Multivariate Statistical Analysis - Exercise Session 10

24.03.2022

Problem 1: Hierarchical Clustering

First we read the data and import required packages.

```
library(RColorBrewer)
library(reshape2)
library(dendextend)

poll <- read.table("polls.txt", header = TRUE, sep = "\t", row.names = 1)
dim(poll)
```

```
## [1] 11  8
```

```
head(poll)
```

```
##                  KOK  SDP KESK VIHR   PS  VAS RKP  KD
## Uusimaa         26.3 17.0  5.4 19.7  8.3  8.0 8.4 3.1
## Varsinais-Suomi 23.7 19.3 14.0 13.2  7.8 11.1 4.9 3.0
## Kanta-Hame      23.2 25.8 16.4  9.4  8.9 10.1 0.0 5.3
## Pirkanmaa       22.5 22.9 11.5 14.8  8.3  9.8 0.2 4.8
## Paijat-Hame     23.6 25.7 11.8  9.3 10.2  5.0 0.0 5.9
## Kymenlaakso     22.1 26.4 14.3 10.0 10.5  6.4 0.7 5.1
```

a) Visualization of the data

It is quite hard to spot clusters from Figure 1.

```
pairs(poll, gap = 0, upper.panel = NULL, bg = brewer.pal(nrow(poll), "Set3"),
      pch = 21, cex = 1.25)
```

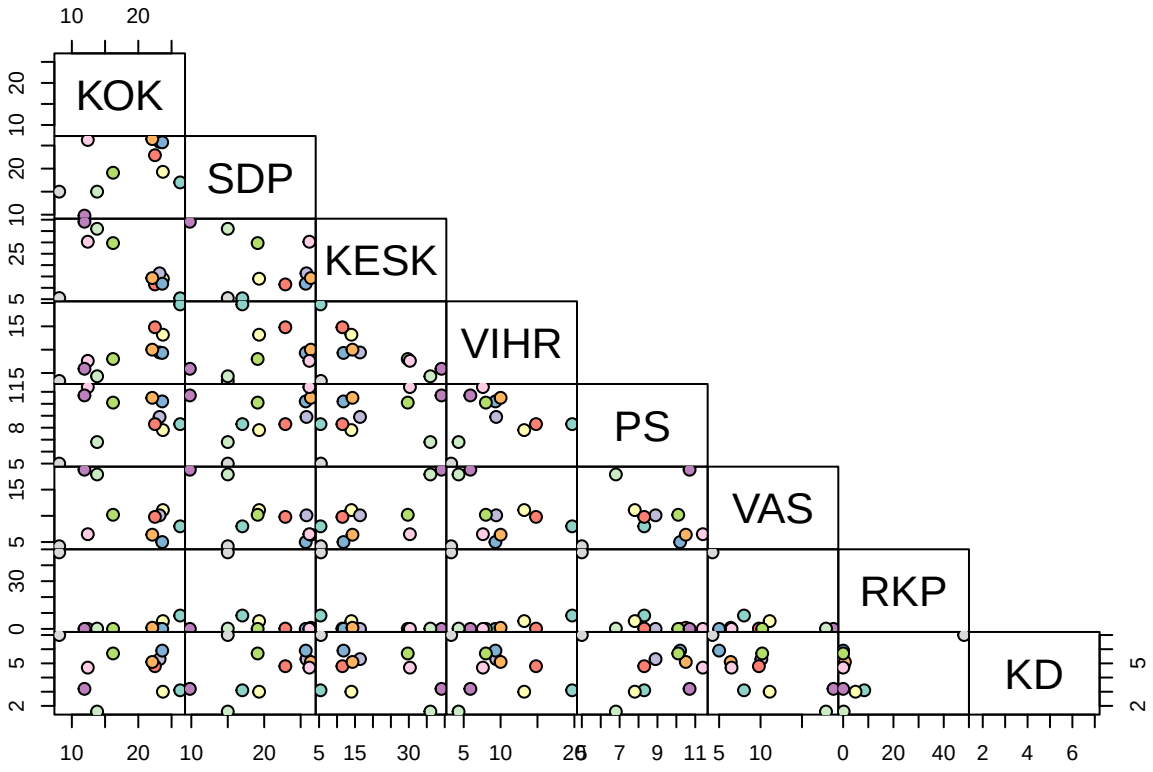

Figure 1: Pairwise scatter plots of the variables. Each region is colored differently.

