Problem 1

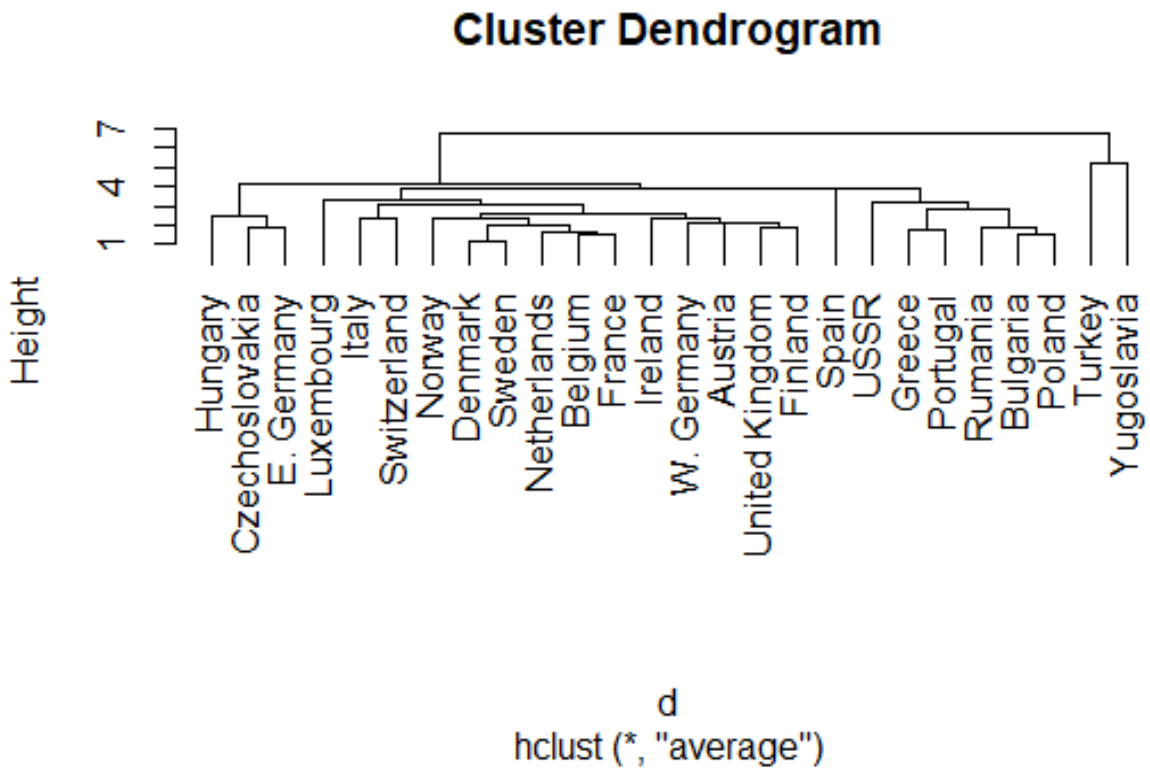
Code is in Problem1.R. The clustering results should depend on factors like economical system, which in turns depend on political system, geographical location can also be a factor sometimes.

From the plots below, as expected, single link clustering yields extended clusters, where close countries get together quickly but with inaccuracies sometimes as it just depends on the closest two points(countries) between clusters. While complete and average clustering yields to rounded clusters, where a group gets clustered to another group based on the farthest or the group decision, resulting on a more collective decision.

