Exercise 3 Trashaj Alberto 1075402

alberto.trashaj

October 2023

1 Point 1

1.1 1.a

In the first point we consider a dataset with 4 observations in 5 variables:

$$x_1 = (blue, 1, 1, 0, 12)$$
$$x_2 = (red, 0, 0, NA, NA)$$
$$x_3 = (red, 1, 0, NA, 17)$$
$$x_4 = (green, 1, 0, 0, 21)$$

Before computing the Gower coefficient between all pairs of observations let's recall the formula:

$$d_G(X_i, X_j) = \sum_{l=1}^{p} \frac{w_l}{s_l} \sigma$$

So, by applying the function written before, and taking into account the variables

$$d_G(X_1, X_2) = \frac{(1+1+1+0+0)}{3} = 1$$

$$d_G(X_1, X_3) = 0.639$$

$$d_G(X_1, X_4) = 0.75$$

$$d_G(X_2, X_3) = 0.5$$

$$d_G(X_2, X_4) = 1$$

$$d_G(X_3, X_4) = 0.482$$

1.2 1.b

Let's compute the Gower dissimilarity using the daisy function in R

```
x <- data.frame(
color <- as.factor(c("blue","red","red","green")),
  x1 <- c(1, 0, 1, 1),
  x2 <- c(1, 0, 0, 0),
  x3 <- c(0, NA, NA, 0),
  x4 <- c(12, NA, 17, 21)
)

str(x)

gower_diss <- daisy(x, metric="gower", type=list(asymm=c(2,3,4), factor=c(1)))
gower_diss
```

Gives as output the followin matrix dissimilarity:

```
   1 2 3
2 1.0000000
3 0.6388889 0.5000000
4 0.7500000 1.0000000 0.4814815
```

which is equal to the manually computed before.

2 Point 2

2.1 2.a

Here we have to show that the Mahlanobis distance and the Simple Matching distance are dissimilarities and also distances:

2.1.1 Mahlanobis Distance

Let's take 3 points in the euclidian space: x, y, z.
The Mahlanobis distance between two points is defined as:

$$d_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

Now, what we want to prove is essentially the following inequality:

$$d_M(x,y) + d_M(y,z) \geq d_M(x,z)$$

We can note that $S = UDU^T$, therefore the Mahlanobis distance between x and y can be written as:

$$D_M(x,y) = \sqrt{(x-y)^T (UDU^T)^{-1}(x-y)}$$

which is equal to

$$= \sqrt{(x-y)^T (U^{(-1)^T} D^{-1} U^{-1})(x-y)}$$

and since U is orthogonal, $U^{-1} = U^T$

$$= \sqrt{(x-y)^T (U^{(T)^T} D^{-1} U^{-1})(x-y)}$$

therefore we have

$$= \sqrt{(x-y)^T (U D^{-1} U^{-1})(x-y)}$$

Now we can define

$$\omega = (x-y)^T U D^{-\frac{1}{2}}$$

The distance than can be summarize as

$$d_M(x,y) = \sqrt{\omega \times \omega^T} = ||\kappa_1||$$

which is an euclidean distance.

Since it is an euclidean distance, every euclidean distance respect the triangle inequality. Therefore, the Mahlanobis distance fulfill the triangle inequality.

2.1.2 Simple matching distance

Let's take again three points: x, y and z.
The sample matching distance is defined as:

$$d_{sm}(x,y) = \frac{1}{p} \sum_{j=1}^{p} 1(x \neq y)$$

As before, we want to prove that the triangle inequality is satisfied:

$$d_{sm}(x,y) + d_{sm}(y,z) \geq d_{sm}(x,z)$$

which is

$$\frac{1}{p} \sum_{j=1}^{p} 1(x \neq y) + \frac{1}{p} \sum_{j=1}^{p} 1(y \neq z) \geq \frac{1}{p} \sum_{j=1}^{p} 1(x \neq z)$$

Let's see what happens if $x \neq y \neq z$: all the indicator function will take value 1.
Therefore:

$$\frac{1}{p} \times 1 + \frac{1}{p} \times 1 \geq \frac{1}{p} \times 1$$

which is always true since $p > 0$. Instead, if $x = y$, the indicator function of the first term take 0 value

$$0 + \frac{1}{p} \sum_{j=1}^{p} 1(y \neq z) \geq \frac{1}{p} \sum_{j=1}^{p} 1(x \neq z)$$

and since $x = y$ we can substitute in the second term of the inequality and we have

$$\frac{1}{p} \sum_{j=1}^{p} 1(x \neq z) \geq \frac{1}{p} \sum_{j=1}^{p} 1(x \neq z)$$

which is clearly always true.
Same steps can be done in the case when $y = z$.
In conclusion, even the sample matching distance satisfy the triangle inequality.