b) Euclidean distances between the regions

With function `dist` we can calculate distances between rows. Function returns a `dist` object.

```
poll_dist <- dist(poll, method = "euclidean", diag = TRUE, upper = FALSE)
poll_dist
```

```
##                   Uusimaa Varsinais-Suomi Kanta-Hame Pirkanmaa Paijat-Hame
## Uusimaa          0.000000
## Varsinais-Suomi 12.262952        0.000000
## Kanta-Hame      19.857240        9.706184   0.000000
## Pirkanmaa       13.558761        7.104928   7.925276  0.000000
## Paijat-Hame     17.975261       11.679469   7.028513  8.203048    0.000000
## Kymenlaakso     18.708020       10.711209   4.808326  7.747903    3.558089
## Pohjois-Savo    30.318311       19.162985  16.563514 20.877260   21.154432
## Pohjois-Karjala 33.604315       23.094805  18.243355 23.210558   21.816508
## Pohjanmaa       47.125789       48.680592  53.546148 52.721153   52.658618
## Kainuu          42.169894       31.694479  31.536804 34.846808   36.667833
## Lappi           38.775250       27.404379  26.346727 30.441748   31.900940
##                 Kymenlaakso Pohjois-Savo Pohjois-Karjala Pohjanmaa    Kainuu
## Uusimaa
## Varsinais-Suomi
## Kanta-Hame
```

```
## Pirkanmaa
## Paijat-Hame
## Kymenlaakso        0.000000
## Pohjois-Savo      18.566367     0.000000
## Pohjois-Karjala   18.903968     9.041571        0.000000
## Pohjanmaa         52.285275    55.384655       56.058987  0.000000
## Kainuu            34.195760    16.682925       22.440365 61.244592  0.000000
## Lappi             29.337178    12.766362       18.208514 58.976606  7.838367
##                     Lappi
## Uusimaa
## Varsinais-Suomi
## Kanta-Hame
## Pirkanmaa
## Paijat-Hame
## Kymenlaakso
## Pohjois-Savo
## Pohjois-Karjala
## Pohjanmaa
## Kainuu
## Lappi            0.000000
```

```
round(as.matrix(poll_dist), 1)
```

```
##                 Uusimaa Varsinais-Suomi Kanta-Hame Pirkanmaa Paijat-Hame
## Uusimaa             0.0            12.3       19.9      13.6        18.0
## Varsinais-Suomi    12.3             0.0        9.7       7.1        11.7
## Kanta-Hame         19.9             9.7        0.0       7.9         7.0
## Pirkanmaa          13.6             7.1        7.9       0.0         8.2
## Paijat-Hame        18.0            11.7        7.0       8.2         0.0
## Kymenlaakso        18.7            10.7        4.8       7.7         3.6
## Pohjois-Savo       30.3            19.2       16.6      20.9        21.2
## Pohjois-Karjala    33.6            23.1       18.2      23.2        21.8
## Pohjanmaa          47.1            48.7       53.5      52.7        52.7
## Kainuu             42.2            31.7       31.5      34.8        36.7
## Lappi              38.8            27.4       26.3      30.4        31.9
##                 Kymenlaakso Pohjois-Savo Pohjois-Karjala Pohjanmaa Kainuu Lappi
## Uusimaa                18.7         30.3            33.6      47.1   42.2  38.8
## Varsinais-Suomi        10.7         19.2            23.1      48.7   31.7  27.4
## Kanta-Hame              4.8         16.6            18.2      53.5   31.5  26.3
## Pirkanmaa               7.7         20.9            23.2      52.7   34.8  30.4
## Paijat-Hame             3.6         21.2            21.8      52.7   36.7  31.9
## Kymenlaakso             0.0         18.6            18.9      52.3   34.2  29.3
## Pohjois-Savo           18.6          0.0             9.0      55.4   16.7  12.8
## Pohjois-Karjala        18.9          9.0             0.0      56.1   22.4  18.2
## Pohjanmaa              52.3         55.4            56.1       0.0   61.2  59.0
## Kainuu                 34.2         16.7            22.4      61.2    0.0   7.8
## Lappi                  29.3         12.8            18.2      59.0    7.8   0.0
```

c) Agglomerative hierarchical clustering with minimum distance by hand

Here we present one way to rank Euclidean distances between regions.

```
d <- as.matrix(poll_dist)
d[upper.tri(d)] <- -1
d <- melt(d, varnames = c("Region1", "Region2"), value.name = "dist")
d <- d[d$dist > 0, ]
d <- d[order(d$dist), ]
d
```

```
##             Region1         Region2      dist
## 50      Kymenlaakso     Paijat-Hame  3.558089
## 28      Kymenlaakso      Kanta-Hame  4.808326
## 27      Paijat-Hame      Kanta-Hame  7.028513
## 15        Pirkanmaa Varsinais-Suomi  7.104928
## 39      Kymenlaakso       Pirkanmaa  7.747903
## 110           Lappi          Kainuu  7.838367
## 26        Pirkanmaa      Kanta-Hame  7.925276
## 38      Paijat-Hame       Pirkanmaa  8.203048
## 74  Pohjois-Karjala    Pohjois-Savo  9.041571
## 14       Kanta-Hame Varsinais-Suomi  9.706184
## 17      Kymenlaakso Varsinais-Suomi 10.711209
## 16      Paijat-Hame Varsinais-Suomi 11.679469
## 2   Varsinais-Suomi         Uusimaa 12.262952
## 77            Lappi    Pohjois-Savo 12.766362
## 4         Pirkanmaa         Uusimaa 13.558761
## 29     Pohjois-Savo      Kanta-Hame 16.563514
## 76           Kainuu    Pohjois-Savo 16.682925
## 5       Paijat-Hame         Uusimaa 17.975261
## 88            Lappi Pohjois-Karjala 18.208514
## 30  Pohjois-Karjala      Kanta-Hame 18.243355
## 62     Pohjois-Savo     Kymenlaakso 18.566367
## 6       Kymenlaakso         Uusimaa 18.708020
## 63  Pohjois-Karjala     Kymenlaakso 18.903968
## 18     Pohjois-Savo Varsinais-Suomi 19.162985
## 3        Kanta-Hame         Uusimaa 19.857240
## 40     Pohjois-Savo       Pirkanmaa 20.877260
## 51     Pohjois-Savo     Paijat-Hame 21.154432
## 52  Pohjois-Karjala     Paijat-Hame 21.816508
## 87           Kainuu Pohjois-Karjala 22.440365
## 19  Pohjois-Karjala Varsinais-Suomi 23.094805
## 41  Pohjois-Karjala       Pirkanmaa 23.210558
## 33            Lappi      Kanta-Hame 26.346727
## 22            Lappi Varsinais-Suomi 27.404379
## 66            Lappi     Kymenlaakso 29.337178
## 7      Pohjois-Savo         Uusimaa 30.318311
## 44            Lappi       Pirkanmaa 30.441748
## 32           Kainuu      Kanta-Hame 31.536804
## 21           Kainuu Varsinais-Suomi 31.694479
## 55            Lappi     Paijat-Hame 31.900940
## 8   Pohjois-Karjala         Uusimaa 33.604315
## 65           Kainuu     Kymenlaakso 34.195760
## 43           Kainuu       Pirkanmaa 34.846808
## 54           Kainuu     Paijat-Hame 36.667833
## 11            Lappi         Uusimaa 38.775250
## 10           Kainuu         Uusimaa 42.169894
## 9         Pohjanmaa         Uusimaa 47.125789
```

```
## 20        Pohjanmaa Varsinais-Suomi 48.680592
## 64        Pohjanmaa     Kymenlaakso 52.285275
## 53        Pohjanmaa     Paijat-Hame 52.658618
## 42        Pohjanmaa       Pirkanmaa 52.721153
## 31        Pohjanmaa      Kanta-Hame 53.546148
## 75        Pohjanmaa    Pohjois-Savo 55.384655
## 86        Pohjanmaa Pohjois-Karjala 56.058987
## 99            Lappi       Pohjanmaa 58.976606
## 98           Kainuu       Pohjanmaa 61.244592
```