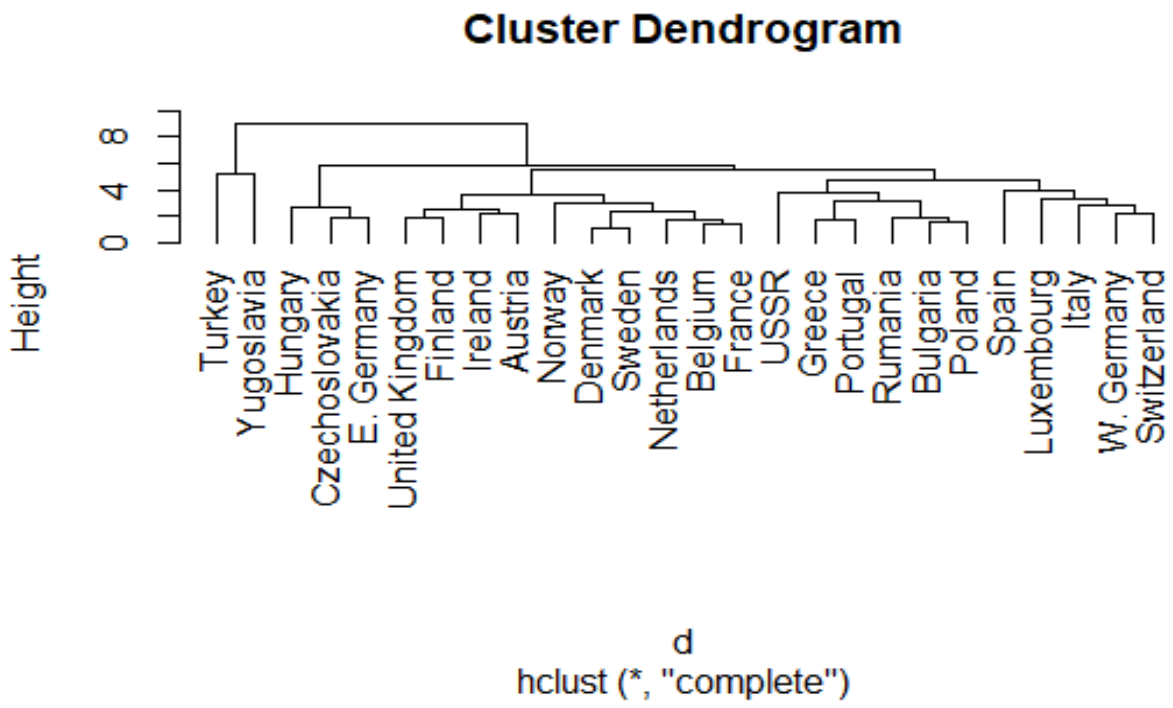
We can see countries that are politically similar are mostly grouped together, like East Germany, Hungary, and Czechoslovakia rather than East and West Germany, however, in the case of countries that are neighbors and politically similar, like France, Belgium, and the Netherlands, they are always grouped together.

The interesting distinction between Single link and other methods can be seen in the case of USSR, I would imagine it close to Czechoslovakia, E. Germany and Hungary, so a single link clustering method would bring them in one cluster after a couple of iterations as USSR is politically similar to these countries, however, in the case of average clustering, an average center distance between a group that contains Greece, Portugal and another that contains E. Germany and Hungary was hard to be brought together, and thus USSR ended up in a different cluster.