2.2 2.b

Here we have to give counterexamples to show that the correlation dissimilarity and the Gower distance do not fulfill the triangle inequality with an example:

2.2.1 Correlation dissimilarity

Let's recall that the correlation dissimilarity can be computed as

$$d_{corr}(x_1, x_2) = \frac{1}{2}(1 - r(x_1, x_2))$$

where $r(x_1, x_2)$ is the sample Pearson correlation coefficient.
Let's use R

```
xx <- c(1,0,3)
yy <- c(2,3,4)
zz <- c(0,1,1)

d_xy <- 0.5 * (1 - cor(xx, yy))
d_xz <- 0.5 * (1 - cor(xx, zz))
d_yz <- 0.5 * (1 - cor(zz, yy))

d_xy + d_yz >= d_xz
```

gives as output FALSE. So, it doesn't fulfill the triangle inequality.

2.2.2 Mahlanobis distance

```
x <- data.frame(
  x2 <- c(1, 0, 0),
  x3 <- c(1, NA, NA),
  x4 <- c(1, 1, NA)
)
gower_diss <- daisy(x, metric="gower")
gower_diss
```

As we can see from the output the triangle inequality is not respected. Dissimilarities : 1 2
2 0.5
3 1.0 0.0

3 Point 3

3.1 a

The Jaccard Distance seems to be more appropriate. This is because it specifically addresses the agreement on a non-default option: this distance would be particularly relevant if the political scientist is interested in understanding the

level of consensus or divergence among respondents regarding an alternative approach to taxes and health system investment.

3.2 b

In this question we have more options to discuss: since the nature of the data, regardless the distance choosen, it would be appropriate to scale the data in order to deal to the fact that we have different units and sources of data.
Given the nature of the data, I would suggest considering both Euclidean distance on scaled data and Mahalanobis distance. These two options take into account variable scaling and correlations, which are likely important factors in assessing avalanche danger.

4 Point 4

In this section we are going to cluster Covid data: the dataframe provided present 559 days observed for 179 countries worldwide.
The goal here is to try to clusterize the countries based on similar developements of the disease.
From a non-statistical point of view, COVID is a disease that spreads through contact, so we expect to find similar disease trends in nearby countries, neighboring ones, or those with strong connections.

```
```{r}
covid2021 <- read.table("~/Desktop/Universita /Unsupervised/covid2021.dat", quo
```
```

```
Select the variables that we need to cluster
```{r}
covid2021cl <- covid2021[,5:559] # This selects the variables for clustering
covid2021cl
```
```

```
Here we plot all the countries
```{r}
plot(1:555,covid2021cl[1,],type="l",ylim=c(0,25),
ylab="New cases over one week per 1000 inhabitants",
xlab="Day (1 April 2020-7 October 2021)")
for(i in 2:179)
points(1:555,covid2021cl[i,],type="l")
```
```

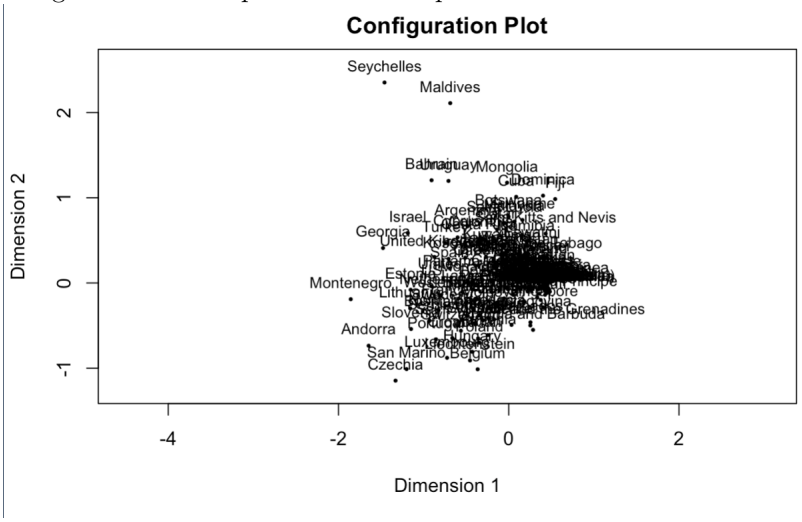
This graph show some huge spikes in some days, expecially around the day 400.
Now that we imported data and visualize the developements of it, we can try to compute some different distance matrix and plot the mds.
Let's compute now the distance matrix using daisy function

```r
library(cluster)
dist_covid_all_eucl <- daisy(covid2021cl, metric = "euclidean")
dist_covid_all_man <- daisy(covid2021cl, metric = "manhattan")
```

```r
# Given those distances matrix, we can visualize
# them in a MDS plot using the smacof library
```

```r
library(smacof)

mdscountries_eucl <- mds(dist_covid_all_eucl,ndim=2)
mdscountries_man <- mds(dist_covid_all_man, ndim= 2)
plot(mdscountries_eucl)
plot(mdscountries_man)
```

This plot doesn't seems so useful since we have too many countries and it's hard to distinguish them, despite this we can see some countries like Maldives and Seychelles plotted far away from the rest of the countries. As we can see from the image below that represent the mds plot of the euclidean distance matrix.