Agglomerative hierarchical clustering is performed according to the following rules:

1. Start from the finest partition: n clusters, each containing one data point x_i.
2. Calculate distances $d_{ij} = d(x_i, x_j)$, where d is an appropriate distance between individuals. In this case we choose d to be Euclidean distance, i.e, $d(x_i, x_j) = \sqrt{(x_i - x_j)^T(x_i - x_j)}$.
3. Find the minimal distance and group together the corresponding individuals.
4. Compute distances between the obtained groups. In this case $d(A, B) = \min_{x_i \in A, x_j \in B} d(x_i, x_j)$.
5. Find the minimal distance and group together the corresponding closest groups.
6. Repeat the steps 4 and 5 until you have one single group.

We refer to the regions by their row numbers, i.e,

$$
\begin{aligned}
&\text{Uusimaa} = 1,\\
&\text{Varsinais-Suomi} = 2,\\
&\text{Kanta-Hame} = 3,\\
&\text{Pirkanmaa} = 4,\\
&\text{Paijat-Hame} = 5,\\
&\text{Kymenlaakso} = 6,\\
&\text{Pohjois-Savo} = 7,\\
&\text{Pohjois-Karjala} = 8,\\
&\text{Pohjanmaa} = 9,\\
&\text{Kainuu} = 10,\\
&\text{Lappi} = 11.
\end{aligned}
$$

Now we are ready to perform clustering by hand with the help of distances `d`.

$$
\begin{aligned}
&(1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11)\\
&(5, 6), (1), (2), (3), (4), (7), (8), (9), (10), (11)\\
&(3, 5, 6), (1), (2), (4), (7), (8), (9), (10), (11)\\
&(3, 5, 6), (2, 4), (1), (7), (8), (9), (10), (11)\\
&(2, 3, 4, 5, 6)(1), (7), (8), (9), (10), (11)\\
&(2, 3, 4, 5, 6)(1), (7), (8), (9), (10, 11)\\
&(2, 3, 4, 5, 6), (7, 8), (9), (10, 11), (1)\\
&(1, 2, 3, 4, 5, 6), (7, 8), (9), (10, 11)\\
&(1, 2, 3, 4, 5, 6), (7, 8, 10, 11), (9)\\
&(1, 2, 3, 4, 5, 6, 7, 8, 10, 11), (9)\\
&(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
\end{aligned}
$$

d) Repeat part c) using the function `hclust`

Function `hclust` takes a `dist` object as an input.

```
poll_clust <- hclust(poll_dist, method = "single")
```

e) Plot the classification tree

```
plot(poll_clust, xlab = "Region / Clusters",
     ylab = "Tree height [Euclidean distance]", main = NA)
```

