Data Driven Decision-Making in Business

Week 3

Meike Morren

You have encountered several clustering techniques, distance measures, linkage methods and methods to determine the optimal number of clusters. In this week you will integrate all your knowledge.

You will use the same data set as last week. First read in the datafile `supermarket-sales.csv` which you can find on canvas. For more information, visit: kaggle. Explore the datafile. In particular, observe how many observations, what is the unit of observation, and how many metric variables are included in the datafile.

```
df <- read.csv("supermarket-sales.csv")
colnames(df)
```

```
##  [1] "Invoice.ID"              "Branch"
##  [3] "City"                    "Customer.type"
##  [5] "Gender"                  "Product.line"
##  [7] "Unit.price"              "Quantity"
##  [9] "Tax.5."                  "Total"
## [11] "Date"                    "Time"
## [13] "Payment"                 "cogs"
## [15] "gross.margin.percentage" "gross.income"
## [17] "Rating"
```

K-modes

In contrast to the other methods, K-modes is only to be performed on categorical data. You do not have to recode the variables to numeric like we did when calculating the Manhattan distances. K-modes only takes the modes (i.e. the most frequent values) which can also be performed on character data. Perform the analysis on the categorical variables available in the dataset:

- Branch
- City
- Customer.type
- Product.line
- Payment

Use the function `kmodes` with `iter.max = 100` for k is 2, 3 or 4 (use the argument `modes =`). Include the whole dataset! Create a `barplot` to visualize how many data points are assigned to which cluster. Inspect the three different barplots and explain what you see, is the distribution even for all tried k-values?

```
catdata <- na.omit(df[, c("Branch", "City", "Customer.type", "Product.line", "Payment")])
kmodes(catdata, modes = 2,iter.max = 100)
```

```
## K-modes clustering with 2 clusters of sizes 599, 401
##
## Cluster modes:
##   Branch      City Customer.type           Product.line Payment
## 1      A    Yangon        Member     Home and lifestyle Ewallet
## 2      C Naypyitaw        Normal Electronic accessories    Cash
##
## Clustering vector:
##    [1] 1 2 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 2 1 2 1 1 1 1
##   [38] 1 2 1 1 2 1 2 2 2 1 1 1 1 2 1 1 2 1 2 1 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 1 2
##   [75] 1 2 2 1 1 2 2 1 2 2 2 2 2 1 1 2 2 2 1 1 2 1 1 2 1 2 2 2 2 1 2 1 2 1 2 2 1
##  [112] 2 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 2 1 2 2 1 1 2 1 2 2 2 2 1 2 1 2
##  [149] 1 1 1 2 1 2 2 1 2 1 1 1 2 1 1 2 2 1 2 1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 1 2 1
##  [186] 1 1 1 1 2 1 1 2 1 1 2 2 1 2 2 2 2 2 1 1 1 2 1 1 2 1 2 1 2 1 1 2 1 1 2 2 1
##  [223] 2 2 1 2 1 2 2 1 2 1 2 1 1 1 2 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 2 1 1 1 1 1 1
##  [260] 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 2 1 1 2 1 1 1 2 2 2 2 1 1 2 1 1 1 2
##  [297] 2 1 1 2 2 1 2 1 2 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 2 2 1 1 1 1 1 2 1 1 2 1 1
##  [334] 1 1 1 1 2 2 1 1 1 1 2 1 1 1 2 2 2 2 2 1 1 2 1 2 2 2 2 1 2 2 1 2 2 2 1 2 1
##  [371] 2 2 2 2 1 1 1 2 1 2 1 2 2 1 1 1 2 1 1 2 2 2 1 1 1 1 1 2 1 2 2 2 1 1 1 1 1
##  [408] 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 2 1 2 1 1 1 2 1 2 2 1 2 2 2 1 1 1 2 2 1 1 2
##  [445] 1 1 2 2 1 1 2 2 1 1 1 1 1 2 2 2 2 1 2 2 1 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 2
##  [482] 2 1 1 2 1 2 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 2 2 1 1 1
##  [519] 1 2 2 2 1 2 1 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 