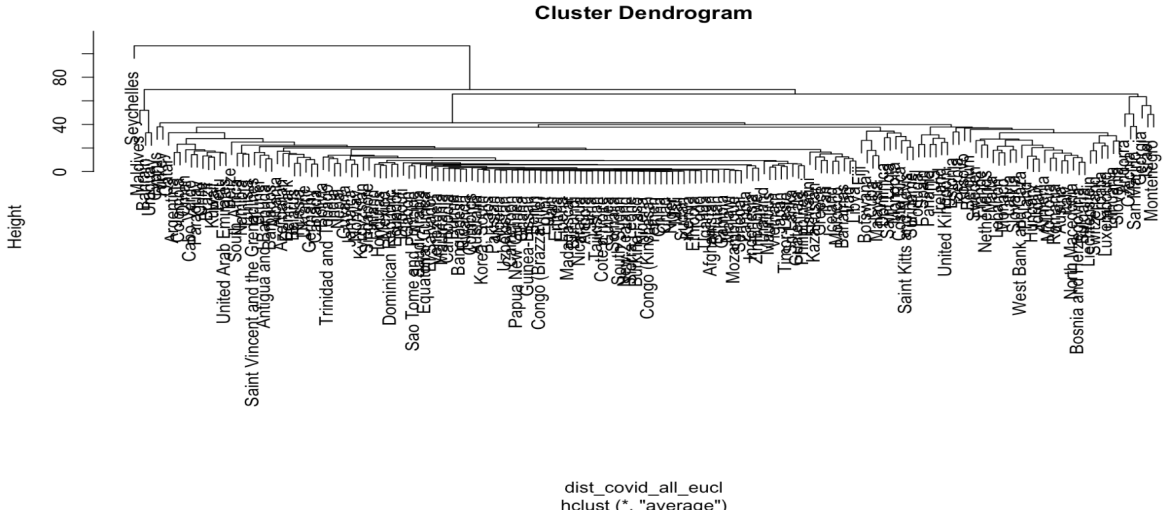


Let's see what happens when we hierarchically clusterize the dataset, using different type of distances

```r
aver_eucl_all <- hclust(dist_covid_all_eucl, method = "average")
sing_eucl_all <- hclust(dist_covid_all_eucl, method = "single")
```

```
comp_eucl_all <- hclust(dist_covid_all_eucl, method = "complete")
plot(comp_eucl_all)
plot(sing_eucl_all)
plot(aver_eucl_all)
```

For example, if we plot the average method we will get this dendrogram



Even this seems too caotic.
My idea to have a better understanding of the Covid developement is to divide the dataframe in sub-dataframes that correspond to divide the world into continents: the original dataframe already present a variable called continent, so we will use that to divide it.

First, let's get the names of the continents in the dataset

```{r}
df_list <- split(covid2021, covid2021$continent)
names <- setNames(df_list, unique(covid2021$continent))
names
```

```{r}
africa <- names$Asia[,5:559]
australia <- names$Africa[,5:559]
europe <- names$`South America`[,5:559]
south_america <- names$`North America`[,5:559]
north_america <- names$Australia[,5:559]
```

```
central_america <- names$`Central America`[,5:559]
asia <- names$Europe[,5:559]

europe
```
```
Now that we splitted our dataframe, let's visualize again what
we did before for every different continent:

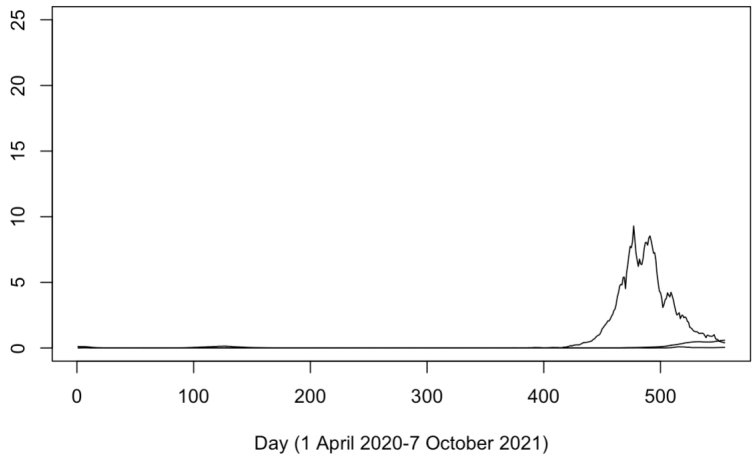
```{r}
continent_names <- list(asia, africa, north_america,
                        south_america, central_america,
                        europe, australia)