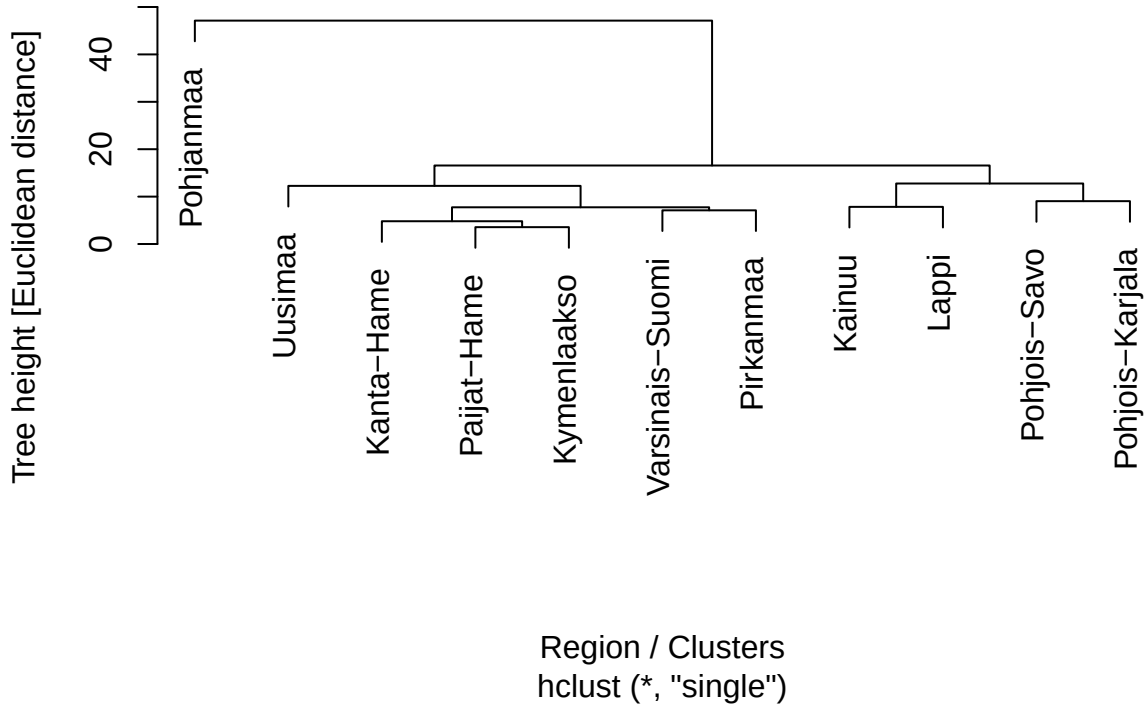

Figure 2: Cluster dendogram.

f) Aggregating the clusters using different linkages

Figure 3 shows that with different linkages we get different results.

```
par(mfrow = c(1, 3))
plot(hclust(poll_dist, method = "single"), main = "Single linkage",
     xlab = "Region / Clusters", sub = NA, hang = -1)
plot(hclust(poll_dist, method = "average"), main = "Average linkage",
     xlab = "Region / Clusters", sub = NA, hang = -1)
plot(hclust(poll_dist, method = "complete"), main = "Complete linkage",
     xlab = "Region / Clusters", sub = NA, hang = -1)
```

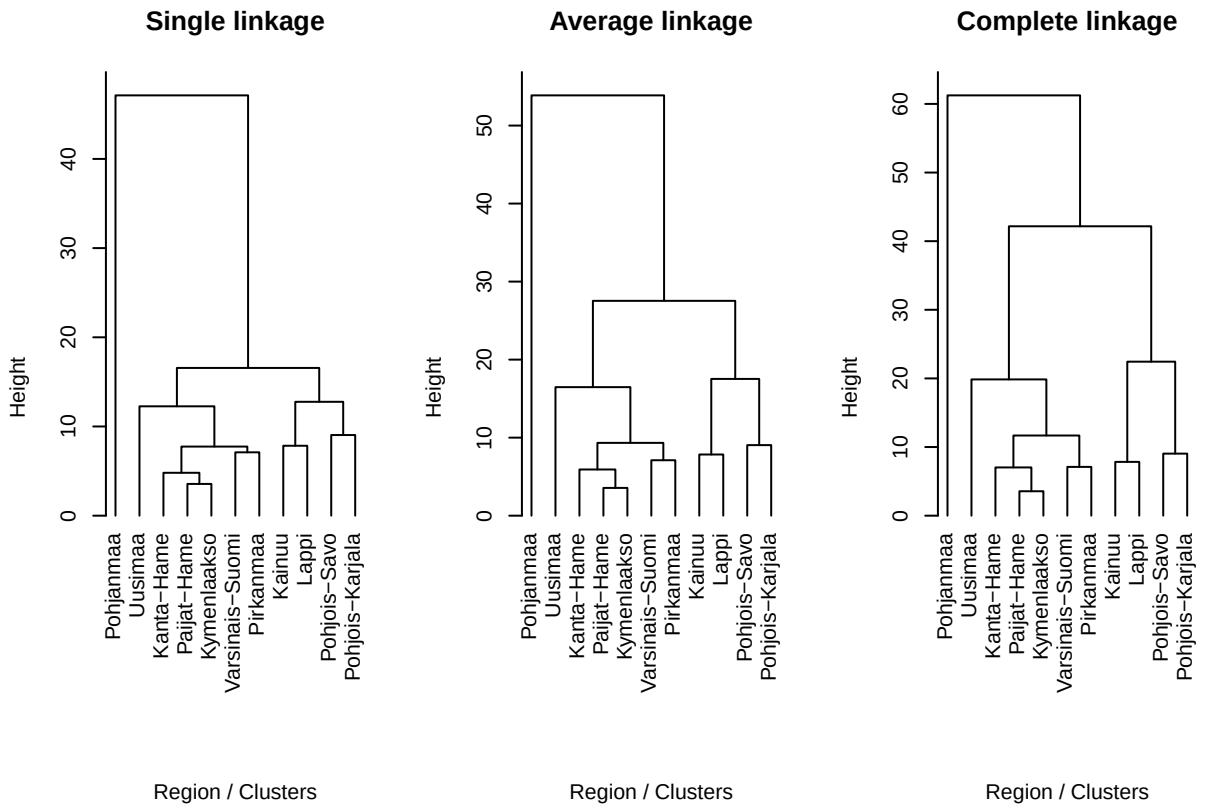

Figure 3: Dendograms with different linkages.

g) Where would you cut the tree?

