15_dohc"      "V15_ohcv"      "V15_ohc"       "V15_l"
## [17] "V15_ohcf"      "V15_dohcv"     "V16_six"       "V16_five"
## [21] "V16_three"     "V16_twelve"    "V16_two"       "V16_eight"
## [25] "V18_2bbl"      "V18_mfi"       "V18_1bbl"      "V18_spfi"
## [29] "V18_spdi"
```

```
carros <- carros[, setdiff(names(carros), names(carros_atributos[altamente_corr]))]

dim(carros)
```

```
## [1] 205  29
```

```
#[1] 205  29

# Verificando se agora, depois de reduzir as dimensões, existem exemplos repetidos
dim(unique(carros))
```

```
## [1] 150  29
```

```
#[1] 150  29

# Desconsiderando exemplos repetidos
carros <- unique(carros)
```

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 conjunto de dados é composto por 205 objetos e 26 atributos referentes a diferentes carros e suas características. Como informado na Tabela 1 do enunciado do exercício, o primeiro atributo (que é do tipo inteiro) refere-se ao risco calculado associado à cada carro, e seu valor pode variar entre -3 (menor risco) e 3 (maior risco).

Primeiramente, observamos os dados de forma resumida, através da função `summary`. Notamos que alguns atributos tem o tipo divergente daqueles indicados na Tabela 1, que descreve o conjunto de dados; como é o caso dos atributos V2, V19, V20, V22, V23, e V26. Na descrição, todos estão indicados como numéricos, mas a função summary retornou este como sendo do tipo *character*, indicando que deve haver valores não numéricos que alteraram inteiramente o tipo do atributo.

Para corrigir esses valores incorretos, primeiro fizemos a conferência se havia algum valor "NA", o que nos retornou *false*. Olhando o conjunto de dados, nos atributos mencionados assim, notamos que apesar de não haver valores vazios ou "NA", havia, porém, valores iguais a "?", explicando porque alguns atributos numéricos estavam sendo tratados como caracteres. Para facilitar a manipulação, todos esses valores foram convertidos para NA, pois assim, pela função `summary`, poderemos obter o valor médio dos atributos e atribuir este valor para as observações faltantes.

Outro tratamento importante é a verificação de amostras duplicadas. Aplicamos a função `unique`, cuja finalidade é identificar apenas as amostras com valores únicos (ou seja, não repetidos) no conjunto de dados. Nesse momento não foram identificados nenhuma amostra duplicada e por este motivo mantivemos o conjunto de dados inicial.