for (j in continent_names){
  plot(1:555,j[1,],type="l",ylim=c(0,25),
  ylab="Continent",
  xlab="Day (1 April 2020-7 October 2021)")

  for(i in 2:179){
    points(1:555,j[i,],type="l")
  }
}

```
```

We can see better now that different continents present very different Covid spreading: for example, in Australia where we have 3 countries, just after 400 days of observations we can see an increase in the number of cases.




```
Now, let's take every single continent and see some
multidimensional scaling plot and some hierarchical clustering.
```

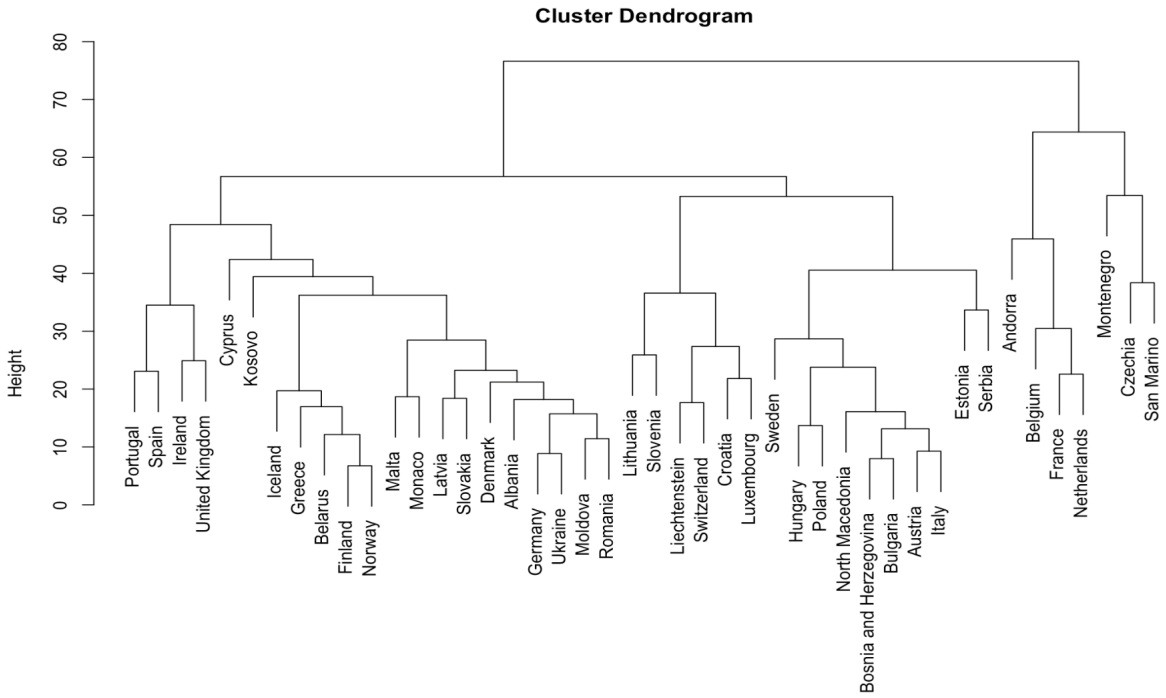
```{r}
dist_eucl_europe <- daisy(europe, metric = "euclidean")
mdscountries_eucl <- mds(dist_eucl_europe, ndim=2)
plot(mdscountries_eucl)
```
Let's see some hierarchical clusters