The difficult part is to choose where to cut the tree, i.e., choose the clusters. This is typically an expert decision as it requires context knowledge. Figure 4 shows the minimum distance tree again.

```
plot(poll_clust,
     xlab = "Region / Clusters", ylab = "Tree height [Euclidean distance]",
     hang = -1, main = NA, sub = NA)
abline(h = 10, lty = 2, col = 2, lwd = 2)
```

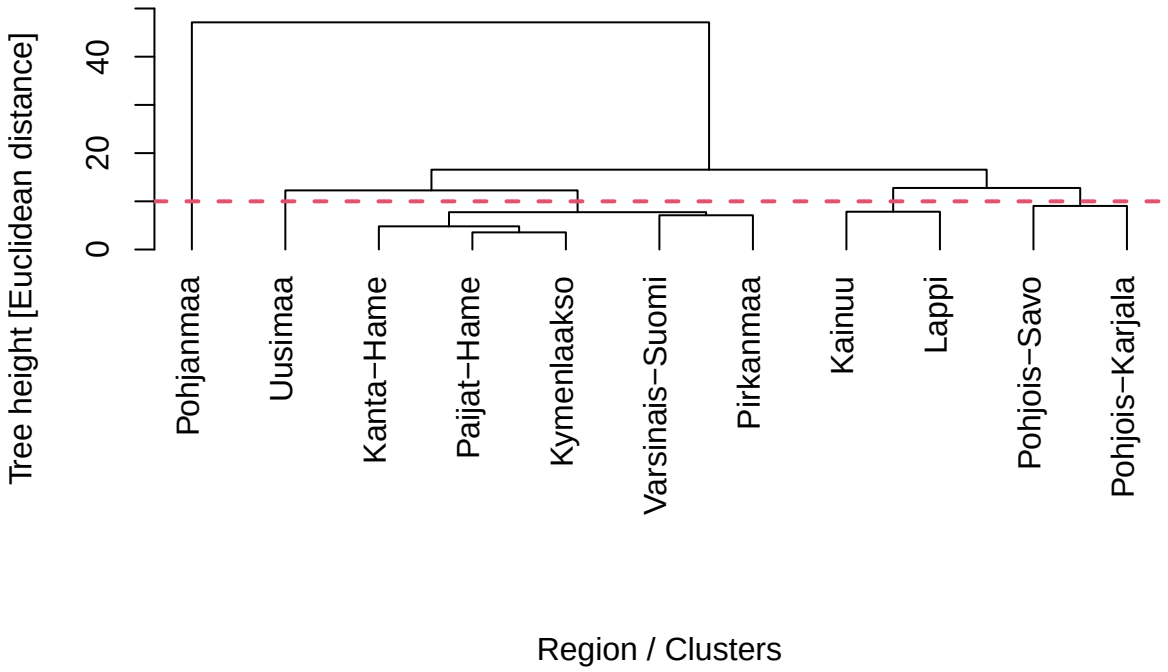

Figure 4: Cluster dendogram.

If we cut the tree at height 10, we obtain 5 clusters. The next step would be to annotate the clusters in a detailed fashion, i.e., to investigate what is the underlying reason the clusters tend to be similar with respect to the covariates used in the analysis. Function `cutree` can be used to cut the tree and obtain the clusters accordingly.

```
cutree(poll_clust, h = 10)
```

```
##         Uusimaa Varsinais-Suomi      Kanta-Hame       Pirkanmaa     Paijat-Hame
##               1               2               2               2               2
##     Kymenlaakso    Pohjois-Savo Pohjois-Karjala       Pohjanmaa          Kainuu
##               2               3               3               4               5
##           Lappi
##               5
```

Library `dendextend` provides more options for plotting dendograms. For example, we can color regions according to which cluster they belong.

```
dend <- as.dendrogram(poll_clust)
dend <- color_labels(dend, h = 10, col = brewer.pal(5, "Dark2"))
plot(dend)
```

