McKenzie Maidl, Samikshya Pandey, Anna Tsai
DATA 5322 Machine Learning II
Written Homework 4
2023-06-05

How Countries Spend Public Funds:
An Exploration Using Supervised and Unsupervised Techniques

Abstract

This report utilized unsupervised and supervised learning techniques to explore data from the Statistics of Public Expenditure for Economic Development database from the years 2000 and 2010. Two types of methods of unsupervised learning were used. First, *Principal Components Analysis* as well as *Singular Value Decomposition* were applied to determine the correlation between predictors and their impact on the variability of the data. Second, both *K-means Clustering* and *Hierarchical Clustering* were used for discovering groups within the data. In addition, *Decision Trees* - a supervised learning method - were created to predict the region of each country in the dataset with the expenditure variables used as predictors. This report's findings included two groups of correlated variables, with spending on agriculture, defense, transport, and communication in one and spending on education, health, social protection, and mining in the other. There were five distinct clusters of shared expenditure patterns in the data, and these remained consistent over the two years of study. In the decision trees, the two most important variables were education and healthcare.

Introduction

Governments arguably have a duty to improve the lives of their citizens and develop society, and one way of doing so is through spending public money. Understanding how governments use public funds can help determine if a government is spending money in a responsible way and help governments understand how to better spend resources and money. The goal of this paper is to study spending trends between different regions of the world to see if there are any differences or commonalities between how different regions spend money and in what sectors. Understanding the spending trend of public expenditure over the years can also help assess how spending across sectors has changed and can help governments re-prioritize their expenditure to different sectors that may require additional investments.

The dataset used in this report was sourced from the Statistics of Public Expenditure for Economic Development (SPEED) database and compiled by the International Food Policy Research Institute. The initial dataset contained 60 predictor variables related to public expenditures across ten different sectors for 147 countries from the years 1980 to 2012. The ten sectors are agriculture, communication, education, defense, health, mining, social protection, fuel and energy, transport, and transport and communication. The predictors in the original dataset were measured in several units including billions of dollars, percent of GDP, and relative purchasing power.

Both unsupervised and supervised learning methods were utilized for this report. First, several unsupervised methods were used to explore the dataset. *Principal Components Analysis* (PCA) was run to understand how the variables in the dataset contributed to its overall variability, as well as how they related to each other. *Singular Value Decomposition* with scaled data - essentially the same method as PCA - was also run to compare and plot the singular vectors. Two types of *clustering*, k-means and hierarchical, were also created to understand the groups within the data. Finally, a supervised method called *decision trees* was used to predict the region of each country in the dataset based on its expenditures. Each of these methods is described in detail in the following section.

Theoretical Background

Unsupervised learning broadly refers to the techniques used to explore data and interpret its underlying structure. Unlike supervised learning, in unsupervised learning there is no response variable (Y) for which predictions are made. Instead, unsupervised learning aims to discover general patterns or clusters in the data. It may be used alone or with supervised learning techniques, for example as part of initial exploratory data analysis. This report focuses on two categories of unsupervised learning - *principal components analysis* and *clustering* (k-means and hierarchical). In addition, a supervised learning method known as *decision trees* is run on the data with the variable 'region' as the predictor.

Principal Components Analysis

High-dimensional datasets contain a large number of features or variables (p) in proportion to the overall number of observations (n). When p is high, it is both difficult to visualize the data and unlikely that all features meaningfully contribute to its underlying structure or variability. Principal components analysis (PCA) is a process through which highly-dimensional data may be represented in a lower dimension. Its objective is to find the lowest dimension of the data that represents as much of the overall variability as possible. This provides information on which combinations of features in a dataset are most important. In addition, because PCA explores the relationships between features, it can also be used to fill in missing data.

In PCA, dimensions, also known as principal components, are found by taking linear combinations of all features ($Z = X_1 + X_2 + \dots X_p$) to find the largest normalized variance. Each feature is normalized such that their sum of squares is equal to one, which keeps variance from being arbitrarily large. The coefficients used to normalize each feature are known as loadings. The first principal component (Z_1) simply finds the linear combination of features with the largest variance, while the second principal component (Z_2) finds the same with the additional constraint that it cannot be correlated with Z_1. In practice, this constraint causes Z_1 and Z_2 to be perpendicular to each other. Subsequent components build upon previous components with similar constraints. See Figure 1 for a general representation of the first two principal components.

PCA is very similar to another technique known as Singular Value Decomposition (SVD). When the data used in SVD are scaled, the technique is the same as PCA. The two methods are run, for example in R, differently. See Appendix A for details.