1 2 2 1 1 1 2 2 1 2 2 2 1
##  [556] 1 1 1 1 1 2 2 2 1 2 1 2 1 2 2 1 1 1 2 1 1 2 2 1 1 2 1 2 1 1 1 1 1 2 1 2 1
##  [593] 1 1 1 2 1 2 2 1 2 2 2 1 2 2 1 2 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1
##  [630] 1 1 1 1 1 1 1 1 2 1 1 1 2 1 2 2 1 2 1 2 2 2 2 1 1 1 1 2 1 1 1 2 2 1 2 2 1
##  [667] 1 1 1 2 1 1 1 2 1 1 1 1 2 1 2 2 2 1 1 1 1 1 2 1 2 1 1 2 2 1 1 1 1 2 2 2 1
##  [704] 1 1 2 2 2 2 1 1 1 2 2 1 1 1 1 1 1 2 2 1 2 1 2 1 2 2 1 1 1 1 2 1 2 1 2 1 1
##  [741] 2 2 1 1 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 1 1 2 1 2 2 2 2 2
##  [778] 2 2 1 2 1 1 2 1 1 2 2 2 1 1 2 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 2 2 2 1 1 2 2
##  [815] 1 1 2 1 1 1 2 1 2 1 1 1 1 1 2 1 2 2 1 1 1 1 1 2 2 1 1 2 1 2 1 1 1 2 2 1 1
##  [852] 1 2 2 1 2 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 2 2 1 1 2 2 2 1 1 1 2 1 1 1 1 1 1
##  [889] 2 1 2 2 2 2 1 2 2 2 2 1 2 1 1 1 2 2 2 2 1 1 1 2 1 1 1 2 2 1 1 1 1 2 1 2 1
##  [926] 2 1 1 2 1 1 2 1 2 1 1 2 1 1 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 2 2 1 2 1
##  [963] 1 2 2 2 1 1 1 1 1 1 2 1 2 1 1 1 2 2 2 1 1 2 2 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1
## [1000] 1
##
## Within cluster simple-matching distance by cluster:
## [1] 1559  878
##
## Available components:
## [1] "cluster"    "size"       "modes"      "withindiff" "iterations"
## [6] "weighted"
```

```r
kmodes(catdata, modes = 3,iter.max = 100)
```

```
## K-modes clustering with 3 clusters of sizes 400, 350, 250
##
## Cluster modes:
##   Branch      City Customer.type           Product.line     Payment
## 1      B  Mandalay        Normal Electronic accessories        Cash
## 2      A    Yangon        Normal     Home and lifestyle     Ewallet
## 3      C Naypyitaw        Member    Fashion accessories Credit card
##
```

```
## Clustering vector:
##    [1] 2 1 2 2 2 1 2 2 2 1 1 1 2 2 2 1 2 2 2 1 3 1 1 2 2 2 1 2 1 2 1 1 1 2 3 3 2
##   [38] 2 2 1 1 3 1 3 3 1 1 1 1 3 3 2 1 3 1 1 2 2 2 1 3 2 1 2 1 2 2 1 2 2 3 3 1 1
##   [75] 2 3 3 3 3 1 3 1 2 3 3 2 3 2 2 1 3 2 2 1 3 2 1 1 2 1 3 3 1 2 1 2 3 2 1 3 1
##  [112] 3 1 2 3 3 1 1 2 1 2 3 1 3 1 2 2 3 3 1 1 2 1 1 2 3 2 2 1 2 3 3 3 3 2 3 2 3
##  [149] 1 2 3 3 2 2 1 2 1 1 1 1 3 2 2 2 1 1 1 2 2 2 2 1 1 1 1 2 2 3 2 3 3 3 2 2 2
##  [186] 1 1 1 2 3 1 1 1 1 2 3 3 2 3 3 3 1 1 1 1 2 3 3 1 1 2 1 1 1 1 2 1 1 1 1 1 1
##  [223] 1 3 2 3 1 3 1 2 1 1 1 1 2 2 3 3 1 2 2 2 3 2 1 2 1 2 2 1 1 3 3 2 2 1 2 2 2
##  [260] 3 2 3 1 2 1 2 3 1 2 2 1 3 2 2 1 1 3 3 3 2 2 3 2 2 2 1 3 1 1 2 1 1 1 2 1 3
##  [297] 1 2 2 3 3 1 1 2 1 2 2 2 2 2 1 3 2 2 3 3 3 3 3 3 1 1 2 2 2 1 2 3 1 1 1 2 2
##  [334] 2 3 2 2 1 3 1 1 1 1 2 2 2 1 3 1 1 3 1 3 1 1 1 3 1 1 1 2 1 2 2 1 3 1 2 1 2
##  [371] 1 1 2 3 2 2 1 2 3 1 2 1 1 3 2 1 1 2 3 1 3 1 2 2 2 2 2 1 1 1 3 3 3 1 3 2 2
##  [408] 1 2 3 1 1 2 2 2 1 3 3 1 1 3 1 3 3 3 1 2 1 1 2 1 2 2 1 1 2 3 2 3 3 3 1 2 3
##  [445] 2 1 3 3 1 1 1 1 2 2 1 1 1 1 3 3 1 1 1 3 2 3 3 1 2 3 3 2 3 1 1 2 2 1 1 2 1
##  [482] 1 2 1 3 1 1 2 1 1 1 3 1 3 1 1 1 1 1 2 1 3 2 1 1 2 1 1 1 3 1 2 2 1 3 3 1 3
##  [519] 2 3 1 3 2 1 2 2 1 1 1 2 2 3 1 2 2 2 3 2 2 3 2 3 1 3 1 1 2 2 1 1 1 1 1 3 1
##  [556] 1 1 3 2 2 1 3 1 2 1 2 3 2 1 2 1 1 2 1 2 1 1 2 2 1 1 2 3 3 1 2 2 2 3 2 3 3
##  [593] 2 2 1 1 2 2 3 2 1 3 2 1 3 1 2 3 2 1 2 3 1 3 2 2 1 3 2 3 2 2 1 1 1 1 2 1 2
##  [630] 2 2 2 2 1 1 1 2 1 1 1 1 3 1 3 3 2 1 1 3 1 1 1 2 1 3 2 3 2 2 3 1 3 3 3 1 2
##  [667] 1 1 3 1 2 1 1 3 2 1 1 2 2 2 1 1 3 2 1 1 1 2 3 2 3 3 2 3 1 2 2 1 2 2 3 1 3
##  [704] 1 1 1 1 3 3 2 2 3 1 2 3 2 2 2 2 3 1 3 1 3 1 3 3 1 3 1 3 2 2 1 1 3 3 1 1 2
##  [741] 1 2 2 2 3 3 3 3 1 3 1 2 2 1 3 2 1 2 2 2 1 1 2 2 2 1 3 1 1 2 1 3 3 3 3 1 3
##  [778] 1 3 1 1 2 3 1 3 2 1 3 3 2 2 3 1 2 2 1 3 2 1 3 1 3 3 2 1 2 2 1 