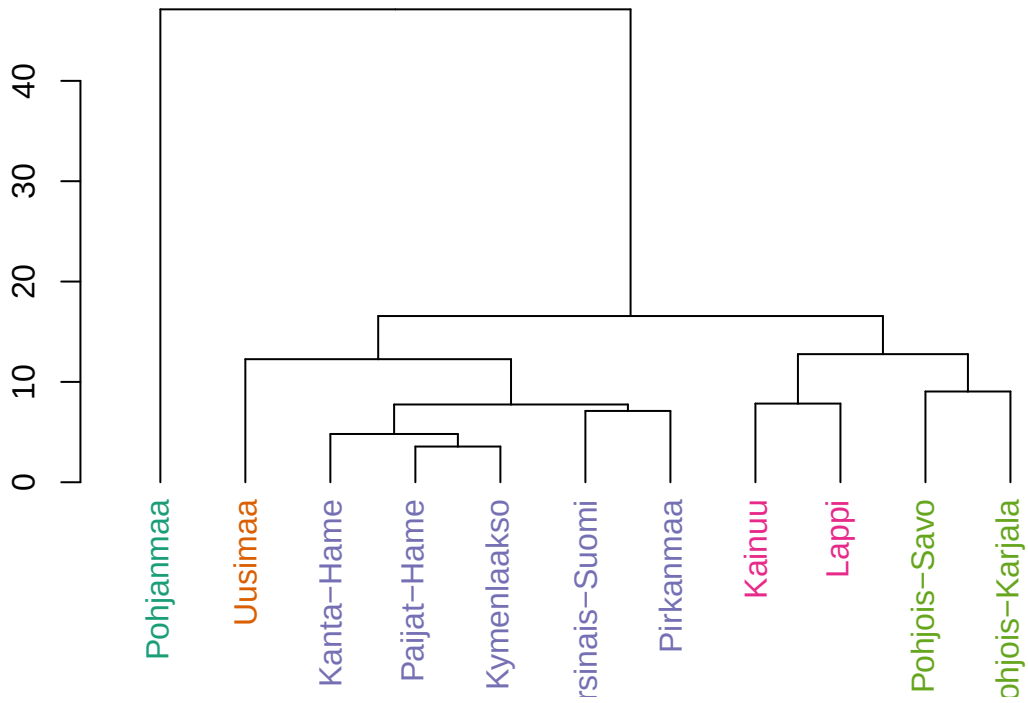

Figure 5: Cluster dendogram where each region is colored according to which cluster they belong.

Problem 2: k-means clustering

```
bank <- read.table("BANK.txt", header = TRUE, sep = "\t")
dim(bank)
```

```
## [1] 200   3
```

```
str(bank)
```

```
## 'data.frame':    200 obs. of  3 variables:
##  $ CODE    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ BOTTOM  : num  9 8.1 8.7 7.5 10.4 9 7.9 7.2 8.2 9.2 ...
##  $ DIAGONAL: num  141 142 142 142 142 ...
```

```
head(bank)
```

```
##   CODE BOTTOM DIAGONAL
## 1    0    9.0    141.0
## 2    0    8.1    141.7
## 3    0    8.7    142.2
## 4    0    7.5    142.0
## 5    0   10.4    141.8
## 6    0    9.0    141.4
```

For practical reasons we change the data type of column `CODE` to a factor.

```
bank$CODE <- factor(ifelse(bank$CODE == 0, "C1", "C2"))
```

a) Apply the k-means algorithm to obtain two clusters

The k-means clustering algorithm has a random part so to be able to reproduce your results, reset the random generator seed number. We can peform k-means clustering with the function `kmeans` that takes the data as an input.

```
set.seed(19032020)

bank_clust <- kmeans(bank[, -1], centers = 2)
bank_clust
```

```
## K-means clustering with 2 clusters of sizes 95, 105
##
## Cluster means:
##      BOTTOM DIAGONAL
## 1 10.660000 139.4663
## 2  8.293333 141.4038
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1 1 1 1 1 1 2
## [112] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 122.30021  90.62381
##  (between_SS / total_SS =  68.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

In two-dimensional case, the results can be easily illustrated with a scatter plot. Figure 6 shows the clustering results.

```
plot(bank[, -1], pch = 16, col = c("blue", "red")[bank_clust$cluster])

# Cluster centers
points(bank_clust$centers[, 1], bank_clust$centers[, 2], pch = 21,
       bg = c("blue", "red"), cex = 2)
```

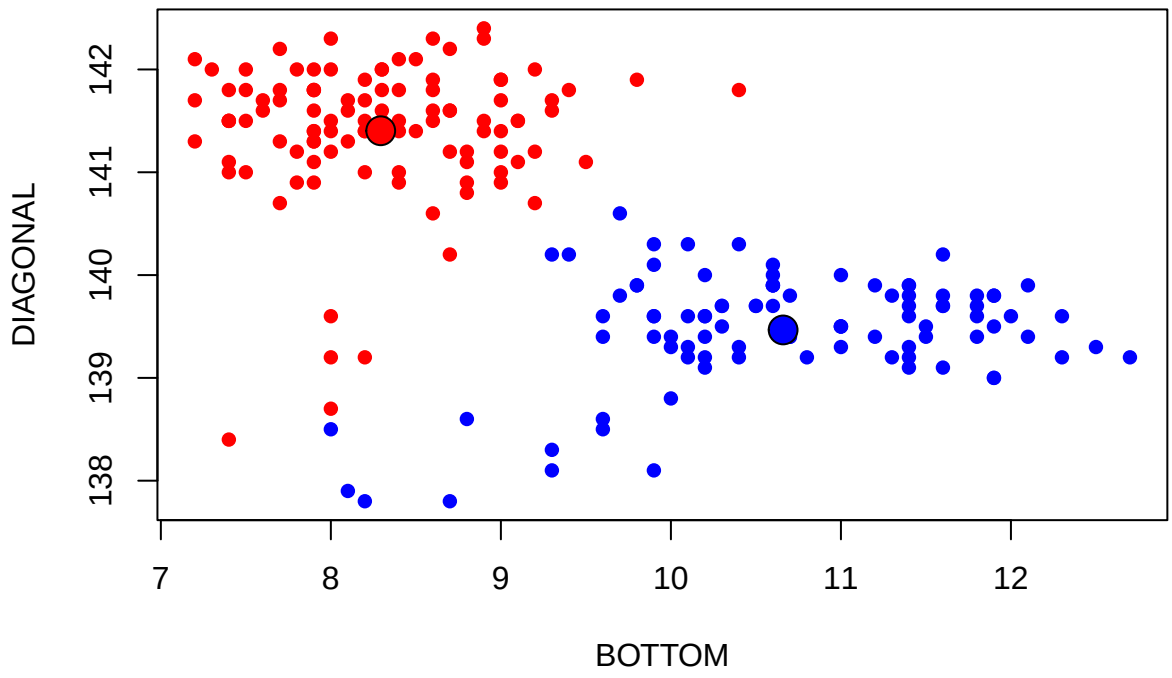

Figure 6: Scatter plot of clustering results. Color represents the clusters and large points represent the centers used by the algorithm.