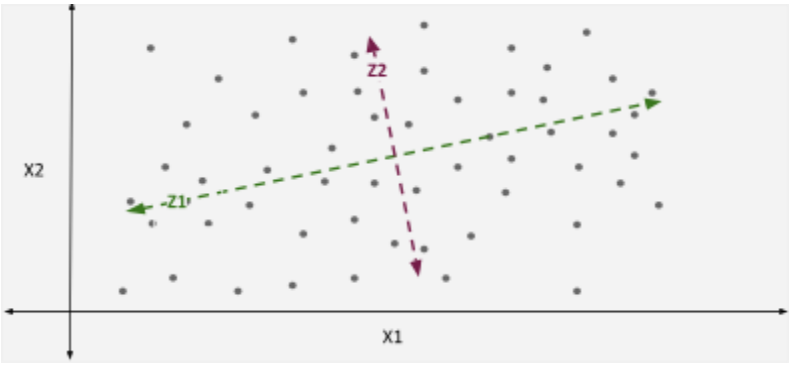


Figure 1: Graphical Example of Principal Component Analysis (Z_1 and Z_2)

Clustering

Clustering is also a type of unsupervised machine learning. Clustering creates subgroups of observations by partitioning the data into distinct groups so observations within a group are similar to one another. Clustering is a useful technique in early stages of data exploration when the goal is to create subgroups based on shared characteristics and explore how groups align. For example, clustering is useful when trying to divide data into segments for advertising and marketing purposes, or when a company wants to create a recommendation list for its users (for things like song or movie recommendations). It is also useful in the medical field where researchers are interested in shared features or characteristics. For example, if researchers are interested in creating subgroups of shared behaviors or patterns of certain cells, DNA structures or protein structures clusters can guide the initial process.

This paper utilizes two kinds of clustering methods: k-means clustering and hierarchical clustering.

K-means Clustering

The k-means clustering method partitions the observations into stated 'k' clusters. In K-means clustering, each observation belongs to one non-overlapping cluster. The main goal for each cluster is to minimize the within-cluster variations. The basic algorithm of k-means clustering follows the following pattern. Based on the stated 'k' cluster provided, each observation is randomly assigned to one cluster. The centroids are calculated and each observation that lies 'closest' to the cluster's centroid is reassigned. Closest here means the minimum euclidean distance between observations and the cluster's centroid.

Hierarchical Clustering

Hierarchical clustering organizes observations into a hierarchy of nested clusters. It starts with each observation as its own cluster and progressively merges clusters based on the similarity between observations. Dendrograms allow for visually observing how clusters are formed. It first assigns each observation into one cluster and merges the lowest point of the dendrogram based on how similar each observation is to another. So, when looking at the dendrograms, it can be inferred how close the observations are based on how close they are at the vertical axis rather than horizontal axis. Any number

of clusters can be obtained based on the horizontal cut to the diagram. In general, the aim is to choose the cut line where the division of observations is evenly split.

There are four types of trees that can be formed based on linkage. Linkage defines how to compute the distance between two sub-clusters and join observations. The four types of linkage are single, complete, average and centroid. Single linkage computes the minimum distance between the observations and builds clusters based on this distance. Complete linkage calculates the maximum distance between observations and creates clusters based on this distance. Average linkage computes the mean intercluster dissimilarity by calculating all pairwise dissimilarity between clusters and records the average of these dissimilarities. Finally, in centroid linkage calculates the dissimilarity between centroids of the cluster.

While hierarchical clustering has high interpretability and provides a simple way to decide how to split the cluster, it can perform worse than k-means clustering if the observations are not nested.

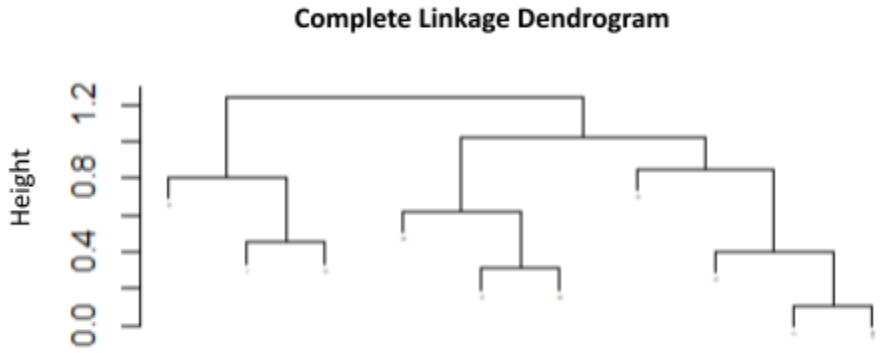


Figure 2: Complete Linkage Dendrogram

Figure 2 shows the dendrogram for a uniform distribution created randomly for 30 data points using complete linkage method. To create a different number of clusters, the tree can be cut using different heights. For example, if Figure 1 is cut at height one, two clusters are created. If the cut is at height 0.8, four clusters would be created.