Para fazer a seleção dos atributos, primeiro buscamos identificar aqueles em que havia muitos valores faltantes, pois esta variável pouco contribuiria para a análise. O atributo V2, por exemplo, apresenta 41 valores NAs (20% do total de observações), e por isso decidimos eliminá-lo. Em seguida, verificamos quantas observações possuiam NAs em qualquer um dos campos, o que nos levou a um total de 10 observações. Como essa quantia é pequena, em relação ao total (4,78%), decidimos substituir esses valores pelas respectivas médias dos atributos.

A etapa seguinte de seleção dos atributos consistia na identificação e remoção dos atributos redundantes, visto que isso permite reduzir a dimensionalidade do conjunto de dados e economizar processamento. Primeiramente transformamos todas as variáveis categóricas em variáveis numéricas pelo método one hot enconding, o que permitiu a utilização das mesmas no cálculo da matriz de correlação. A matriz de correlação tem o objetivo de identificar quais dos atributos são mais ou menos correlacionados com os demais, sendo, portanto, uma boa medida da redundância. Decidimos retirar do nosso conjunto de dados todos os atributos cuja correlação seja de no mínimo 50% em relação a um ou mais dos outros atributos. Esse corte permitiu que o número de atributos do nosso conjunto de dados fosse reduzido de 54 para 29.

Decidimos fazer a transformação de todas as variáveis não numéricas - ao invés de transformar apenas aquelas que julgávamos necessárias manter - para poder incluí-las na matriz de correlação e só então, a partir dessa matriz, termos um referencial de qual manteríamos ou não. Observando atentamente os atributos indicados como altamente correlacionados, notamos como alguns até poderiam ter sido eliminados desde o início, como é o caso do atributo v4 (fuel_type) e v18 (fuel_system), pois o segundo atributo é suficiente para obter informações a respeito do primeiro (correlação igual a 1). De qualquer forma, nossa decisão em fazer uma análise criteriosa da correlação, antes de remover qualquer atributo se deve ao fato de não termos conhecimento inicial sobre o universo analisado.

Por fim, decidimos checar novamente se após a redução dos atributos havia amostras repetidas e desta vez tivemos um resultado positivo: 55 amostras eram repetidas. Retirando-as, nosso conjunto de dados final ficou com 150 observações e 29 atributos.

Atividade 2 – Agrupamento com o *K-means*

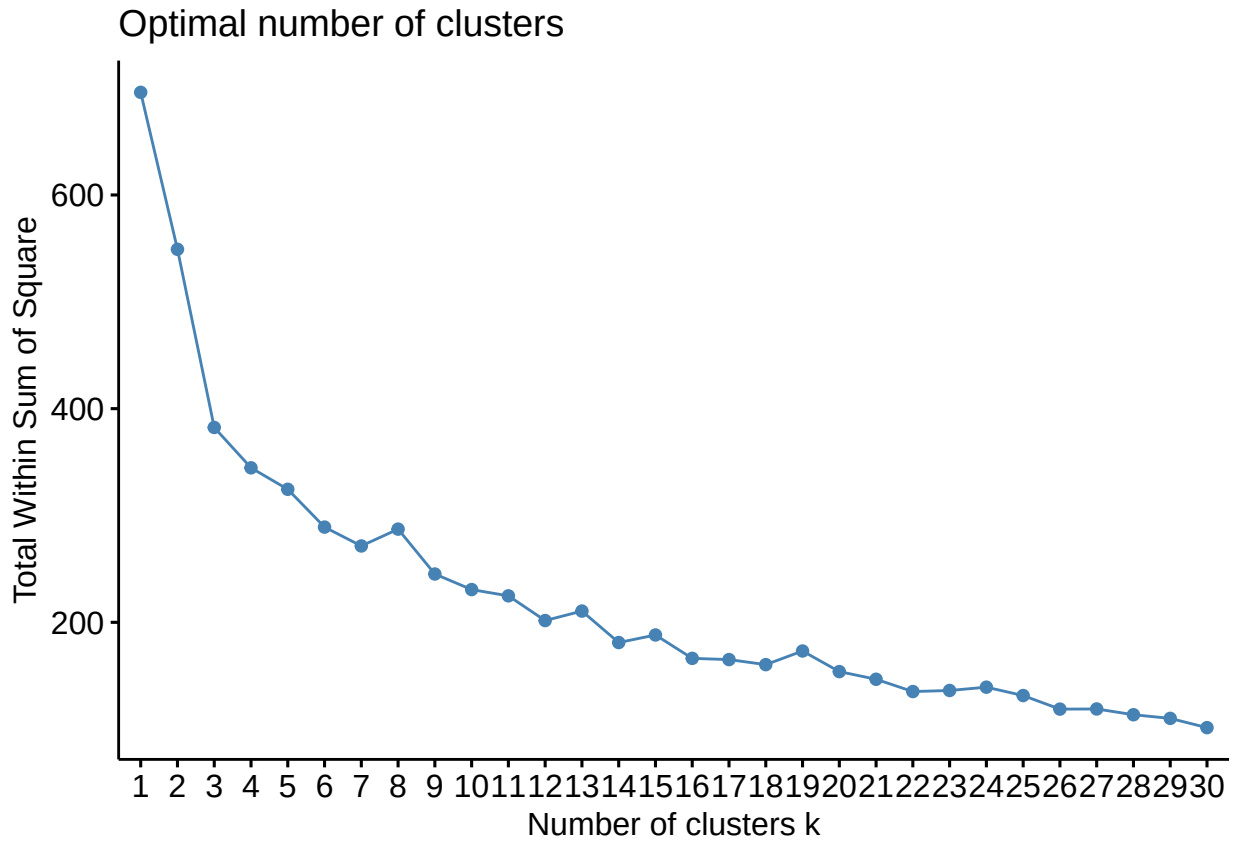
Nesta atividade, você deverá agrupar os dados com o algoritmo *K-means* e utilizará duas métricas básicas para a escolha do melhor K: a soma de distâncias intra-cluster e o coeficiente de silhueta.

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.

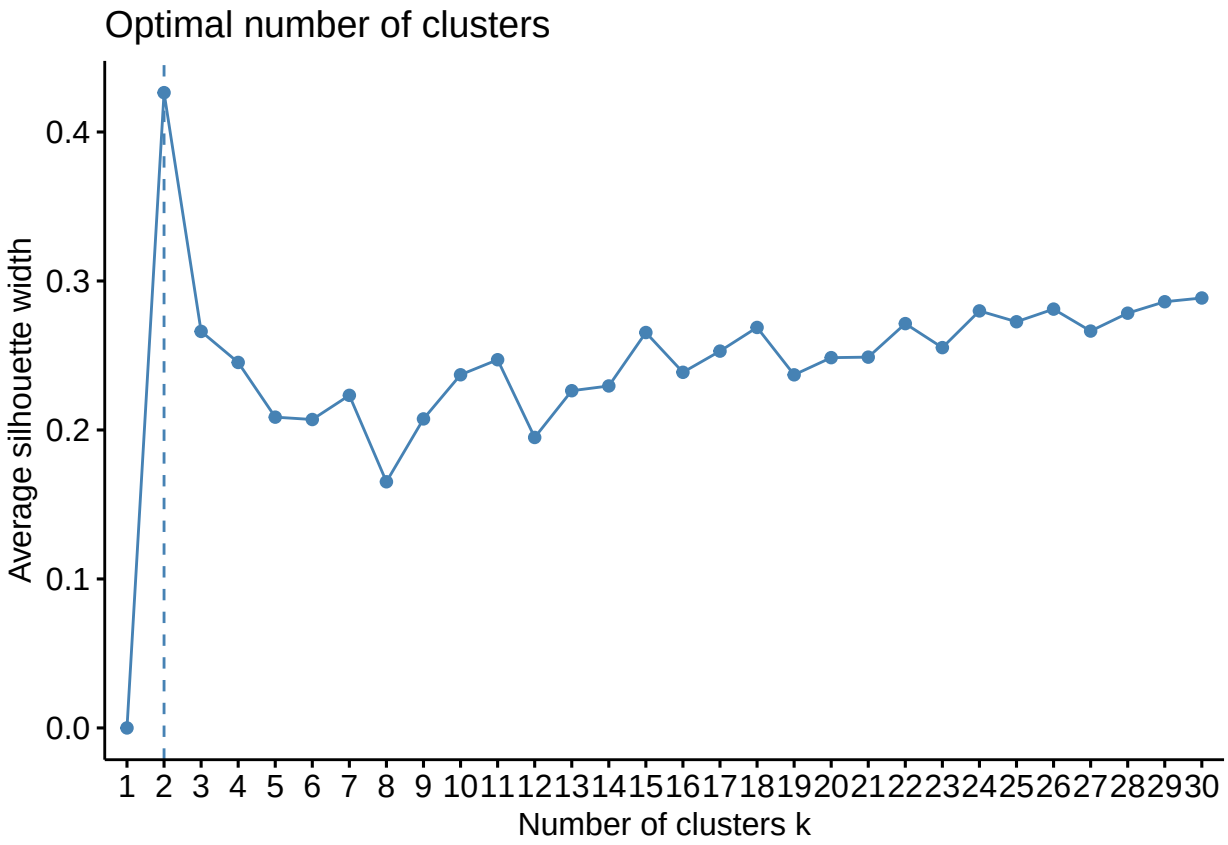
```
# Construindo um gráfico com as distâncias intra-cluster
```