```
# You can also add true labels but this makes the image hard to read
# text(bank[, -1], labels = bank$CODE, pos = 3)
```

b) How many observations are classified to a wrong category?

Simple way to investigate this is to calculate the confusion matrix.

```
table(bank_clust$cluster, bank[, 1])
```

```
##
##       C1  C2
##   1    0  95
##   2 100   5
```

By the confusion matrix all observations in class `C1` are correctly classified. On the other hand, five observations of class `C2` are wrongly classified to the class `C1`. Thus 2.5% of observations are wrongly classified.

```
5 / nrow(bank)
```

```
## [1] 0.025
```

c) Changing the seed number with $k = 2$

You can run the k-means clustering algorithm multiple times and see how the results change. In this case only the cluster number which `C1` and `C2` are associated changes but the proportions do not. Overall, as the initial centroids of k-means are randomly selected, the random generator seed number changes the results. However, it should not change the results dramatically if there truly exists a structure in the data, which can be detected by the k-means algorithm.

```
# Perform k-means clustering 100 times
bank_clust100 <- replicate(100, kmeans(bank[, -1], centers = 2)$cluster)

# Calculate the cluster proportions
apply(bank_clust100, 2, function(x) table(x) / nrow(bank))
```

```
##    [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## 1 0.525 0.475 0.475 0.525 0.475 0.525 0.525 0.525 0.525 0.525 0.525 0.525 0.475
## 2 0.475 0.525 0.525 0.475 0.525 0.475 0.475 0.475 0.475 0.475 0.475 0.475 0.525
##   [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## 1 0.525 0.475 0.475 0.475 0.475 0.525 0.475 0.475 0.475 0.525 0.525 0.475 0.475
## 2 0.475 0.525 0.525 0.525 0.525 0.475 0.525 0.525 0.525 0.475 0.475 0.525 0.525
##   [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39]
## 1 0.475 0.475 0.525 0.475 0.475 0.475 0.475 0.525 0.475 0.525 0.475 0.525 0.475
## 2 0.525 0.525 0.475 0.525 0.525 0.525 0.525 0.475 0.525 0.475 0.525 0.475 0.525
##   [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
## 1 0.475 0.525 0.475 0.475 0.525 0.475 0.525 0.525 0.475 0.525 0.525 0.475 0.475
## 2 0.525 0.475 0.525 0.525 0.475 0.525 0.475 0.475 0.525 0.475 0.475 0.525 0.525
##   [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65]
## 1 0.475 0.475 0.525 0.475 0.475 0.475 0.475 0.525 0.525 0.525 0.475 0.525 0.475
## 2 0.525 0.525 0.475 0.525 0.525 0.525 0.525 0.475 0.475 0.475 0.525 0.475 0.525
##   [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78]
## 1 0.525 0.475 0.525 0.525 0.525 0.475 0.525 0.525 0.525 0.525 0.475 0.475 0.475
## 2 0.475 0.525 0.475 0.475 0.475 0.525 0.475 0.475 0.475 0.475 0.525 0.525 0.525
##   [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91]
## 1 0.475 0.525 0.475 0.525 0.525 0.475 0.525 0.525 0.475 0.475 0.475 0.525 0.525
## 2 0.525 0.475 0.525 0.475 0.475 0.525 0.475 0.475 0.525 0.525 0.525 0.475 0.475
##   [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## 1 0.475 0.525 0.475 0.525 0.525 0.475 0.475 0.475  0.475
## 2 0.525 0.475 0.525 0.475 0.475 0.525 0.525 0.525  0.525
```

d) Changing the seed number with $k = 3$

Figure 7 shows clustering results for $k = 3$.

```
bank_clust3 <- kmeans(bank[, -1], centers = 3)
plot(bank[, -1], pch = 16,
     col = c("blue", "red", "yellow")[bank_clust3$cluster])