Decision Trees

Decision trees are a type of supervised learning that can be applied to both regression and classification problems. Decision trees work by partitioning the data into a series of regions, represented as branches on a tree. Each node on a tree represents a condition for partitioning the data (for example color = red or color = blue). Decision trees are read top to bottom starting at the first node then moving down the tree until a node with no more possible decisions is reached (called a terminal node). Decision trees have high interpretability but can be very sensitive to the data the tree is trained on. Ways to improve the accuracy of decision trees include pruning, bagging, random forest, and boosting.

Pruning

Pruning is a process that takes a large decision tree and makes it smaller in order to reduce overfitting. Pruning starts with a single decision tree with many terminal nodes and performs cross-validation on subtrees with different numbers of terminal nodes to find the optimal number of terminal nodes.

Bagging

Bagging is one of several 'ensemble' methods that help address the limitations of decision trees. These methods make use of multiple models to create one single model. Bagging is an extension of bootstrapping where we take multiple samples from the training set data and average the predictions to create a single model. For classification models the final decision in bagging is made by 'majority vote' (James et al. 342).

Random Forests

Random forests are similar to bagging in that they make use of many samples of the data to create each tree, but instead of considering all predictors at each step, random forests limit which predictors can be considered at each step. The advantage of random forests over bagging is that by limiting the number of predictors considered it is possible to reduce the influence of strong variables that heavily influence predictions.

Boosting

Like bagging and random forests, boosting is an ensemble method. However, unlike bagging and random forests, boosting is not a bootstrap method. Rather, each sequential tree is created using information from the previous tree to improve the results.

Decision trees were chosen as the supervised learning method because of their interoperability. This paper will compare a number of models fitted to data for two different years with the goal of isolating the most important predictor variables, which is something that decision trees do well at.

Methodology

Data Preparation

The original dataset included entries for each year between 1980 and 2012. Because the goal of this project was to compare expenditures over time, the data was filtered to produce two datasets - one for the year 2000 and another for the year 2010. These years were chosen, in part, because the data for them was more complete than for earlier years. In addition, while the most recent entry in the database was from 2012, this year had more data missing than at the turn of the decade. Generally the data was more complete for countries in the Eurozone and less complete for countries outside the Eurozone.

The SPEED dataset included multiple columns for each sector but represented in different units, for example billions of dollars, percent of GDP, and relative purchasing power. To standardize measures across all countries, the variables with percent of GDP as the unit were used for this effort. All other

columns were dropped. In addition, two columns with percent of GDP as the unit - fuel and communication - were also dropped due to the majority of countries not reporting this data.

Next, rows with null values were dropped. Then to ensure the two datasets were comparable, only countries with an entry in both datasets were used. Ultimately, 12 variables (including country and region) were studied for 54 countries. The 10 numeric variables included the percent of total GDP spent on the following sectors: agriculture, education, health, defense, transport, transport and communication (grouped), social protection, mining, other sectors, and total. The countries can be seen in Table 1.

Region	Countries
East Asia/Pacific	Fiji, Philippines, Singapore, Thailand
Europe/Central Asia	Belarus, Bulgaria, Kazakhstan, Latvia, Lithuania, Romania, Ukraine
Eurozone	Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Germany, Greece, Hungary, Iceland, Ireland, Italy, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Slovakia, Slovenia, Spain, Sweden, United Kingdom
Latin America/Caribbean	Chile, El Salvador, Guatemala, Jamaica
Middle East/North Africa	Bahrain, Jordan, Kuwait, Lebanon, Oman, Tunisia, Turkey
South Asia	Nepal, Pakistan, Sri Lanka
Sub-Saharan Africa	Kenya, Lesotho, Mauritius, Namibia, South Africa

Table 1: Countries Studied, Grouped by Region

Finally, two numeric data sets were created. These included all variables except for country and region, and were used for the unsupervised learning techniques.

Principal Components Analysis

Principal components analysis was performed on the two datasets separately. For each analysis, the numeric data was first scaled to have unit variance with a mean of zero and standard deviation of one. Scaling the data is important to ensure that variables with a large mean and standard deviation do not have an outsized impact. The loadings for each principal component were then recorded, and these loadings were used to calculate the proportion and cumulative proportion of all variance explained by each principal component. These measures were then plotted to help to determine the appropriate number of principal components to use for further analysis.