1 3 1 2 3 1
##  [815] 1 1 3 2 1 1 1 2 3 2 1 2 1 3 1 2 1 1 1 2 1 2 2 1 1 3 2 1 2 3 2 2 2 3 3 2 2
##  [852] 2 2 1 2 1 1 2 1 2 3 2 1 1 1 3 1 3 3 2 2 3 1 2 2 2 3 1 2 1 1 3 1 2 2 2 2 2
##  [889] 3 2 1 1 3 1 1 1 3 3 3 2 3 1 2 2 2 3 3 1 2 1 1 1 2 2 2 1 3 2 1 1 3 1 3 3 3
##  [926] 1 1 2 1 1 1 3 2 2 1 3 2 2 2 2 2 3 2 2 2 2 3 1 3 1 1 1 1 3 1 2 3 1 1 2 3 1
##  [963] 2 3 1 1 2 2 2 1 1 1 1 2 3 3 2 1 1 1 3 2 2 1 1 1 1 1 3 1 2 1 2 1 3 2 1 2 2
## [1000] 2
##
## Within cluster simple-matching distance by cluster:
## [1] 839 686 399
##
## Available components:
## [1] "cluster"    "size"       "modes"      "withindiff" "iterations"
## [6] "weighted"
```

```
kmodes(catdata, modes = 4,iter.max = 100)
```

```
## K-modes clustering with 4 clusters of sizes 271, 335, 132, 262
##
## Cluster modes:
##   Branch      City Customer.type            Product.line     Payment
## 1      B  Mandalay        Member Electronic accessories Credit card
## 2      C Naypyitaw        Member      Health and beauty        Cash
## 3      B  Mandalay        Normal      Health and beauty     Ewallet
## 4      A    Yangon        Normal     Home and lifestyle        Cash
##
## Clustering vector:
##    [1] 2 2 4 2 4 2 1 2 1 1 1 1 4 4 4 1 1 4 4 3 2 3 1 4 4 4 3 4 3 4 1 1 3 4 2 2 4
##   [38] 4 2 1 1 2 1 2 2 1 1 1 1 2 2 4 1 2 3 2 4 3 4 2 2 2 1 1 1 2 2 1 4 2 2 2 1 2
##   [75] 4 1 2 1 2 2 2 3 2 2 2 2 2 1 4 3 2 2 4 1 2 4 3 2 4 3 2 2 2 4 3 1 2 4 2 2 1
##  [112] 2 3 4 2 2 1 1 4 3 4 2 1 2 1 4 4 2 2 3 3 4 3 1 2 2 4 4 3 4 2 2 2 2 4 2 4 2
##  [149] 1 4 1 2 4 2 2 1 1 3 3 3 2 4 4 2 1 3 2 4 4 1 3 1 2 1 3 4 1 2 4 2 2 2 4 2 4
```

```
##  [186] 1 1 1 4 2 1 3 2 3 4 2 2 4 2 2 2 1 2 3 1 3 1 2 3 1 4 2 3 1 1 4 3 4 3 1 3 3
##  [223] 2 2 4 2 3 2 1 4 3 1 3 1 2 4 2 2 1 4 4 4 2 4 1 4 1 4 1 1 1 2 2 4 4 1 1 4 1
##  [260] 1 4 2 1 4 3 1 2 1 4 4 3 2 4 4 3 3 2 2 2 1 4 2 4 2 2 3 2 2 1 4 1 2 4 4 3 2
##  [297] 2 4 4 2 2 1 2 4 1 1 4 4 1 4 1 2 1 2 2 2 2 2 2 2 2 2 2 3 4 4 3 4 2 1 4 3 4 4
##  [334] 1 2 1 4 3 1 1 1 3 1 2 4 4 4 2 4 3 2 4 1 1 2 1 2 2 3 3 1 2 2 4 2 2 2 4 2 1
##  [371] 1 1 2 2 4 4 1 2 2 1 1 2 3 2 4 1 2 4 2 1 2 3 1 1 3 3 4 3 1 1 2 2 2 3 2 1 4
##  [408] 1 4 2 3 3 2 4 4 3 2 2 3 4 2 2 2 1 2 1 2 1 3 4 1 2 4 1 1 2 2 4 2 1 2 1 4 2
##  [445] 4 1 2 2 1 1 1 1 4 3 4 1 1 1 2 2 2 1 2 2 1 2 2 3 2 2 2 4 1 1 4 4 4 2 3 4 2
##  [482] 2 4 1 2 1 3 4 2 1 3 1 3 2 3 3 2 2 1 4 1 2 2 3 1 1 1 1 1 2 1 4 4 4 2 2 3 2
##  [519] 4 2 1 2 4 2 4 4 1 1 1 4 3 1 3 2 4 2 1 4 4 2 4 2 1 1 1 1 4 4 3 4 3 3 3 1 4
##  [556] 1 1 2 4 4 1 2 3 1 3 4 2 4 1 2 1 1 4 1 4 1 3 2 4 3 2 2 2 1 3 4 4 4 2 4 2 2
##  [593] 4 4 1 3 4 2 2 4 2 2 2 3 2 3 4 2 4 1 4 2 1 2 4 4 1 2 1 2 4 4 1 1 1 1 4 1 4
##  [630] 4 4 4 1 3 1 1 4 2 1 3 1 2 1 2 2 4 2 1 2 2 3 3 4 1 1 4 1 4 4 1 3 2 1 2 2 4
##  [667] 1 3 2 1 4 1 3 2 4 1 1 4 2 4 1 1 2 4 1 1 1 4 2 4 2 2 2 2 2 4 4 3 1 2 2 1 1
##  [704] 1 1 3 3 2 2 4 4 2 2 2 2 4 4 1 4 1 3 2 1 2 1 2 2 1 2 1 1 3 4 3 1 2 2 2 1 4
##  [741] 2 2 4 4 2 2 1 2 1 2 1 4 1 3 2 4 1 4 4 4 1 1 4 2 1 3 2 3 1 4 1 2 2 2 2 1 2
##  [778] 3 2 1 2 4 1 2 2 4 2 2 2 4 4 2 1 1 4 3 2 4 3 2 3 2 2 4 1 4 4 4 3 2 3 4 2 4
##  [815] 4 3 2 4 1 1 3 4 2 4 1 1 1 1 2 1 4 3 1 2 1 4 4 1 2 2 4 1 1 2 4 1 1 2 2 4 4
##  [852] 4 2 3 2 3 3 4 3 4 2 4 1 3 4 2 1 2 2 4 1 2 1 4 4 2 2 1 4 1 1 2 1 4 4 4 4 1
##  [889] 2 4 2 1 2 1 1 3 2 2 2 4 1 3 1 4 2 2 2 3 4 1 1 2 4 4 4 2 2 4 3 1 2 3 2 2 2
##  [926] 3 1 1 1 3 3 2 4 2 3 2 2 4 2 2 4 2 4 4 2 4 1 1 2 3 3 1 1 2 1 4 2 1 2 4 2 4
##  [963] 4 2 1 1 4 4 4 1 1 1 1 4 2 1 4 1 1 1 2 4 4 2 2 3 3 1 2 1 4 3 4 3 2 2 3 4 4
## [1000] 4
##
## Within cluster simple-matching distance by cluster:
## [1] 493 655 165 457
##
## Available components:
## [1] "cluster"    "size"       "modes"      "withindiff" "iterations"
## [6] "weighted"
```