```r
# Selecionando somente os atributos de interesse
carros_scaled <- carros[,3:length(carros)]
# Normalizando os atributos numéricos (os 3 primeiros, já que os outros são one-hot-encoding)
carros_scaled[,1:3] <- scale(carros_scaled[,1:3])
df <- carros_scaled
df <- na.omit(df)

# Determina o numero de grupos
set.seed(123)
fviz_nbclust(df, kmeans, method="wss", k.max = 30)
```



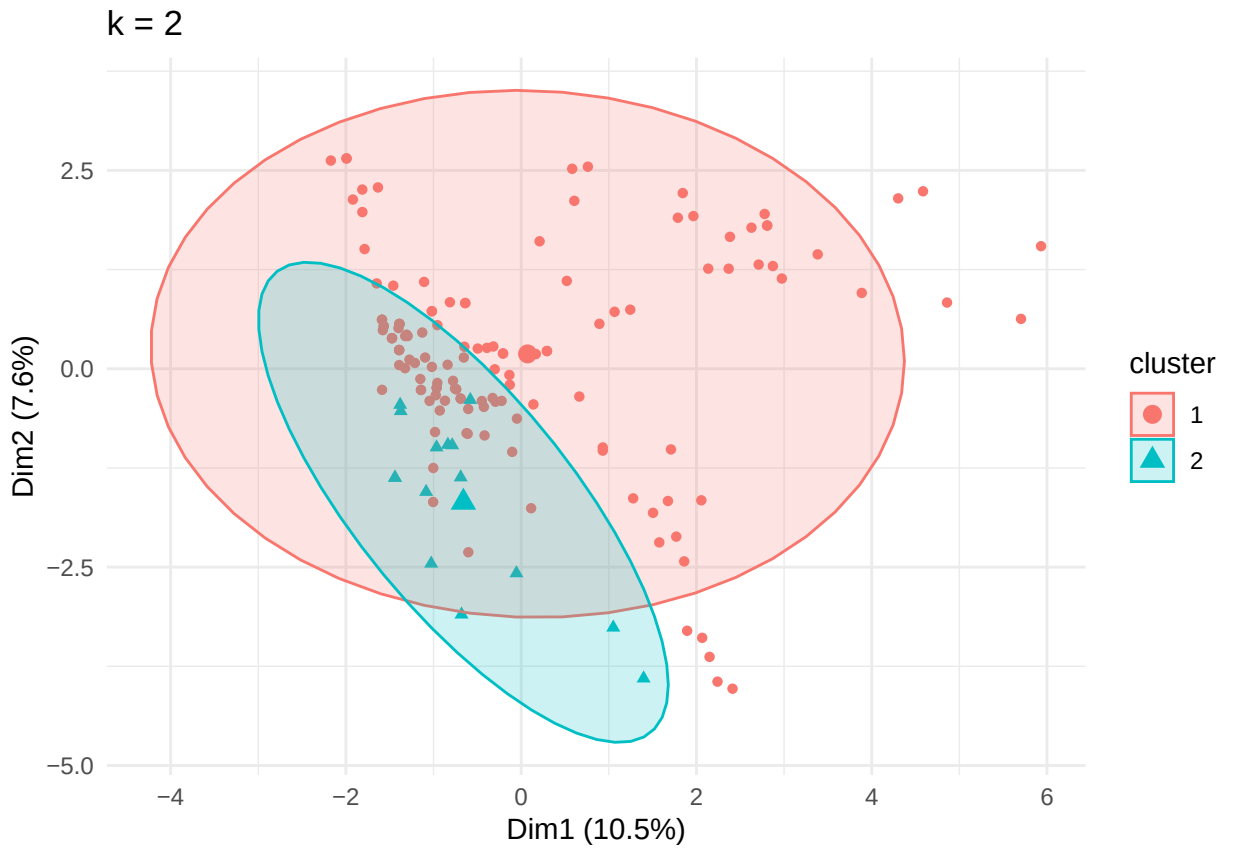
b) *Gráfico da Silhueta:* Construa um gráfico com o valor da silhueta para K variando de 2 a 30.