In addition, Singular Value Decomposition was run on each data set. The data was first scaled for both analyses, so the method was essentially the same as the previously run PCA analyses. The function used to perform SVD in R also provides standard deviations directly.

The results for PCA and SVD results were used to inform the decision tree analysis.

Clustering

For the clustering method, similar to other methods, two datasets were used: 2000 and 2010. Only numeric datasets were used, thus dropping two columns: countries and regions. Clusters are decided with the goal of minimizing the within-cluster sum of squares (WSS), which measures how far away each centroid is from each class instance. Lower WSS indicates that observations within the cluster groups are similar to each other. In order to determine the number of clusters to use, WSS was calculated for different cluster values. Based on the clusters chosen, further analysis was done to observe if patterns about the spending behaviors and clusters could be diagnosed.

For the hierarchical clustering method, all four methods of linkage: single, complete, average and centroid were used. Following the clustering based on these processes, the cluster comparisons for 2000 and 2010 datasets were done to see how the countries placed in each cluster have changed for different clusters across these datasets.

Decision Trees

Three types of trees were fit to each dataset: a basic tree, a boosted tree, and a tree using random forests. Bagging was considered as a methodology but the 'gbm' package in R does not support bagging for multi-class classification. There are other options for bagging and multi class but bagging was omitted for simplicity. A 60/40 train/test split was used for each dataset. Because the datasets are the same size the rows in the train/test split were the same for both years.

Basic Tree: A simple decision tree using all predictors with region as the response variable was fit to both datasets using the tree function from the 'tree' library in R.

Bagging: A bagged model was fit to both datasets using all predictors and the 'region' variable as the response. The results from bagging were then used to examine variable importance.

Random Forest: A random forest model was fit to both datasets and tuned on values of 'mtry' (number of variables to be considered at each step) and 'ntree' (number of trees) as follows:

Mtry	Ntree	Error
2	100	0.682
2	250	0.682
2	500	0.682
3	100	0.682
3	250	0.682
3	500	0.682
4	100	0.682
4	250	0.682
4	500	0.682

Table 2: Random Forest Results - 2000 Data

Mtry	Ntree	Error
7	100	0.636
7	250	0.636
7	500	0.636
8	100	0.682
8	250	0.636
8	500	0.636
9	100	0.636
9	250	0.682
9	500	0.636

Table 3: Random Forest Results - 2010 Data

The error does not change based on 'mtry' or 'ntree' for 2000 which is not what is expected. Higher and lower values of 'mtry' and tree were also tried with similar results, so it is possible that there is something else going on with the data or there are more appropriate values that were not tried in the tuning process. The best results for 2010 are 'mtry' = 7 and 'ntree' = 100, 250, or 250, 'mtry' = 8 and 'ntree' = 250 or 500, and 'mtry' = 9 and 'ntree' = 100 or 500.

Computational Results

Principal Components Analysis

The proportion and cumulative proportion of variance explained are plotted for 2000 and 2010 in Figures 3 and 4, respectively. For both, it was determined that six principal components provided a reasonable amount of variance explained. After six components, the additional variance explained tapered off, and simplicity was prioritized over minor improvements.

The loadings for six principal components for each year are displayed in Table 4. Large loadings indicate a larger impact on the principal component.

Between the two years, it was determined that none of the variables had a low enough impact on the overall variability to remove from the data.