K-mediods

Using `gross.income`, `Rating`, `Tax.5.`,`Unit.price`, you will explore cosine versus Euclidean distances using K-mediods. First you need to calculate the cosine distance matrix using your code from week 2. Save the cosine distance matrix. Include the whole dataset! Also, don't forget to standardize the variables!

```
df$ZGross.income <- scale(df$gross.income)
df$ZRating <- scale(df$Rating)
df$ZTax.5. <- scale(df$Tax.5.)
df$ZUnit.price <- scale(df$Unit.price)

dfz <- na.omit(df[,c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")])
Matrix <- t(as.matrix(dfz))
cosine_dist <- 1-crossprod(Matrix) /(sqrt(colSums(Matrix^2)%*%t(colSums(Matrix^2))))
cosine_dist <- as.dist(cosine_dist)
```

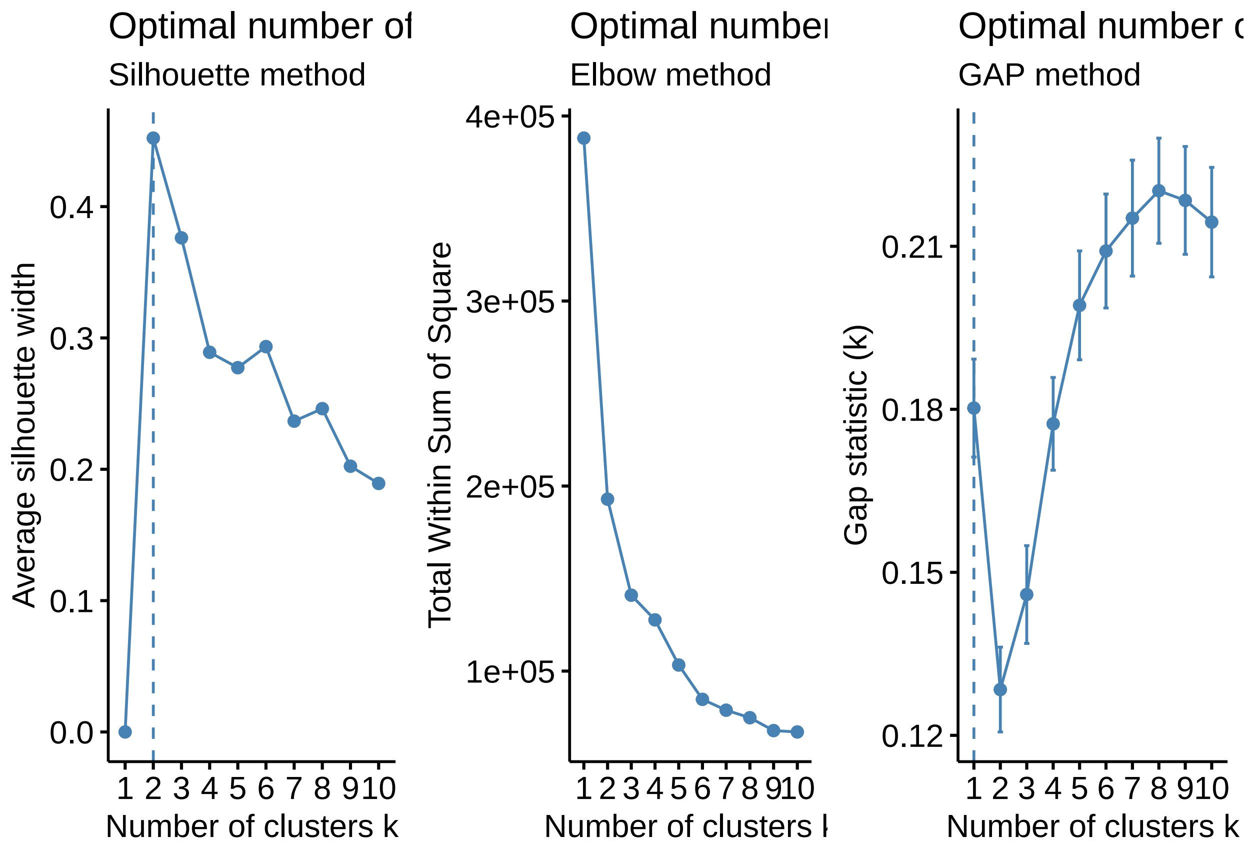
We saw last week that cosine distances are able to compare variables that differ in magnitude. In more technical terms, the cosine distance do not include the length of the vector while the Euclidean distances do. Using K-means we know that cosine distances work better when you include multiple variables with different magnitude (even after being standardized). Before you start, take a moment to reflect on what you expect how cosine and Euclidean distance differ in results when using K-mediods.

K-mediods is very similar to K-means but takes the mediods as cluster centers. Inside the `fviz_nbclust` function, you can use `pam` to calculate the results using K-mediods, and use `diss` to include the cosine distance you've calculated above. Compare the methods `silhouette`, `wss` (=elbow method) and `gap`. The gap method might take some time to compute (i.e a few minutes). When using `gap`, use `nboot = 50` to limit the estimation time. Make sure to use the standardized values of `gross.income`, `Rating`, `Tax.5.`,`Unit.price`.

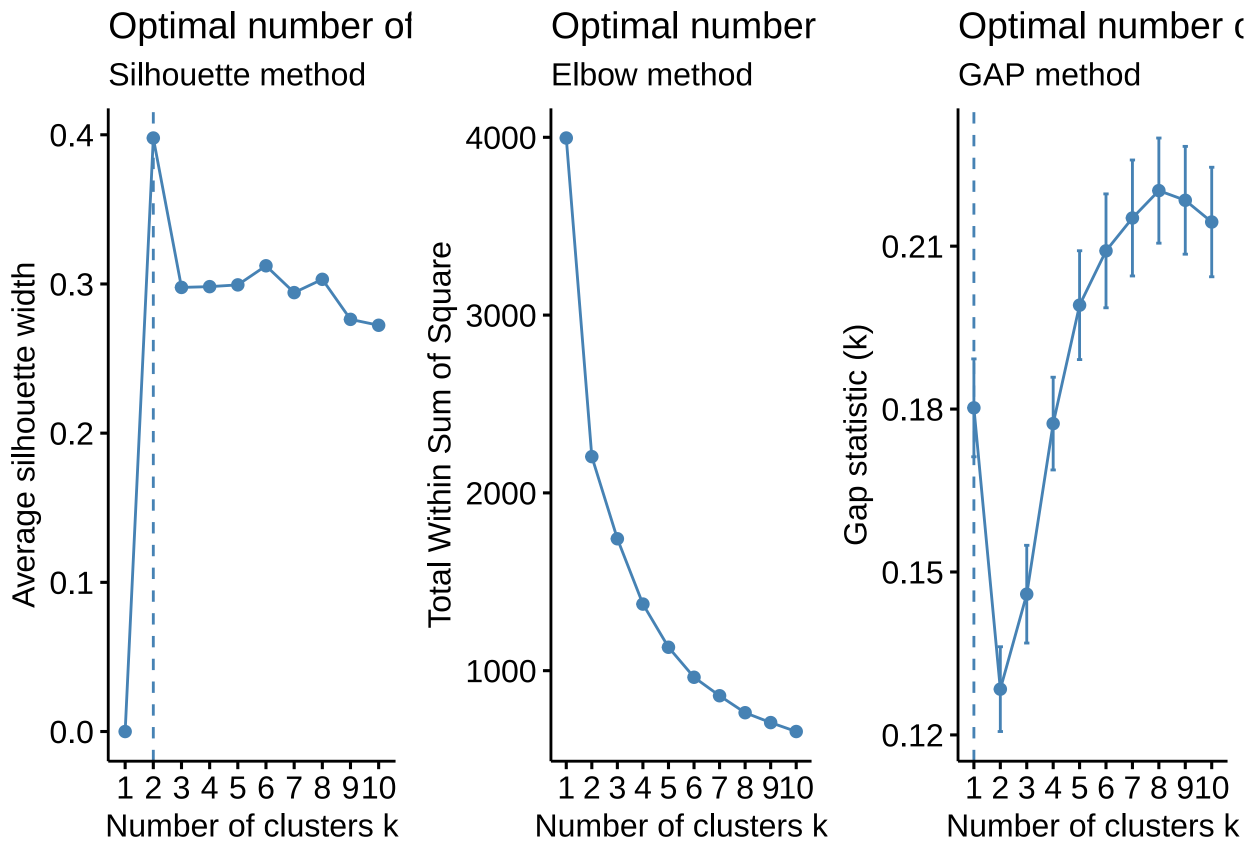
```
f1se<-fviz_nbclust(dfz, pam, method = "silhouette",
                   diss = dist(cosine_dist)) +
  labs(subtitle = "Silhouette method")

f1ee<-fviz_nbclust(dfz, pam, method = "wss",
                   diss = dist(cosine_dist)) +
  labs(subtitle = "Elbow method")

f1ge<-fviz_nbclust(dfz, pam, nboot = 50, method = "gap_stat",
                   diss = dist(cosine_dist)) +
  labs(subtitle = "GAP method")

f1sa<-fviz_nbclust(dfz, pam, method = "silhouette"
                   ) +
  labs(subtitle = "Silhouette method")

f1ea<-fviz_nbclust(dfz, pam, method = "wss") +
  labs(subtitle = "Elbow method")

f1ga<-fviz_nbclust(dfz, pam, nboot = 50, method = "gap_stat") +
  labs(subtitle = "GAP method")
```

Plot the results. You can bind these (saved) plots together in a so-called grid using `grid.arrange` and its argument `nrow`. This implies we expect two grids where each grid contains 3 plots aligned on one row. One grid for the Euclidean distance and one for the cosine distance. Each individual plot should be a one of the methods `wss`, `silhouette` or `gap`.

```
grid.arrange(f1se, f1ee, f1ge, nrow=1)
```




```
grid.arrange(f1sa,f1ea, f1ga, nrow=1)
```



QUESTION TO BE ANSWERED INSIDE RMARKDOWN:

Discuss your findings and reason about possible values for K (K stands for amount of clusters).