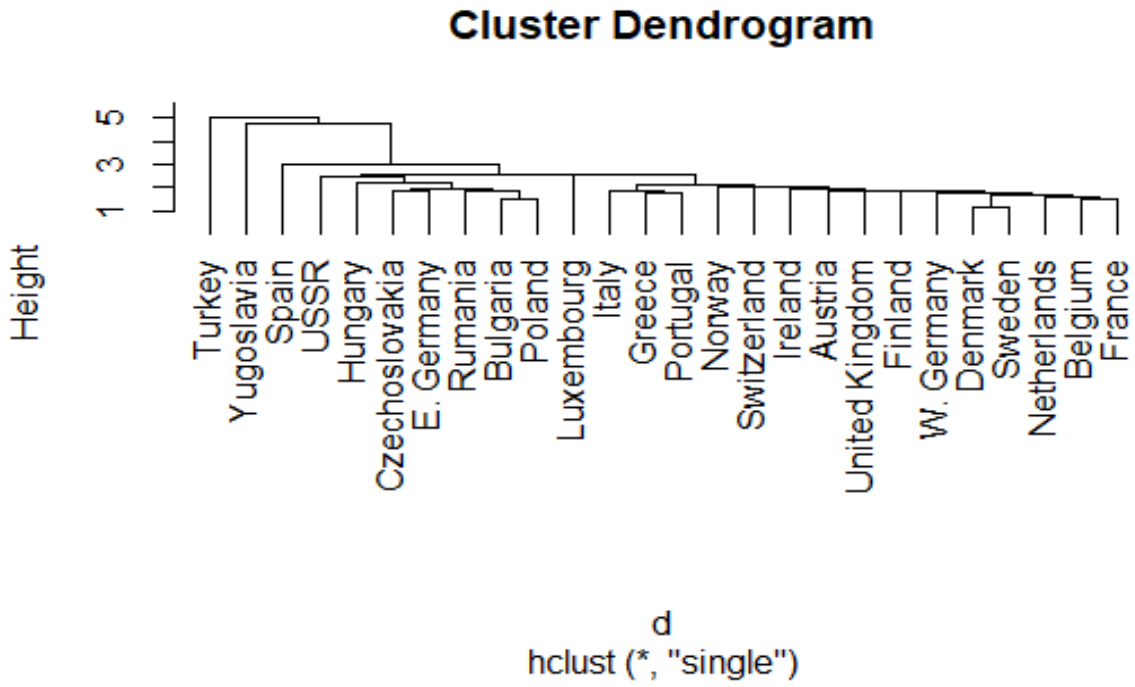
Average group clustering



Complete link clustering

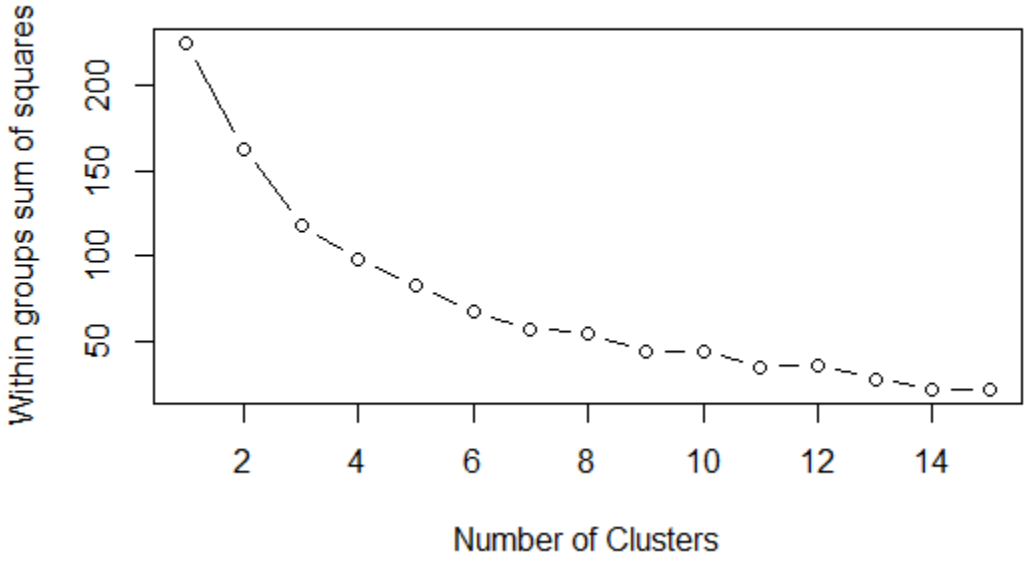


Single link clustering:



Using K-Means

From the error plot with the number of clusters, it seems to me that K = 6 is a good choice for number of clusters as the error decrease became kind of flat afterwards (knee method)



Problem 2

Code is in Problem2.R, it's split into functions, logic is as follows:

1- Read data into a list of list of files, where each file is represented as a data frame
2- Call splitData function to split data into training and testing per category with a check, if testing set was empty (could happen when category contains 3 files only), force taking one file from training set
3- Call QuantizeList for all categories (training data only) to quantize each file into signals according to the given parameters (signal height). Compile all quantize signals into one list.
4- Call kmean function over the quantized signal list
5- Featurize the training data by building a histogram for each file signals according to their proximity to the clustered centers (FeaturizeData function)
6- Call randomForest against the featurized training data to build a classification model
7- Predict the classes of the untouched testing data based on the built classification model

Part A

Using signal size equal to 32 * 3, and number of clusters equal to 400, accuracy is around 72% and thus error rate is around 28%. Confusion matrix for the 14 categories is below:

```
          Reference
Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14
        1   3  0  0  0  0  0  0  0  0  0  0  0  0  0
        2   0 17  0  2  0  0  0  0  0  0  0  0  0  5
        3   0  0  4  0  0  0  0  0  0  0  0  0  0  0
        4   0  0  0  9  0  0  0  0  0  0  0  0  0  0
        5   0  0  0  0 21  0  0  0  0  1  0  0  1  0
        6   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        7   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        8   0  1  0  0  0  0  0 12  1  0  2  1  0  0
        9   0  0  0  0  0  0  0  0  0  0  0  0  0  0
        10  0  1  0  0  0  2  1  1  1 25  2  2  2  0
        11  0  2  0  0  0  0  0  0  1  1 18  4  1  1
        12  0  1  0  1  0  0  0  7  0  0  2  9  0  1
        13  0  0  1  0  0  0  0  0  0  0  0  0  1  0
        14  0  0  0  0  0  0  0  0  0  0  0  0  0 13
```

Part B

Here are different accuracy numbers for different settings of number of clusters and signal size (fixed length sample), different settings lead to different numbers.

Setting (Signal Size/Clusters number)	Accuracy
32*3/400	72%
32*3/300	68.8%
32*3/500	72.1%
25*3/400	73.4%
25*3/300	78.3%
25*3/500	73.7%
20*3/300	78.8%