# Cluster centers
points(bank_clust3$centers[, 1], bank_clust3$centers[, 2], pch = 21,
       bg = c("blue", "red", "yellow"), cex = 2)
```

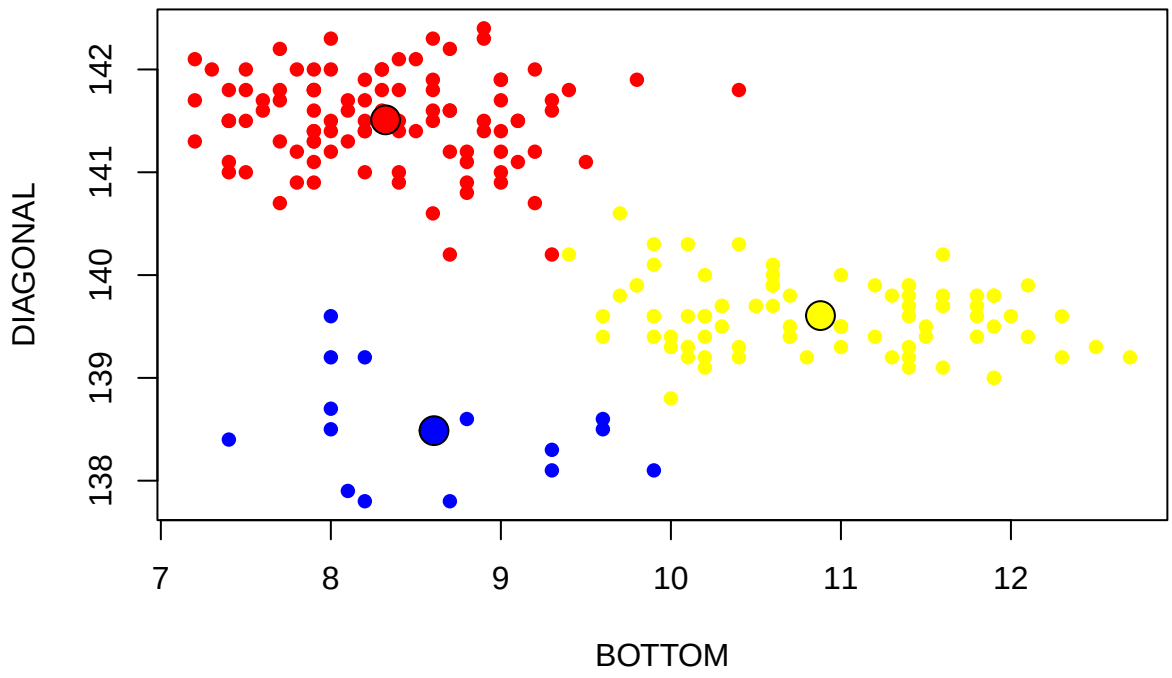

Figure 7: Scatter plot of clustering results with $k = 3$. Color represents the clusters and large points represent the centers used by the algorithm.

As we can see from below snippet of code, proportions of clusters change depending on the seed number when $k = 3$. Therefore, results for k-means clustering are not as stable when we use three clusters instead of two.

```
# Perform k-means clustering 100 times
bank_clust100 <- replicate(100, kmeans(bank[, -1], centers = 3)$cluster)

# Calculate the cluster proportions
apply(bank_clust100, 2, function(x) table(x) / nrow(bank))
```

```
##    [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## 1 0.270 0.235 0.495 0.495 0.235 0.235 0.495 0.495 0.235 0.110 0.495 0.495 0.235
## 2 0.495 0.495 0.235 0.235 0.270 0.495 0.270 0.235 0.495 0.395 0.270 0.235 0.270
## 3 0.235 0.270 0.270 0.270 0.495 0.270 0.235 0.270 0.270 0.495 0.235 0.270 0.495
##   [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## 1 0.295 0.235 0.075 0.495 0.295 0.270 0.495 0.270 0.250   0.4 0.250 0.235 0.075
## 2 0.455 0.495 0.420 0.235 0.250 0.495 0.270 0.495 0.455   0.5 0.455 0.495 0.505
## 3 0.250 0.270 0.505 0.270 0.455 0.235 0.235 0.235 0.295   0.1 0.295 0.270 0.420
##   [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39]
## 1 0.250 0.270 0.495 0.270 0.235 0.250 0.270 0.295 0.270 0.075 0.250 0.250 0.235
## 2 0.295 0.495 0.270 0.235 0.495 0.295 0.235 0.455 0.235 0.505 0.455 0.455 0.495
## 3 0.455 0.235 0.235 0.495 0.270 0.455 0.495 0.250 0.495 0.420 0.295 0.295 0.270
```

```
##   [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
## 1 0.420 0.295   0.4 0.295 0.270 0.270 0.270   0.4 0.270 0.495 0.295 0.235 0.270
## 2 0.075 0.455   0.1 0.455 0.235 0.495 0.235   0.1 0.235 0.270 0.455 0.270 0.495
## 3 0.505 0.250   0.5 0.250 0.495 0.235 0.495   0.5 0.495 0.235 0.250 0.495 0.235
##   [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65]
## 1 0.295 0.235 0.235 0.235 0.235 0.235 0.270 0.235 0.270   0.5 0.495 0.495 0.270
## 2 0.250 0.495 0.270 0.495 0.270 0.270 0.235 0.495 0.495   0.4 0.235 0.270 0.495
## 3 0.455 0.270 0.495 0.270 0.495 0.495 0.495 0.270 0.235   0.1 0.270 0.235 0.235
##   [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78]
## 1 0.270 0.495 0.270 0.075 0.505 0.235 0.270 0.235 0.295 0.455 0.295 0.420 0.270
## 2 0.235 0.235 0.235 0.505 0.420 0.495 0.495 0.270 0.455 0.250 0.455 0.075 0.495
## 3 0.495 0.270 0.495 0.420 0.075 0.270 0.235 0.495 0.250 0.295 0.250 0.505 0.235
##   [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91]
## 1 0.420 0.495 0.270 0.235   0.1 0.495 0.270 0.495 0.270 0.235   0.1 0.505 0.455
## 2 0.505 0.235 0.235 0.495   0.4 0.235 0.495 0.270 0.495 0.495   0.4 0.420 0.250
## 3 0.075 0.270 0.495 0.270   0.5 0.270 0.235 0.235 0.235 0.270   0.5 0.075 0.295
##   [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## 1 0.235 0.495 0.495 0.235 0.250 0.495 0.495 0.495  0.270
## 2 0.270 0.270 0.235 0.495 0.455 0.270 0.270 0.235  0.235
## 3 0.495 0.235 0.270 0.270 0.295 0.235 0.235 0.270  0.495
```

Remark that typically in the clustering problem the labels are unknown. Therefore, it is essential to use expert validation / analyze the obtained clusters carefully. As we can see here, the clustering algorithm will find as many clusters as we tell it to find!