---

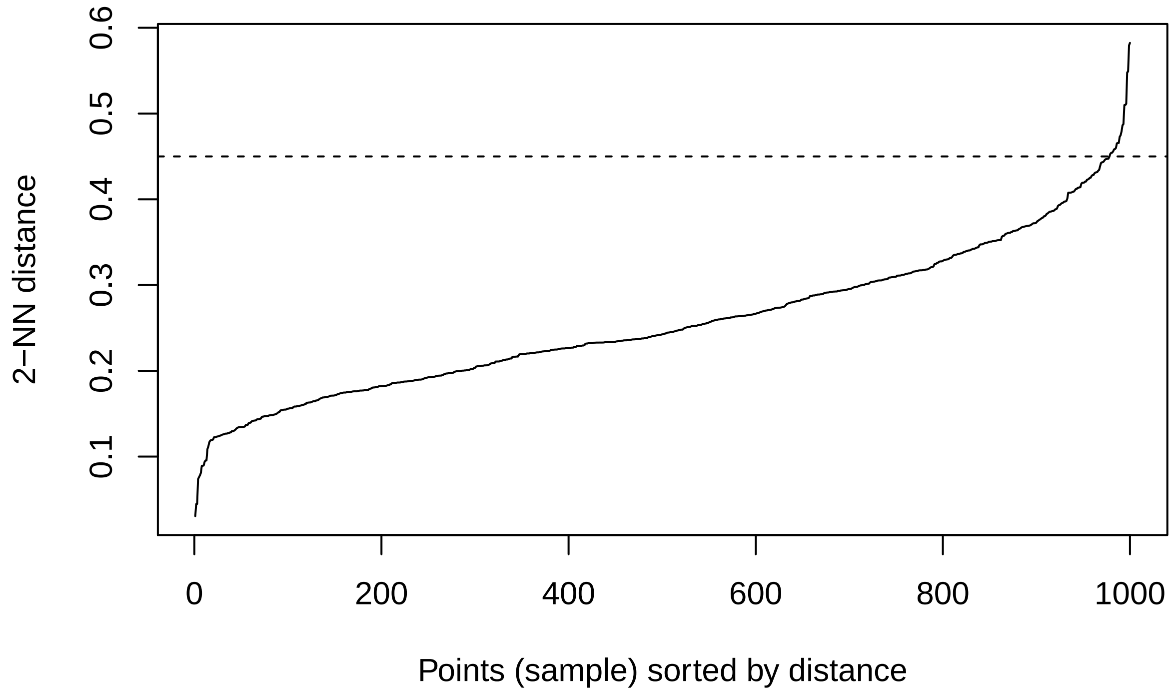
please elaborate your answer (4-5 sentences) about here kmediods is a form of hard clustering, which means that a point must belong to a cluster. Moreover, the kmediods clustering is robust to outliers. The different methods are very close in terms of the optimal number of clusters. The silhouette method gives 2 clusters, the GAP 1, and the elbow method we can assume is either 2 or 3, as the steep curve ends there. The low number of clusters would indicate that the data objects are very close

---

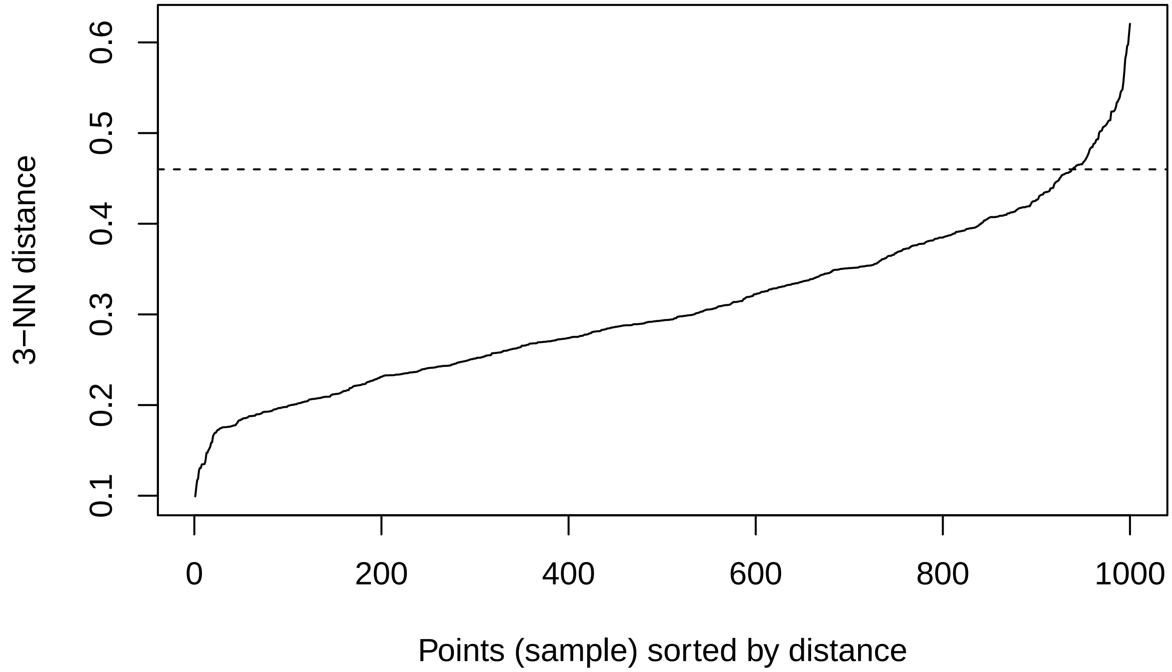
DBSCAN

Next we will implement DBSCAN (Density-based spatial clustering of applications with noise) on the same standardized variables from above. You will obtain the knee-plot to determine the optimal epsilon value. Use the function `dbscan::kNNdistplot` and call the function 3 times for 3 different k values and show the results. Also, draw a horizontal line in the plot at the location of the knee using `abline` on the next line after using the function `dbscan`. Use 2,3, and 4 as reasonable values for K clusters.