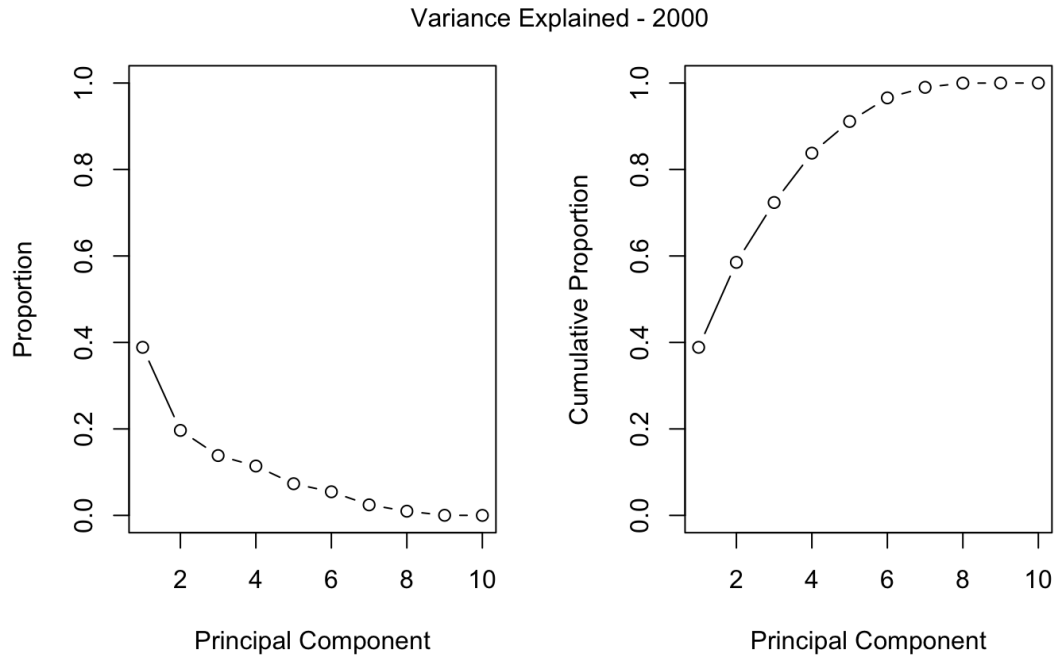


Figure 3: Results of PCA on 2000 Data

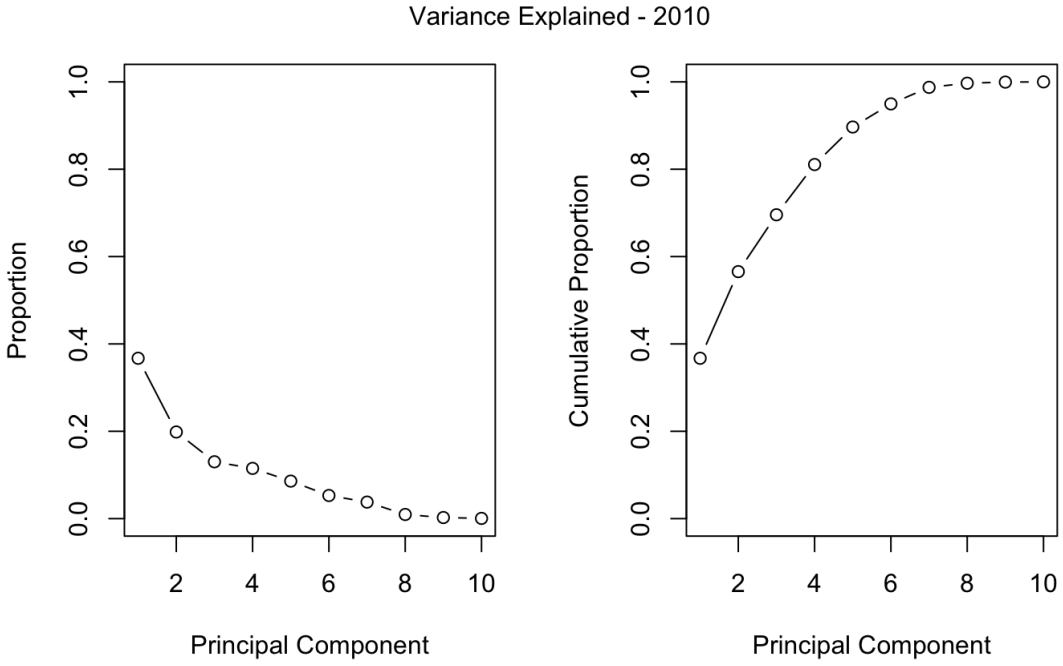


Figure 4: Results of PCA on 2010 Data

Percentage of Sector Expenditure in Total GDP	2000	2010
Agriculture	0.118	0.069
Education	-0.361	-0.323
Health	-0.031	-0.154
Defense	0.579	-0.086
Transport	0.327	0.248
Transport and Communication (grouped)	0.272	0.156
Social Protection	-0.077	-0.203
Mining	-0.332	-0.528
Other Sectors	0.419	0.668
Total	0.215	0.088

Table 4: Loadings for the Sixth Principal Component

The first and second singular vectors from the SVD analysis can be seen in Figure 5. Each point is colored by the region the country is in.