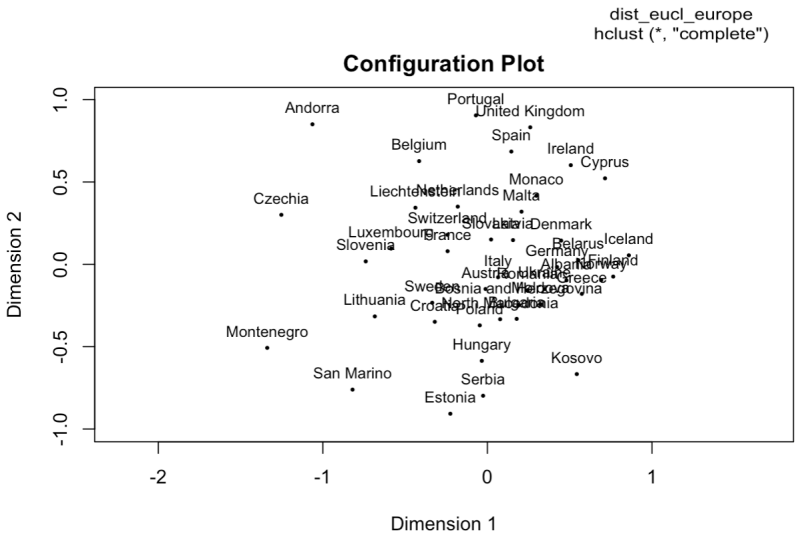
```{r}
clust_europe_single <- hclust(dist_eucl_europe, method = "single")
clust_europe_average <- hclust(dist_eucl_europe, method = "average")
clust_europe_complete <- hclust(dist_eucl_europe, method = "complete")

plot(clust_europe_single)
plot(clust_europe_average)
plot(clust_europe_complete)

```

From a intuitive point of view what would we expect is that the countries which are geographically close should be clustered in the same group, since Covid is a influenza virus, i.e., it's spread out by contact.

As we would expect, in fact, some countries which are geographically near (see Spain and Portugal, or Ireland and United Kingdom) are clustered in the same group when we use the complete and average method.

Moreover, the mds now it's more clear.



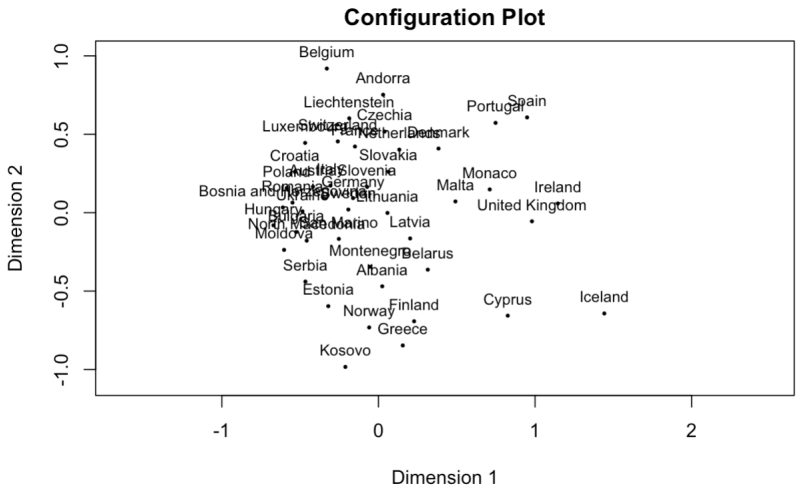


Let's see what happens if we choose a different type of distance as the correlation distance:

```{r}
```

```
europe_corr <- cor(t(europe), method="pearson")

dist_corr_europe <- as.dist(0.5-(europe_corr)/2)
dist_corr_europe