```r
# Construindo um gráfico com os valores da silhueta

# Determina o numero de grupos (medida de silhueta)
fviz_nbclust(df, 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).

```
# k escolhido: 2
# Justificativa: Pelo gráfico Elbow Curve, não é possível determinar um k ótimo, mas
# nos mostra vários candidatos, como 2, 3, 6, 7, etc. Já o gráfico de silhueta nos dá como
# o melhor k o 2.

# Aplicando o k-means com o k escolhido
km.res <- eclust(carros_scaled, "kmeans", k=2, nstart=25, graph=FALSE)

# Construindo um gráfico de dispersão

  # versao 1
fviz_cluster(km.res, geom="point", ellipse.type="norm") +
  ggtitle("k = 2") + theme_minimal()
```




```
  # versao 2: tentativa de saber quem eram os carros
#fviz_cluster(km.res, geom="point", data =carros_scaled, shape = 16) +
#  geom_point(aes(shape = as.factor(carros$V3)), alpha = 0.5) +
#  ggtitle("k = 2") + theme_minimal()

  # versao 3: tentativa de saber quem eram os symboling
#fviz_cluster(km.res, geom="point", data =carros_scaled, shape = 16) +
#  geom_point(aes(shape = as.factor(carros$V1)), alpha = 0.5) +
#  ggtitle("k = 2") + theme_minimal()

#fviz_cluster(km.res, geom="point", ellipse.type="norm", data =carros_scaled, shape = 16) +
#  geom_point(aes(shape = as.factor(carros$V1)), alpha = 0.5) +
#  ggtitle("k = 2") + theme_minimal()
```

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:

Analisando o gráfico Elbow Curve, não é possível determinar um k ótimo, porém ele nos mostra vários candidatos, como 2, 3, 6, 7, etc. Por outro lado, ao analisar o gráfico de silhueta, o próprio desenho já nos dá k=2 como o melhor. No ponto em onde k=2, a silhueta apresentou maior valor do que nos demais k testados.

Após decidirmos por k=2, geramos o gráfico de dispersão onde são mostrados os clusters gerados, vermelho e azul. Pelo gráfico, verificamos que o cluster vermelho agrupou bem mais exemplos que o azul.

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.

```
dados <- carros_scaled
set.seed(123)

# Executa algoritmo de agrupamentos DBSCAN

# Experimento com valores de eps e minPts

#densidade=1.0, obtemos muitos outliers (noise points)
dbscan::dbscan(dados, eps=1.0, minPts=2)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1, minPts = 2
## The clustering contains 19 cluster(s) and 48 noise points.
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
## 48  9  3  2  2 35  7  7  7  3  2  5  6  2  2  2  2  2  2  2
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=1.0, minPts=3)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1, minPts = 3
## The clustering contains 9 cluster(s) and 68 noise points.
##
##  0  1  2  3  4  5  6  7  8  9
## 68  9  3 35  7  7  7  3  5  6
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=1.0, minPts=5)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1, minPts = 5
## The clustering contains 4 cluster(s) and 99 noise points.
##
##  0  1  2  3  4
## 99 34  5  5  7
##
## Available fields: cluster, eps, minPts
```

```r
dbscan::dbscan(dados, eps=1.0, minPts=8)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1, minPts = 8
## The clustering contains 1 cluster(s) and 118 noise points.
##
##   0   1
## 118  32
##
## Available fields: cluster, eps, minPts
```

```r
#densidade=1.5, obtemos menos outliers,
dbscan::dbscan(dados, eps=1.5, minPts=2)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1.5, minPts = 2
## The clustering contains 10 cluster(s) and 9 noise points.
##
##   0   1   2   3   4   5   6   7   8   9  10
##   9 109   4  10   3   4   2   2   2   2   3
##
## Available fields: cluster, eps, minPts
```

```r
dbscan::dbscan(dados, eps=1.5, minPts=3)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1.5, minPts = 3
## The clustering contains 6 cluster(s) and 17 noise points.
##
##   0   1   2   3   4   5   6
##  17 109   4  10   3   4   3
##
## Available fields: cluster, eps, minPts
```

```r
dbscan::dbscan(dados, eps=1.5, minPts=5)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1.5, minPts = 5
## The clustering contains 3 cluster(s) and 32 noise points.
##
##  0  1  2  3
## 32 97 10 11
##
## Available fields: cluster, eps, minPts
```

```r
dbscan::dbscan(dados, eps=1.5, minPts=8)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1.5, minPts = 8
## The clustering contains 3 cluster(s) and 41 noise points.
```

```
##
##  0  1  2  3
## 41 92  8  9
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=1.5, minPts=13)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1.5, minPts = 13
## The clustering contains 1 cluster(s) and 63 noise points.
##
##  0  1
## 63 87
##
## Available fields: cluster, eps, minPts
```

```
#densidade=2.0, começa a englobar conforme aumenta minPts, mas ainda com outliers
dbscan::dbscan(dados, eps=2.0, minPts=2)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2, minPts = 2
## The clustering contains 3 cluster(s) and 0 noise points.
##
##   1   2   3
## 133  15   2
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.0, minPts=3)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2, minPts = 3
## The clustering contains 2 cluster(s) and 2 noise points.
##
##   0   1   2
##   2 133  15
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.0, minPts=5)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2, minPts = 5
## The clustering contains 2 cluster(s) and 4 noise points.
##
##   0   1   2
##   4 132  14
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.0, minPts=8)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2, minPts = 8
## The clustering contains 2 cluster(s) and 5 noise points.
##
##   0   1   2
##   5 131  14
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.0, minPts=13)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2, minPts = 13
## The clustering contains 1 cluster(s) and 21 noise points.
##
##   0   1
##  21 129
##
## Available fields: cluster, eps, minPts
```

```
  #minPts=3, 5 ou 8, ficam poucos de fora

#abaixo densidade para 2.5, engloba todos e tem k=2 para qualquer minPts
dbscan::dbscan(dados, eps=2.5, minPts=2)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2.5, minPts = 2
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
## 135  15
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.5, minPts=3)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2.5, minPts = 3
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
## 135  15
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.5, minPts=5)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2.5, minPts = 5
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
## 135  15
##
## Available fields: cluster, eps, minPts
```

```
dbscan::dbscan(dados, eps=2.5, minPts=8)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2.5, minPts = 8
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
## 135  15
##
## Available fields: cluster, eps, minPts
```