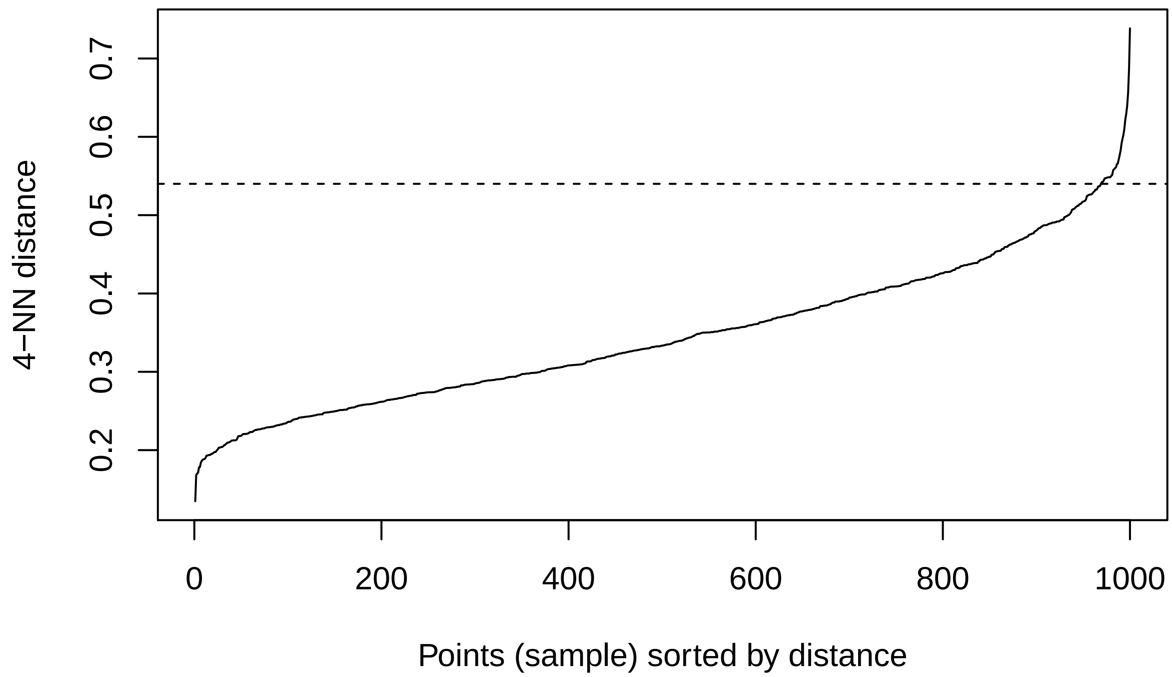
```
dbscan::kNNdistplot(df[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")], k =2)
abline(h=0.45, lty = 2)
```




```
dbscan::kNNdistplot(df[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")], k =3)
abline(h=0.46, lty = 2)
```




```
dbscan::kNNdistplot(df[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")], k =4)
abline(h=0.54, lty = 2)
```



QUESTION TO BE ANSWERED INSIDE RMARKDOWN:

Interpret the differences in epsilon values across the k-solutions, and explain what the optimal epsilon value for k=3 means.

---

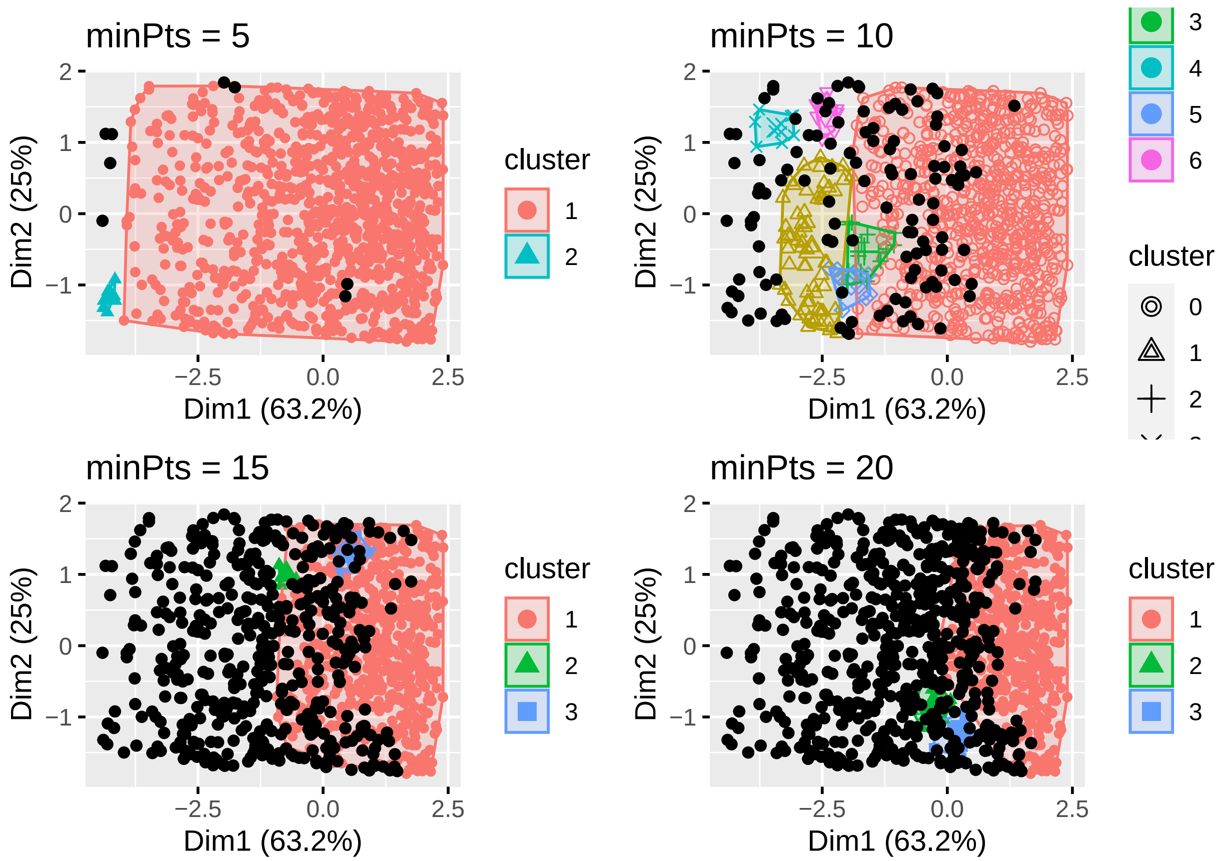
please elaborate your answer (4-5 sentences) about here The optimal epsilon value can be found on the point of maximum curvature. In the plot there are two equal points, so of course we look at the higher epsilon value there. This value defines the distance that separates the two points that are most far apart. for k=3, the distance is almost similar to k =2, but the maximum distance of k = 4 is more apart.