```
Now that we have a dist object we can plot the MDS of the distance matrix

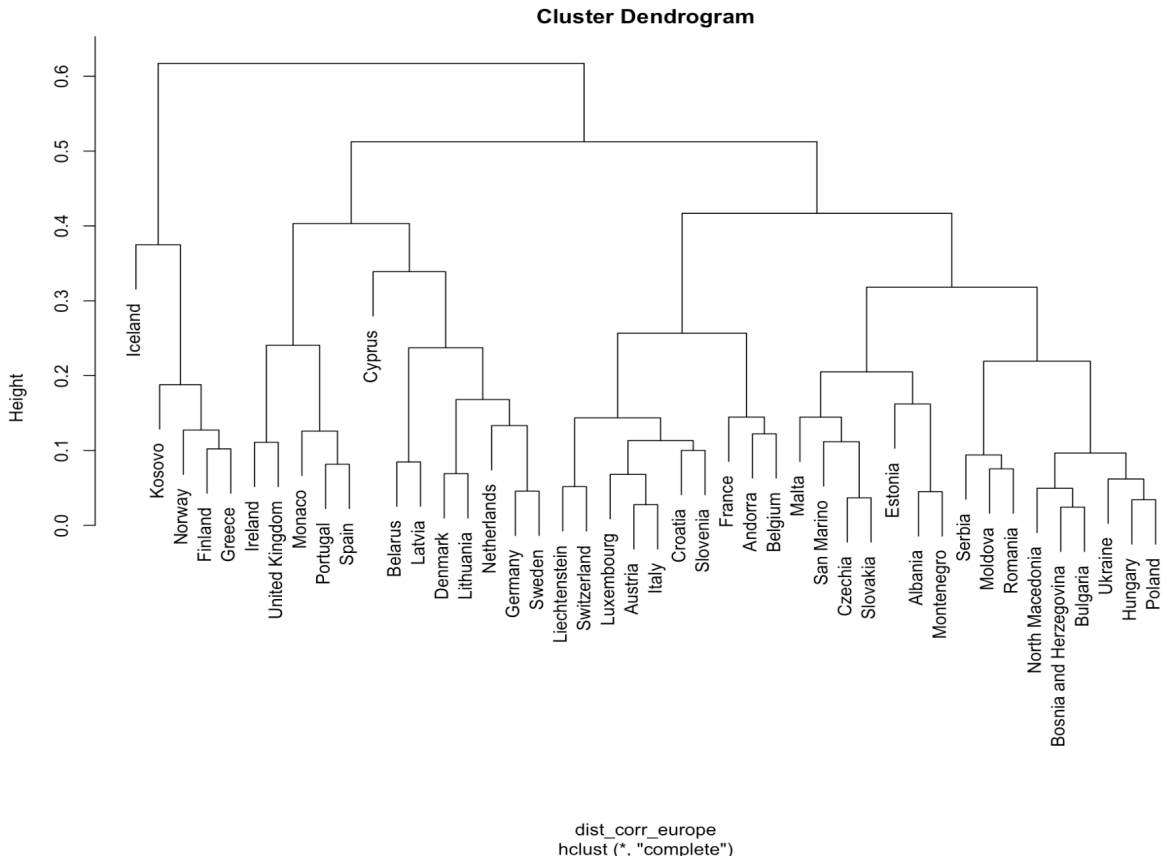
```{r}
mds_europe_corr <- mds(dist_corr_europe, ndim=2)
plot(mds_europe_corr)
```



From this plot we can see, as we said before, that some countries which are close along the longitude and the latitude are also close in our MDS plot.

Let's see what happens if we apply the hclust function to this new distance

```{r}
clust_europe_single_corr <- hclust(dist_corr_europe, method="single")
clust_europe_average_corr <- hclust(dist_corr_europe, method="average")
clust_europe_complete_corr <- hclust(dist_corr_europe, method="complete")

plot(clust_europe_single_corr)
plot(clust_europe_average_corr)
plot(clust_europe_complete_corr)
```



Similar result we get by using the correlation distance: as before, some countries which are geographically close share more or less the same number of cases for same days, clustering them therefore in the same group.

Let's try to answer to the question, which is the optimal number of clusters? We have seen in class different methods, let's try to apply the silhouette method to find the optimal number of clusters.

```
europeclust <- hclust(dist_corr_europe, method="average")
asw <- NA
clusk <- list()
sil <- list()
for (k in 2:30){
clusk[[k]] <- cutree(europeclust,k)
sil[[k]] <- silhouette(clusk[[k]], dist=dist_corr_europe)
asw[k] <- summary(silhouette(clusk[[k]],
                  dist=dist_corr_europe))$avg.width
}
plot(1:30,asw,type="l",xlab="Number of clusters",ylab="ASW",ylim=c(0.15,0.6))
```

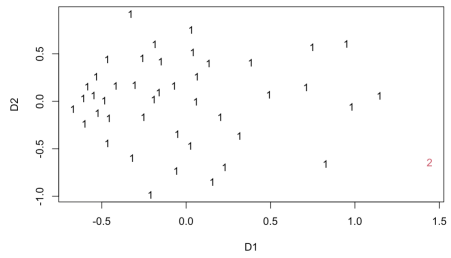
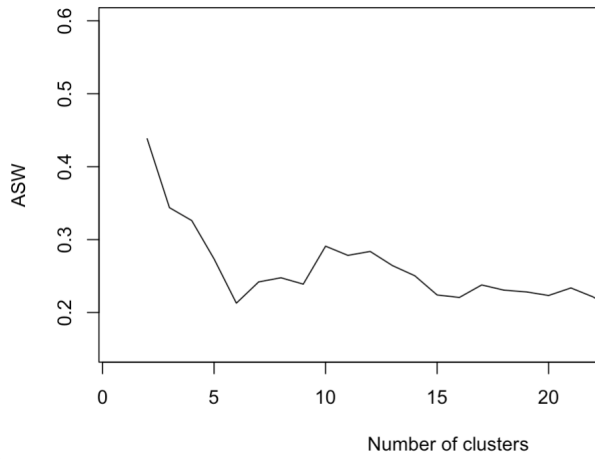