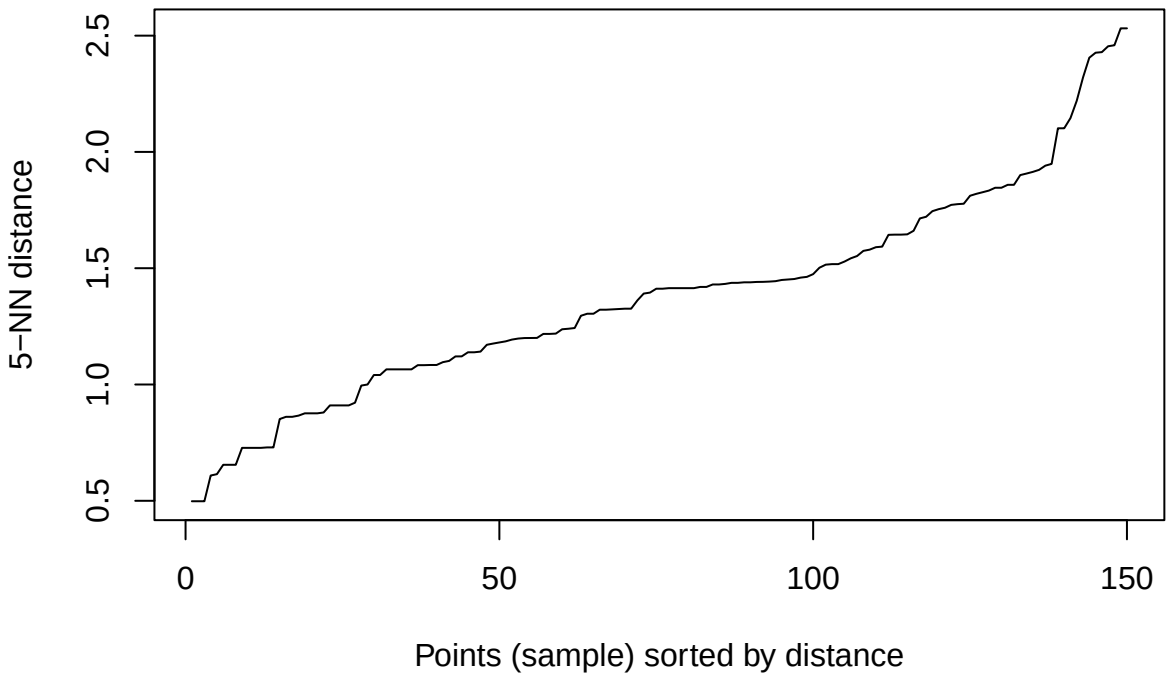
```
dbscan::dbscan(dados, eps=2.5, minPts=13)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 2.5, minPts = 13
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
## 135  15
##
## Available fields: cluster, eps, minPts
```

```
# CONCLUSAO: minPts com 3, 5 ou 8 parecem bons => decidimos por minPts=5

#
```

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.