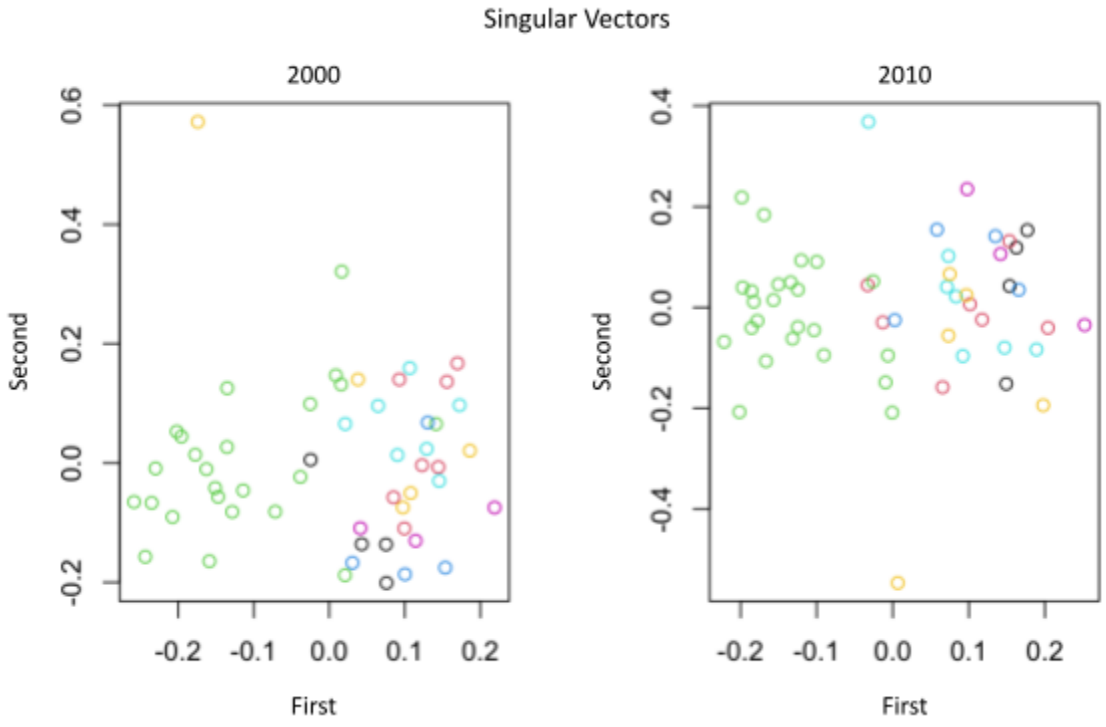


Figure 5: First Two Singular Vectors of SVD Analysis

Clustering

K-means

The WSS value stabilizes after cluster 5 for both 2000 and 2010 datasets and therefore for further analysis for both datasets, 5 clusters were used. The WSS value against different clusters value is displayed in the figure below:

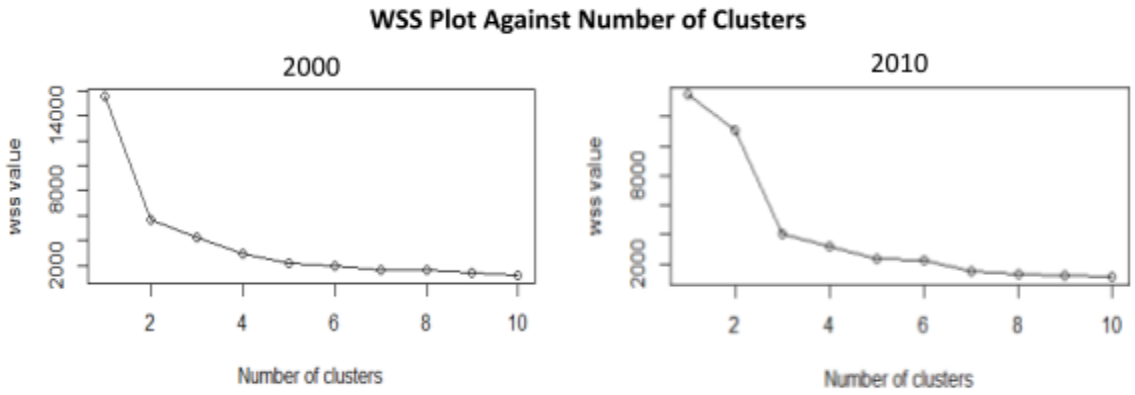


Figure 6: WSS Plot Against Number of Clusters for 2000 and 2010 dataset

Figure 7 displays the clusters indicating expenditure behaviors for the defense and transportation sectors in 2010. Five clusters were used.