Figure 1: Enter Caption

```
plot(mds_europe_corr$conf, col=clusk[[2]], pch=clusym[clusk[[2]]])
```



We have as a maximum value of silhouette method K = 2

If we plot the mds results with the clusters created by cutting the tree when two clusters are created we get the following plot:

Since this is a repetitive task, let's write a function that performs the same analysis for all the other continents:

```r
perform_analysis <- function(data) {

  # Calculate Euclidean distance
  dist_eucl <- daisy(data, metric = "euclidean")

  # Perform MDS
  mds_result <- mds(dist_eucl, ndim = 2)

  # Plot MDS results
  plot(mds_result)
```

```
#  Perform hierarchical clustering with different linkage methods
clust_single <- hclust(dist_eucl, method = "single")
clust_average <- hclust(dist_eucl, method = "average")
clust_complete <- hclust(dist_eucl, method = "complete")

plot(clust_single)
plot(clust_average)
plot(clust_complete)

#  Calculate Pearson correlation
corr_matrix <- cor(t(data), method = "pearson")

# Calculate distance based on correlation
dist_corr <- as.dist(0.5 - (corr_matrix) / 2)

# Perform MDS on correlation-based distance
mds_corr_result <- mds(dist_corr, ndim = 2)

# Plot MDS results for correlation-based distance
plot(mds_corr_result)

# Perform hierarchical clustering on correlation-
#based distance with different linkage methods

clust_single_corr <- hclust(dist_corr, method = "single")
clust_average_corr <- hclust(dist_corr, method = "average")
clust_complete_corr <- hclust(dist_corr, method = "complete")

plot(clust_single_corr)
plot(clust_average_corr)
plot(clust_complete_corr)

clusters <- hclust(dist_corr, method="average")
asw <- NA
clusk <- list()
sil <- list()
for (k in 2:30){
clusk[[k]] <- cutree(clusters,k)
sil[[k]] <- silhouette(clusk[[k]], dist=dist_corr)
asw[k] <- summary(silhouette(clusk[[k]], dist=dist_corr))$avg.width
}
plot(1:30,asw,type="l",xlab="Number of clusters",ylab="ASW",ylim=c(0.15,0.6))

plot(mds_corr_result$conf, col=clusk[[3]], pch=clusym[clusk[[3]]])
```

```
}
```

```
    #Africa analysis
```{r}
perform_analysis(africa)
```

#South America analysis
```{r}
perform_analysis(south_america)
```

#Asia analysis
```{r}
perform_analysis(asia)
```

#Central America analysis
```{r}
perform_analysis(central_america)
```

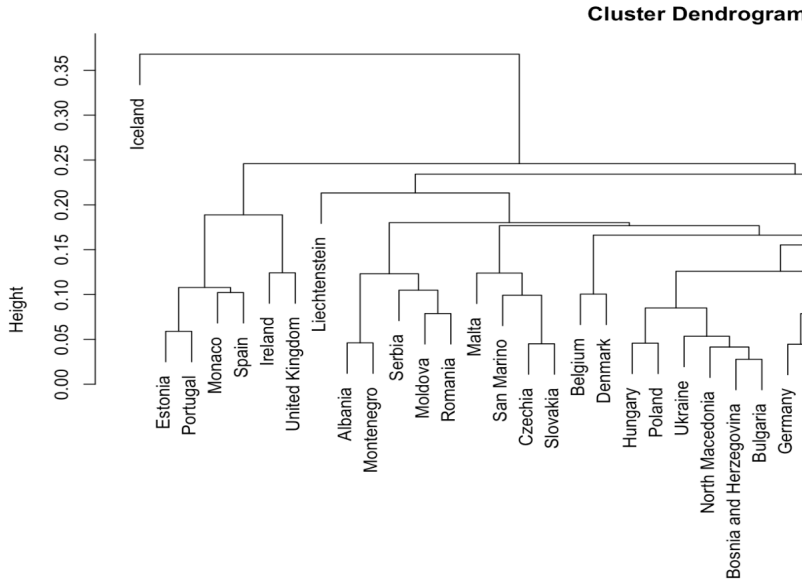
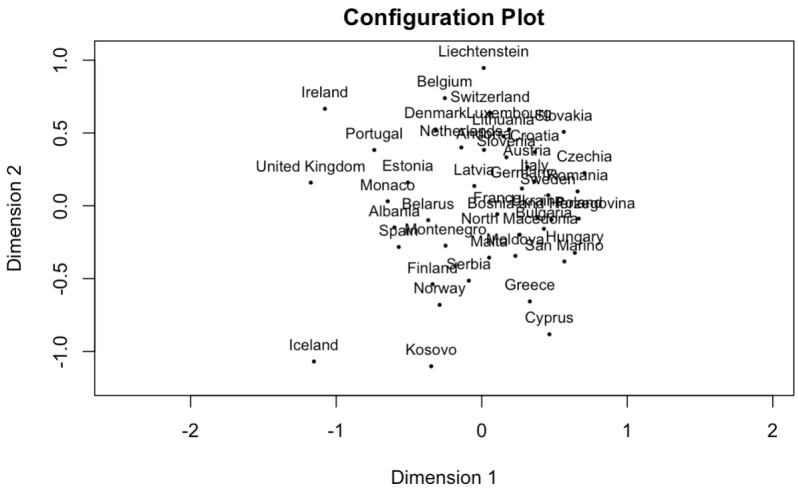
For the North America unfortunately we have just two countries: Canada and USA, therefore a proper analysis as before it's not suitable. Although those two countries can be a good example of trying a new distance based on the so called cross-correlation measure presented by John Paparizzos in the paper "k-Shape: Efficient and Accurate Clustering of Time Series".