Use epsilon value and experiment with MinPts

Now that we have determined the best epsilon value given k we will use the function `fpc::dbscan` to cluster the data points. Choose the most suitable K-value and experiment with the parameter `MinPts`. The documentation might give you a hint for a reasonable value, try 4 different `MinPts` value and report the effect of this parameter on the outcome. We expect 1 grid with 2x2 plots.(hint: use `grid.arrange`).

```
dfx<-na.omit(df[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")])
db <- fpc::dbscan(dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],eps=.5,MinPts = 5)
f1<-fviz_cluster(db, dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")], stand = FALSE, geo
db <- fpc::dbscan(dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],eps=.5,MinPts = 10)
f2<-fviz_cluster(db, dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")], stand = FALSE, geo
db <- fpc::dbscan(dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],eps=.5,MinPts = 15)
```

```
f3<-fviz_cluster(db, dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],stand = FALSE, geom
db <- fpc::dbscan(dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],eps=.5,MinPts = 20)
f4<-fviz_cluster(db, dfx[, c("ZGross.income", "ZRating", "ZTax.5.", "ZUnit.price")],stand = FALSE, geom

grid.arrange(f1, f2, f3, f4, ncol=2)
```

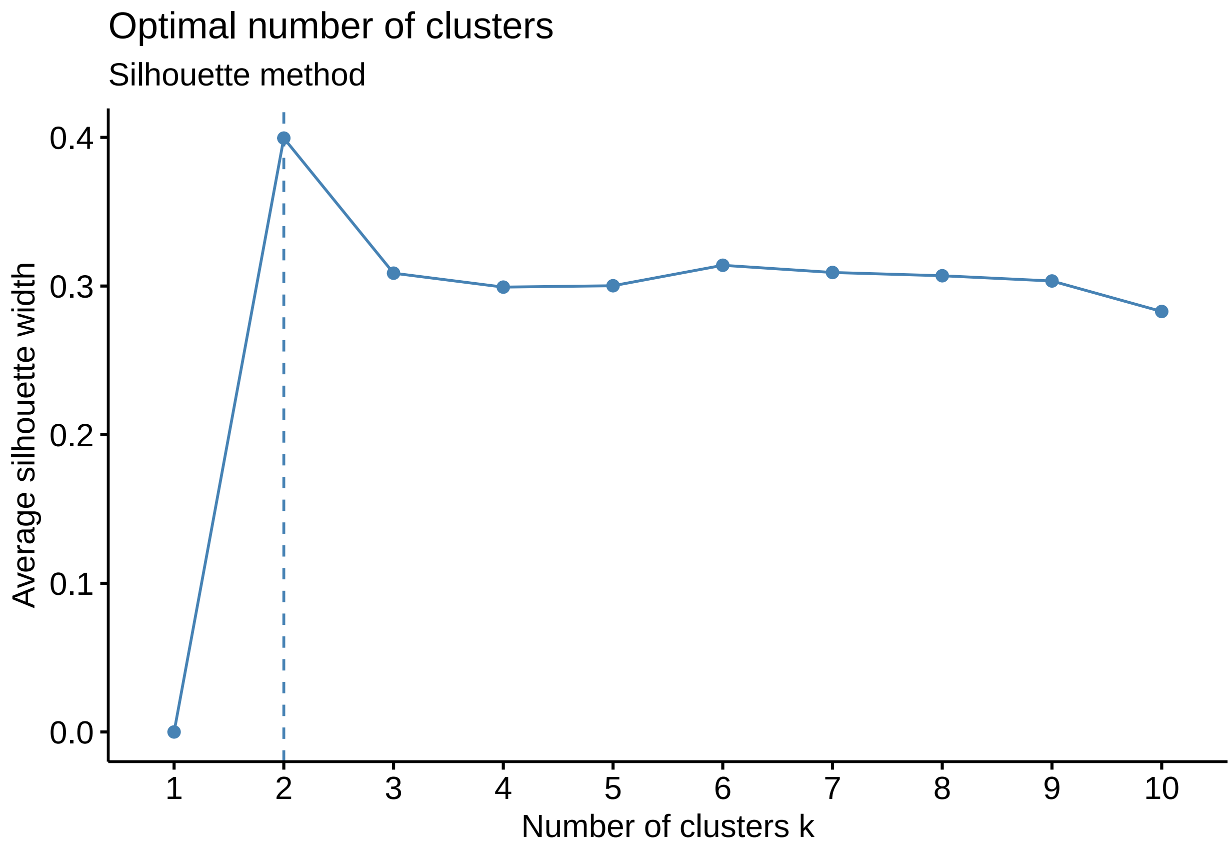


Compare fit

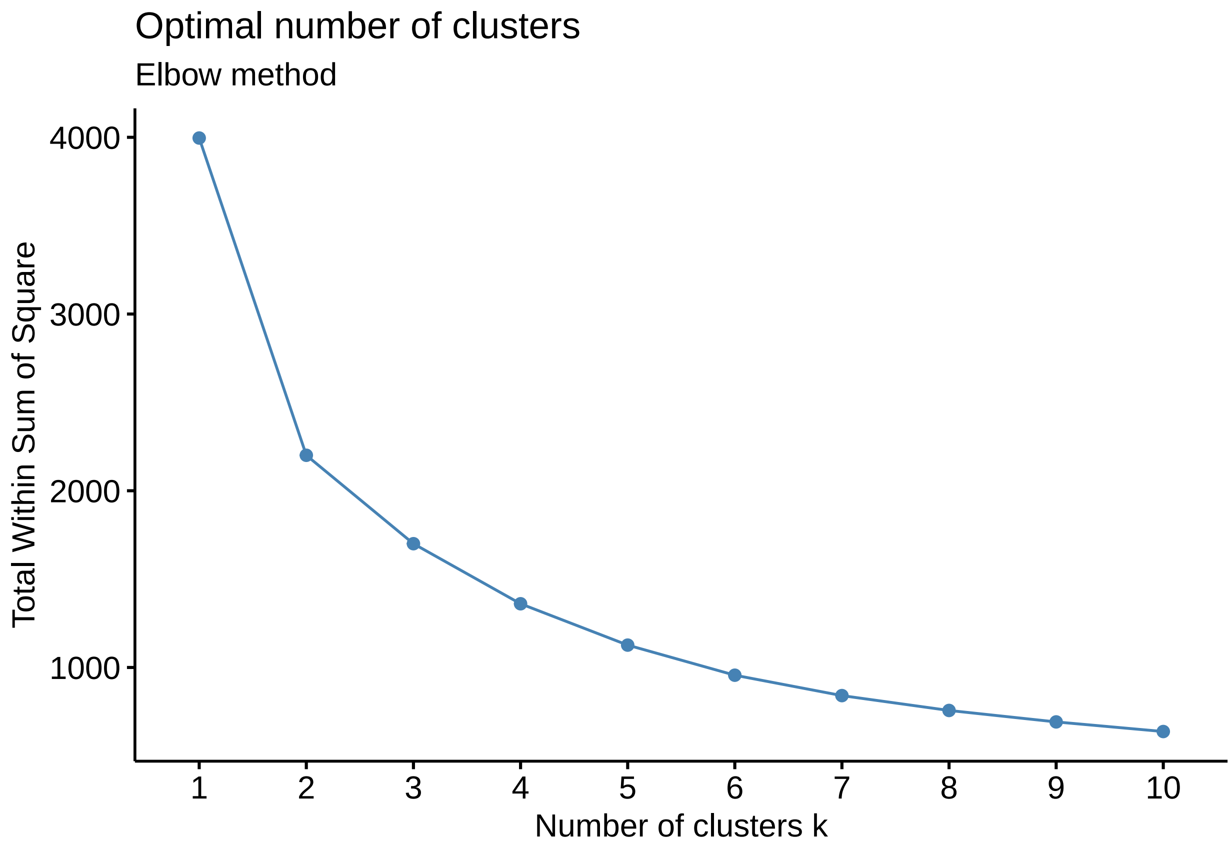
Optimal number of clusters

Finally, you will compare K-means with DBSCAN using the elbow method, gap statistic, and silhouette score. Your goal will be to find the optimal number of clusters, given a distance measure that does justice to the complexity of the data, and a clustering method that suits the data best. You can estimate K-means using `fviz_nbclust` (see assignment 2), with repeating the analysis 25 times (to get to the best solution, see explanation in lecture 3). Use the argument `nstart = 25` to do this, and don't forget to set the seed `set.seed(1234)` on the line before you call `fviz_nbclust`. For the gap statistic you can use again `nboot = 50` to shorten the estimation time.

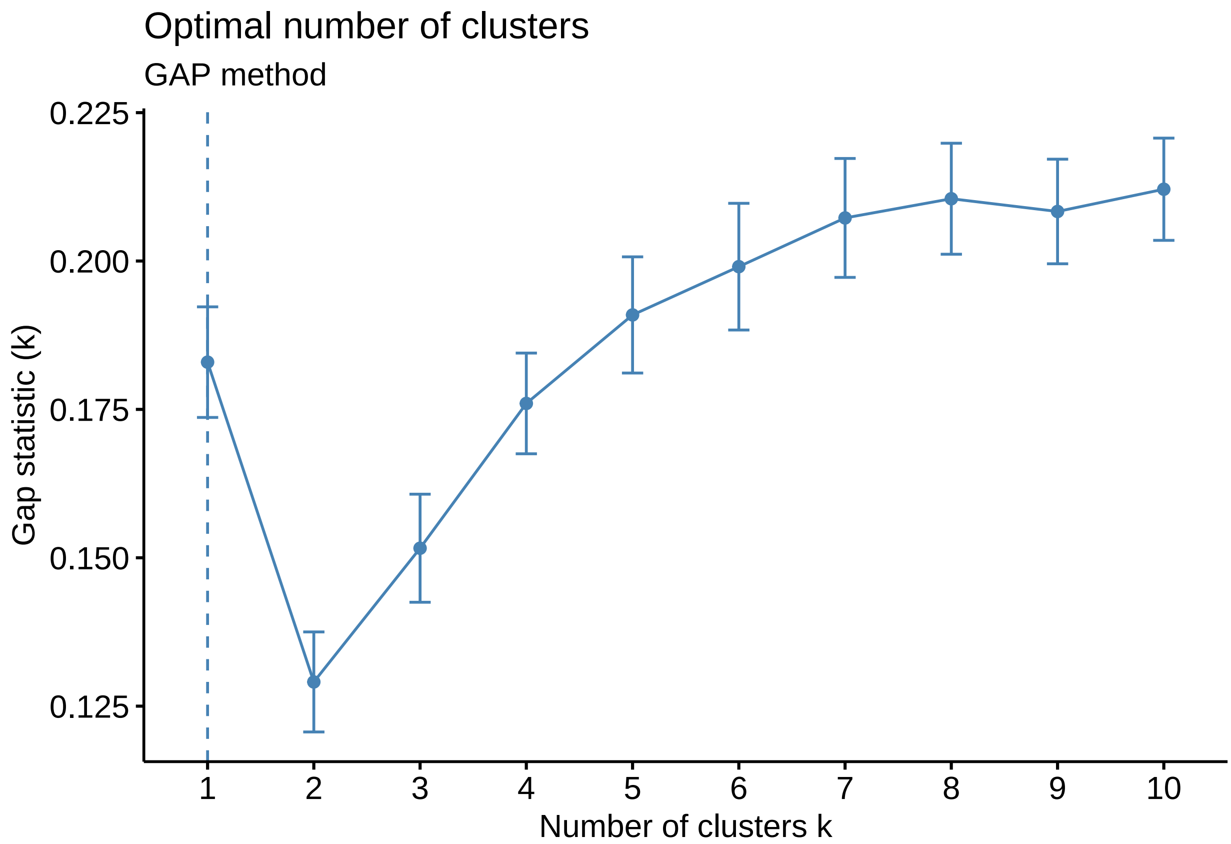
```
set.seed(1234)
fviz_nbclust(df[,c("ZGross.income","ZRating","ZTax.5.","ZUnit.price")], kmeans, nstart = 25, method = "