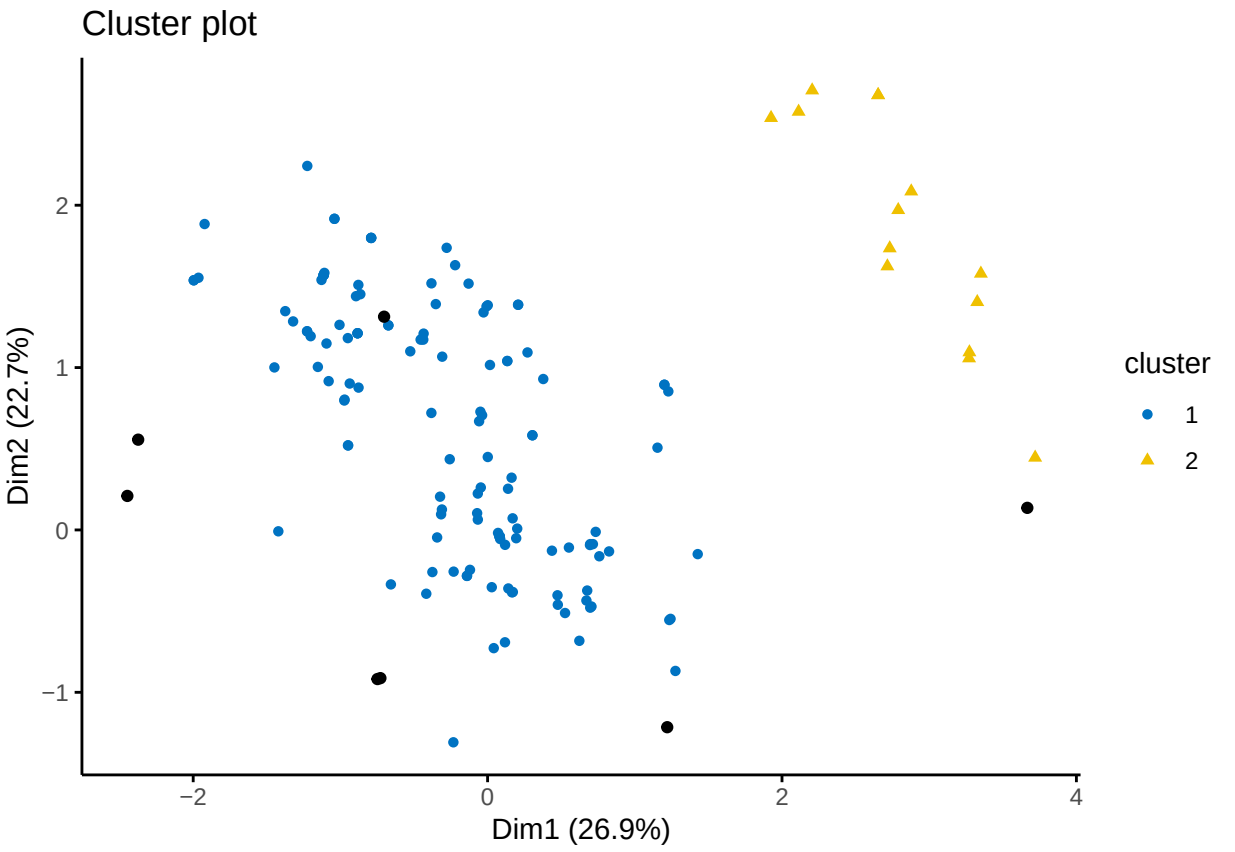
```
# Encontrando o melhor eps com o kNNdistplot
dbscan::kNNdistplot(dados, k = 5)
```




```
#CONCLUSAO: eps=1.9, analisando o grafico pelo "cotovelo"
```

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 o DBscan com os parâmetros escolhidos
db <- dbscan::dbscan(dados, eps=1.9, minPts=5)

# Construindo um gráfico de dispersão
fviz_cluster(db, data=dados, 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:

Para experimentar várias densidades (eps) e minPts, fixamos inicialmente o eps=1.0 e testamos alguns minPts. Depois, aumentamos o eps em 0.5 e testamos novamente para os mesmos minPts. Fizemos isso até o resultado não mais variar o agrupamento.

Para densidade=1.0, obtemos muitos outliers (noise points).

Aumentando para 1.5, obtemos menos outliers, ainda com muito outliers.

Com 2.0, começou a englobar conforme aumenta minPts. Com minPts=3, 5 ou 8, poucos exemplos ficaram de fora.

Com densidade igual em 2.5, engloba todos os exemplos e gerou 2 clusters para qualquer minPts.

Após a experimentação, principalmente para o eps=2.0, minPts com 3, 5 e 8 parecem bons. Então, decidimos por minPts=5.

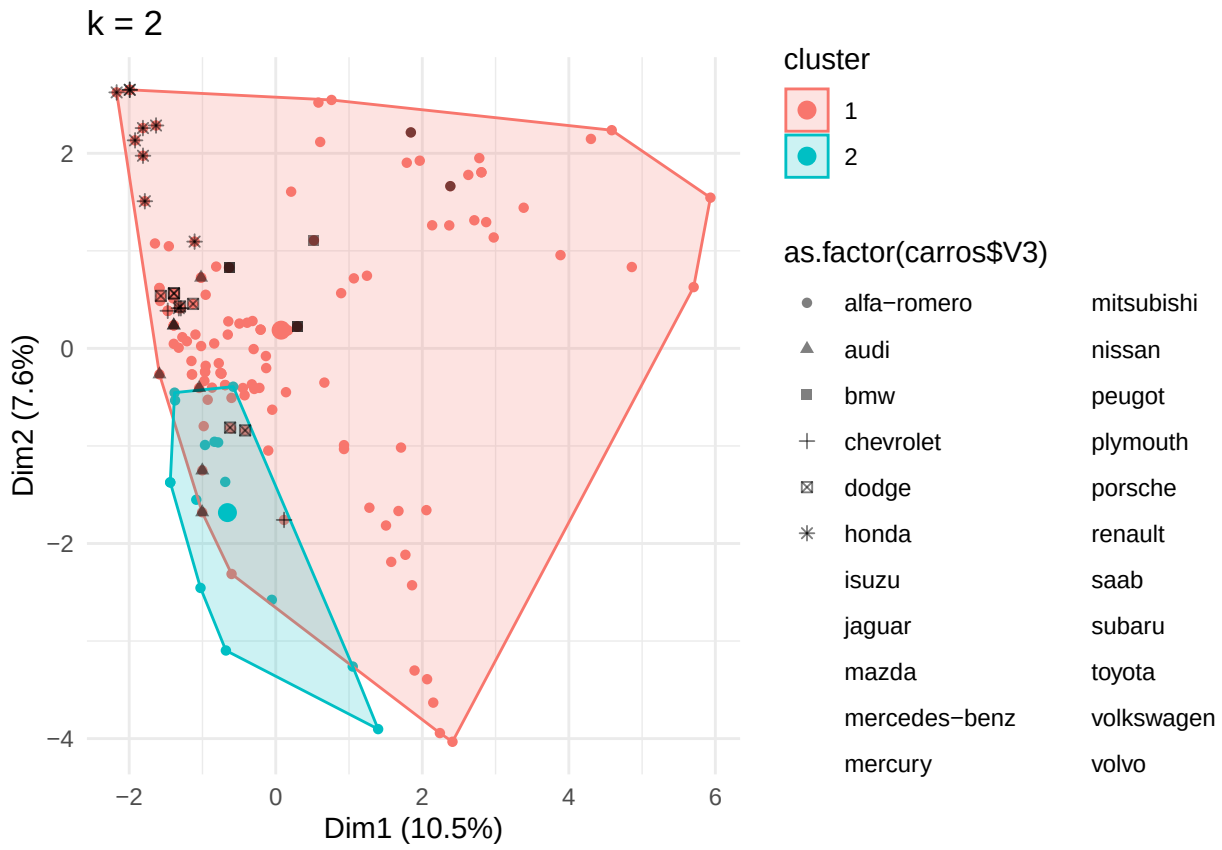
Após decidirmos pelo minPts=5, executamos o kNNdistplot para escolher o melhor eps. Analisando o gráfico pelo "cotovelo", verificamos que em eps=1.9 o gráfico dá um salto maior. Decidimos por eps=1.9.