```{r}
usa <- north_america[1,]
canada <- north_america[2,]
plot(1:555, usa, type = "l")
plot(1:555, canada, type ="l")
```

The function that compute the cross-correlation is inside the package "dtwclust"

```{r}
#install.packages("dtwclust")
library(dtwclust)
```

```{r}
shape_based_distance_northamerica <- sbd(as.numeric(usa), as.numeric(canada))
shape_based_distance_northamerica
```
```

We have as output a list of 2: a dist object which is the distance of the two time series considered, and yshift object which is the vector of distances when one time series is moved in the time axis.

Now, we can compute a new distance matrix based on the "sbd" function: let's take again Europe as an example, and then write a generic function to apply it to other continents.

```
    ```{r}
n <- nrow(europe)
matrix_sbd <- matrix(data=NA, nrow = n, ncol = n)

for (i in 1:n){
  for (j in 1:i){
    x <- europe[i,]
    y <- europe[j,]

    x <- as.numeric(x)
    y <- as.numeric(y)

    shape_based_distance_eu <- sbd(x, y)
    print(shape_based_distance_eu$dist)
    # Assign the distances directly
    matrix_sbd[i, j] <- shape_based_distance_eu$dist
    matrix_sbd[j, i] <- shape_based_distance_eu$dist
  }
}

country_names <- rownames(europe)

rownames(matrix_sbd) <- country_names
colnames(matrix_sbd) <- country_names

```

    ```{r}
clustering_europe_sbd_single <- plot(hclust(as.dist(matrix_sbd),
                                            method = "single"))
clustering_europe_sbd_average <- plot(hclust(as.dist(matrix_sbd),
                                            method = "average"))
clustering_europe_sbd_complete <-  plot(hclust(as.dist(matrix_sbd),
                                            method = "complete"))
mds_europe_sbd <- plot(mds(as.dist(matrix_sbd)))
```
```

Here in this code we perform a hierarchical clustering on the new defined distance matrix.



The results can be seen in the printed plot




```
```{r}
sbd_analysis <- function(dataframe) {
  n <- nrow(dataframe)
  matrix_sbd <- matrix(data=NA, nrow = n, ncol = n)

  for (i in 1:n){
    for (j in 1:i){
      x <- dataframe[i,]
```
```

```
      y <- dataframe[j,]

      x <- as.numeric(x)
      y <- as.numeric(y)

      shape_based_distance_eu <- sbd(x, y)

      # Assign the distances directly
      matrix_sbd[i, j] <- shape_based_distance_eu$dist
      matrix_sbd[j, i] <- shape_based_distance_eu$dist
    }
  }

  country_names <- rownames(dataframe)

  rownames(matrix_sbd) <- country_names
  colnames(matrix_sbd) <- country_names

  clustering_single <- hclust(as.dist(matrix_sbd), method = "single")
  clustering_average <- hclust(as.dist(matrix_sbd), method = "average")
  clustering_complete <-  hclust(as.dist(matrix_sbd), method = "complete")
  mds_result <- mds(as.dist(matrix_sbd), ndim = 2)

  result <- list(
    distance_matrix = matrix_sbd,
    clustering_single = clustering_single,
    clustering_average = clustering_average,
    clustering_complete = clustering_complete,
    mds = mds_result
  )

  plots <- list(
    clustering_single = plot(clustering_single),
    clustering_average = plot(clustering_average),
    clustering_complete = plot(clustering_complete),
    mds = plot(mds_result)
  )

  return(list(result = result, plots = plots))
}
```

```
```

```{r}
sbd_analysis(africa)
sbd_analysis(asia)
```

```
sbd_analysis(south_america)
sbd_analysis(central_america)
```