  labs(subtitle = "Silhouette method")
```




```
set.seed(1234)
fviz_nbclust(df[,c("ZGross.income","ZRating","ZTax.5.","ZUnit.price")], kmeans, nstart = 25, method = "w
  labs(subtitle = "Elbow method")
```




```
set.seed(1234)
fviz_nbclust(df[,c("ZGross.income","ZRating","ZTax.5.","ZUnit.price")], kmeans, nstart = 25, nboot = 50
  labs(subtitle = "GAP method")
```



QUESTION TO BE ANSWERED INSIDE RMARKDOWN:

What is the most optimal number of clusters? Which distance measure and clustering method give the most clear results?

---

please elaborate your answer (4-5 sentences) about here
Two out of three methods indicate that the right number of clusters is 2. This is clear for the silhouette method, and for th

---

Cluster, train, split and evaluate.

Now that we have found a suitable value for K, a corresponding value for `eps` and `MinPts`, we can split the dataframe into a training set containing 85% of the data and a test set being 15% of the data. Use `set.seed(1234)` like before. We will "train" the cluster algorithm on the training set and then inspect the assignment of the data points on the test set. What you do notice? Is the distribution of data points according to what you expected? Explain by visualizing (e.g. a `barplot`) how many data points are assigned to which cluster.

```
# insert your code here
```

We assume the distribution of the training data and the test data are similar, however, this might very well not be the case. If not, what would be a logical next step? You don't have to actually do the next step.

FINAL DISCUSSION OF RESULTS INSIDE RMARKDOWN:

Write a conclusion about the various methods used and conclude your findings. (10-12 sentence max.) In your conclusion consider at least the following attributes:

- *K-modes, K-means and their differences*
- *Euclidean vs Manhattan distance*
- *DBSCAN, the epsilon value and the min points parameter*
- *optimal number of clusters and why (give at least 1 argument that supports your finding)*
- *a possible next step in your analysis based on the results*
- *conclude your findings*

---

please elaborate your answer (4-5 sentences) about here

Besides the obvious different codes for the k-modes and k-means, the methods that are applied give a very similar results. A