Após decidirmos por minPts=5 e "calcularmos" eps=1.9, geramos um gráfico de dispersão, onde os 2 clusters foram mostrados. Pelo gráfico, os clusters estão bem separados. Na imagem 2D, nenhum ponto de um cluster

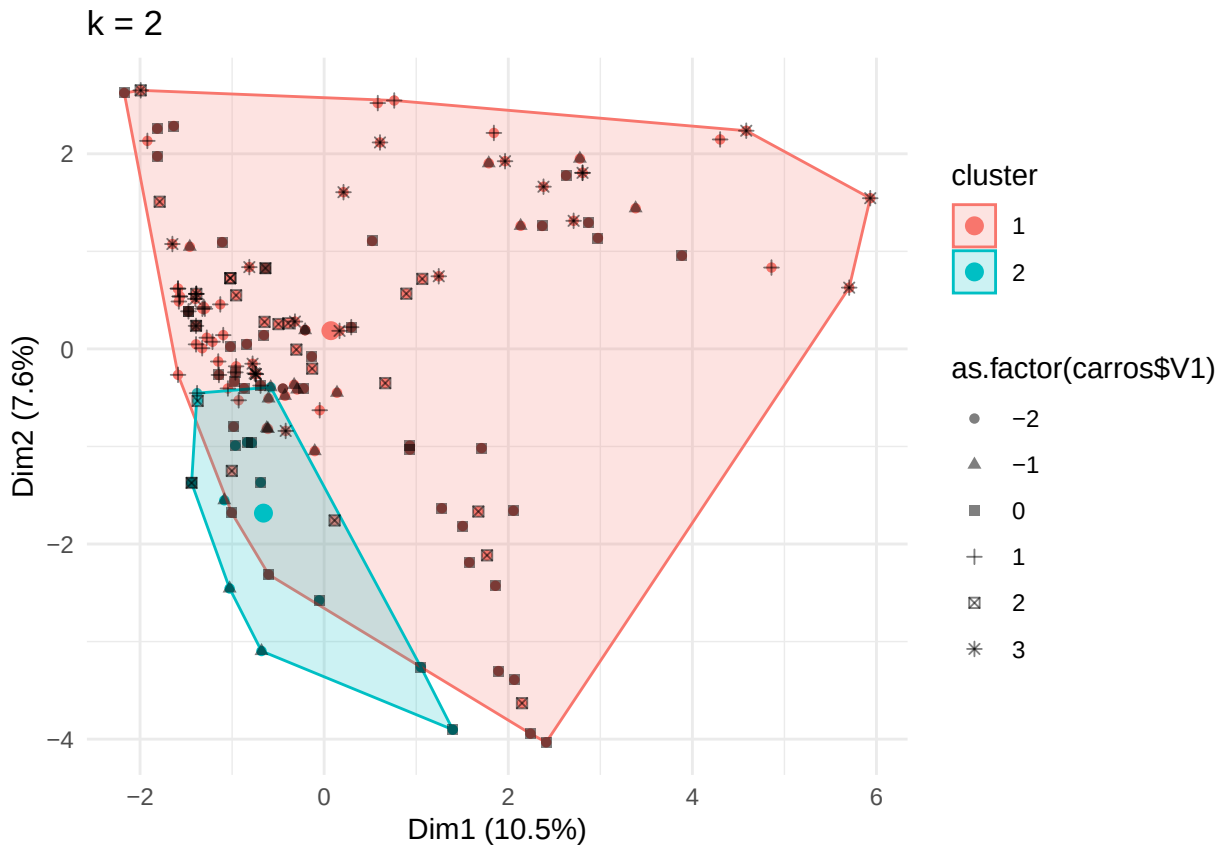
ficou dentro de outro, ou seja, os 2 clusters estão bem definidos. O resultado ainda apresentou alguns outliers, 7 exemplos em preto.

Atividade 4 – Comparando os Algoritmos

```
# k-means: visualizando "make" nos agrupamentos (nao ficou bom)
fviz_cluster(km.res, geom="point", data =carros_scaled, shape = 16) +
  geom_point(aes(shape = as.factor(carros$V3)), alpha = 0.5) +
  ggtitle("k = 2") + theme_minimal()
```




```
# k-means: Visualizando "symboling" nos agrupamentos (ficou melhor)
fviz_cluster(km.res, geom="point", data =carros_scaled, shape = 16) +
  geom_point(aes(shape = as.factor(carros$V1)), alpha = 0.5) +
  ggtitle("k = 2") + theme_minimal()
```




```
# Montando um data.frame com os campos de interesse (symboling e make)
# e os campos dos clusters do k-means e DBSCAN
df <- data.frame(symboling  = carros$V1,
                 make       = carros$V3,
                 km_cluster = km.res$cluster,
                 db_cluster = db$cluster)

# Comparando: symboling VS km_cluster
table(df[,c(1,3)])
```

```
##          km_cluster
## symboling  1  2
##        -2  2  0
##        -1 14  4
##        0  41  8
##        1  37  1
##        2  21  2
##        3  20  0
```

```
# Comparando: symboling VS db_cluster
table(df[,c(1,4)])
```

```
##          db_cluster
## symboling  0  1  2
##        -2  0  2  0
```

```
##        -1  0 14  4
##        0   1 41  7
##        1   4 33  1
##        2   0 21  2
##        3   2 18  0
```

```
# Comparando: make VS km_cluster
table(df[,c(2,3)])
```

```
##                km_cluster
## make             1  2
##   alfa-romero    2  0
##   audi           7  0
##   bmw            5  0
##   chevrolet      3  0
##   dodge          7  0
##   honda         10  0
##   isuzu          4  0
##   jaguar         2  0
##   mazda          8  2
##   mercedes-benz  4  3
##   mercury        1  0
##   mitsubishi     9  0
##   nissan        11  1
##   peugot         3  2
##   plymouth       5  0
##   porsche        4  0
##   renault        2  0
##   saab           5  0
##   subaru        12  0
##   toyota        17  3
##   volkswagen     6  3
##   volvo          8  1
```

```
# Comparando: make VS db_cluster
table(df[,c(2,4)])
```

```
##                db_cluster
## make             0  1  2
##   alfa-romero    1  1  0
##   audi           0  7  0
##   bmw            0  5  0
##   chevrolet      0  3  0
##   dodge          0  7  0
##   honda          0 10  0
##   isuzu          0  4  0
##   jaguar         0  2  0
##   mazda          0  8  2
##   mercedes-benz  1  3  3
##   mercury        0  1  0
##   mitsubishi     0  9  0
##   nissan         0 11  1
##   peugot         1  3  1
```

```
##   plymouth       0  5  0
##   porsche        2  2  0
##   renault        0  2  0
##   saab           0  5  0
##   subaru         0 12  0
##   toyota         2 15  3
##   volkswagen     0  6  3
##   volvo          0  8  1
```

```
# Cluster do km
table(df[,c(3)])
```

```
##
##   1   2
## 135  15
```

```
# Cluster do db
table(df[,c(4)])
```

```
##
##   0   1   2
##   7 129  14
```

```
# Exemplos que estão em clusters diferentes
df[df$km_cluster != df$db_cluster,]
```

```
##     symboling          make km_cluster db_cluster
## 3           1   alfa-romero          1          0
## 75          1 mercedes-benz          1          0
## 111         0        peugot          2          0
## 127         3       porsche          1          0
## 129         3       porsche          1          0
## 166         1        toyota          1          0
## 167         1        toyota          1          0
```

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:

Ao realizar uma análise dos dois gráficos de cluster gerados, do K-means e do DBSCAN, ainda que analisando em 2D, o DBScan parece ter agrupado melhor os exemplos, pois visualmente estão bem separados.

Em termos de cluster, ambos os métodos retornaram 2 agrupamentos. Aliás, as respostas estão, de certa forma, muito parecidas: enquanto o K-means retornou um cluster com 135 exemplos e outro com 15, o DBSCAN retornou 129 e 14 (além dos 7 considerados como outliers). E, olhando melhor e fazendo uma busca por agrupamentos diferentes, somente os outliers do DBSCAN não batem com o do K-means. Se voltássemos atrás e decidíssemos por um eps=2.5 (mantendo o minPts=5), não teríamos outliers no DBSCAN e todos os agrupamentos seriam os mesmos. Fizemos esse teste, mas mantemos os valores experimentados.

Para analisar o `symboling`, tentamos primeiramente plotar o k-means, mas não ficou muito visível. Depois fizemos comparações relacionando k-means e DBSCAN com `symboling` (código e tabela acima). Pelas tabelas geradas, não há uma separação em termos de risco. A única separação notada é que, se um exemplo está no grupo 2, este não está com `symboling` extremos, -2 ou 3.

Assim como fizemos com o symboling, tentamos plotar o campo `make`, mas ficou menos visível que o anterior. Ao fazer a tabela comparativa com o make (código e tabela acima), a única separação notada é que se um exemplo está no grupo 2, este pertence a uma dessas marcas: mazda, mercedes-benz, nissan, peugot, toyota, volkswagen ou volvo. As outras marcas possuem exemplos nos 2 clusters.