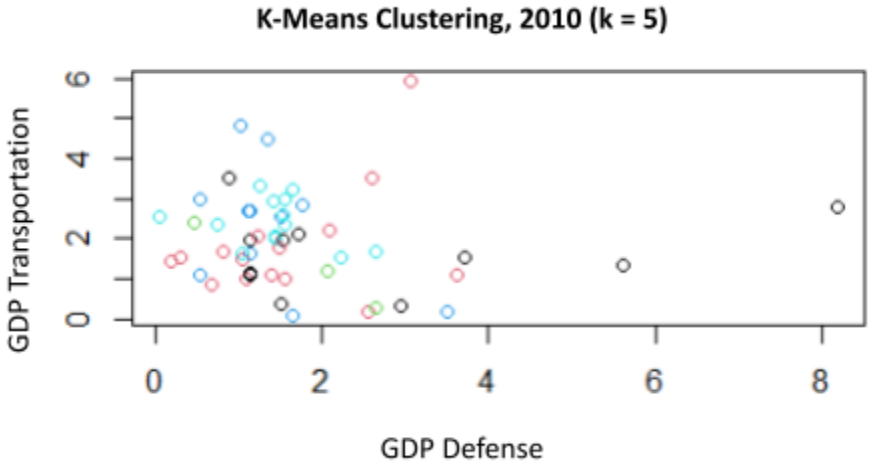


Figure 7: Clusters for Defense and Transportation, 2010

Figure 8 displays the countries in each of the five clusters for the 2010 data. Each cluster is a unit on the x-axis and shown by color, while the true region of each country is shown via the y-axis.

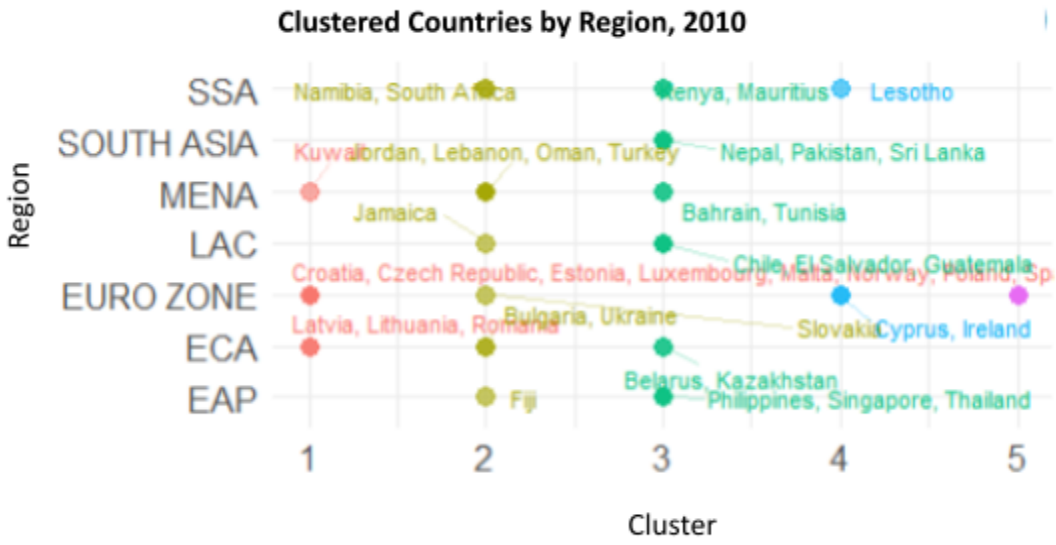


Figure 8: Clusters of countries compared to actual region for 2010 dataset

Hierarchical Clustering

For both 2000 and 2010 datasets, the dendrograms created with the complete linkage method had the best even split of the dataset and thus the complete linkage method was used.

Both years had four clusters. However, cluster division was drastically different for the 2000 and 2010 datasets as can be observed in Figure 9.

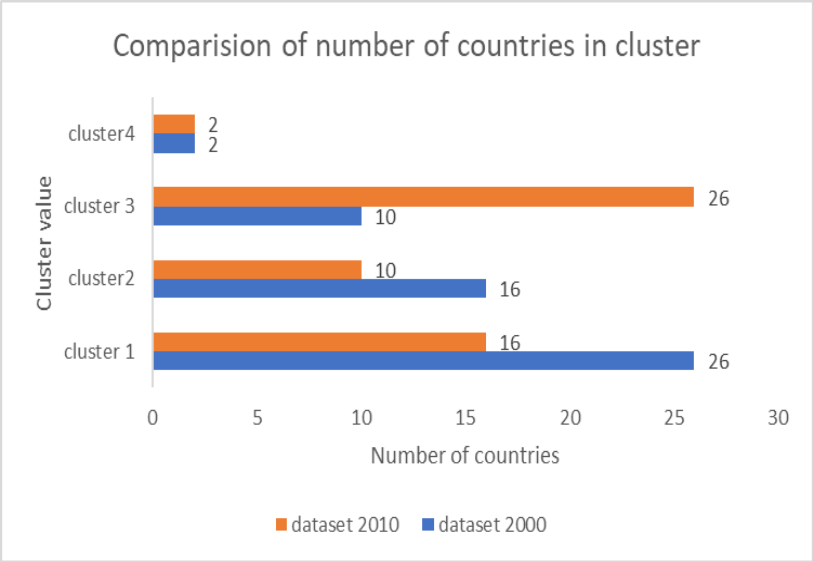


Figure 9: Number of Countries in Each Cluster, 2000 and 2010

Decision Trees

The results of the decision tree were as follows:

Basic Tree: 2000
Accuracy: 59.1%
Variables used in model: gdphealth_ppp, gdptransp_ppp, gdpdefense_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	1	0	1	1	0	0	0
ECA	0	0	0	0	0	0	0
Euro Zone	0	0	10	0	0	0	0
LAC	0	2	0	0	0	1	2
MENA	0	1	0	0	2	1	0
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 5: 2000 Basic Tree Confusion Matrix

Bagging: 2000
Accuracy: 63.6%
Most important variables: gdphealth_ppp, gdpsp_ppp, gdptrans_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	0	0	1	0	0	0	0
ECA	1	2	0	0	0	0	0
Euro Zone	0	0	10	0	1	0	0
LAC	0	0	0	1	0	1	1
MENA	0	1	0	0	1	1	1
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 6: 2000 Bagging Confusion Matrix ('mtry' = 10 and 'ntree' = 500)

Random Forest: 2000
Accuracy: 68.2%
Most important variables: gdphealth_ppp, gdpsp_ppp, gdptotal_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	1	0	1	1	0	0	0
ECA	0	0	0	0	0	0	0

Euro Zone	0	0	10	0	0	0	0
LAC	0	2	0	0	0	1	2
MENA	0	1	0	0	2	1	0
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 7: 2000 Best Random Forest Confusion Matrix

Basic Tree: 2010

Accuracy: 54.5%

Variables used in model: gdptotal_ppp, gdpeducation_ppp, gdpdefense_ppp, gdpag_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	0	0	0	0	0	0	0
ECA	1	1	0	0	0	1	1
Euro Zone	0	0	8	0	0	0	0
LAC	0	2	1	1	0	0	0
MENA	0	0	2	0	2	1	1
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 8: 2010 Basic Tree Confusion Matrix

Bagging 2010

Accuracy: 63.6%

Most important variables: gdptotal_ppp, gdpheatlh_ppp, gdpeducation_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	0	0	0	0	1	1	0
ECA	1	1	0	0	0	1	1
Euro Zone	0	1	11	0	0	0	0
LAC	0	0	0	1	0	0	0
MENA	0	1	0	0	1	0	1
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 9: 2010 Bagging Confusion Matrix ('mtry' = 10 and 'ntree' = 500)

Random Forest: 2010

Accuracy: 63.6%

Most important variables: gdphealth_ppp, gdpsp_ppp, gdptransp_ppp

	Actual						
Predicted	**EAP**	**ECA**	**Euro Zone**	**LAC**	**MENA**	**South Asia**	**SSA**
EAP	0	0	1	0	0	0	0
ECA	1	2	0	0	0	0	0
Euro Zone	0	0	10	0	1	0	0
LAC	0	0	0	1	0	1	1
MENA	0	1	0	0	1	1	1
South Asia	0	0	0	0	0	0	0
SSA	0	0	0	0	0	0	0

Table 10: 2010 Best Random Forest Confusion Matrix

Discussion

Principal Components Analysis

From the loadings in the PCA analysis both 2000 and 2010 had the highest loadings for education, transport, communication and mining. Defense had a large loading in 2000 but a smaller loading in 2010.

The most interesting finding from these analyses is the change in defense spending over the decade. It could be hypothesized that the 2008 financial crisis impacted defense spending globally, with countries turning inward to manage the crisis rather than prioritizing defense.

It is also notable from the first two singular vectors (Figure 5) that the countries in the EuroZone are grouped together, while the other countries in the dataset are more blended. As noted, countries in the EuroZone make up a disproportionate proportion of the data, and more data could lead to more accurate clustering and analysis.

Clustering

K-means Clustering

The WSS distance of the five clusters was 2281.72 in the dataset for the year 2000. Similarly, this value, also for five clusters, was 2298.92 for the 2010 data. This shows clusters have similar distribution for cluster division.

While it was difficult to build definitive patterns for spending expenditure of GDP and their cluster, some correlation could be seen in between expenditure habits. For example, clusters that had high GDP percentage expenditure for defense also had high transportation and education investments. This could mean that countries that have higher GDP value tend to have higher expenditure capabilities for education, defense and transportation as well. (Figure 7)

Looking into the clustering for 2010 dataset, the cluster divisions of countries followed similar patterns to actual economic standings of countries (GDP). For example, cluster 1 consisted of mainly Western European countries indicating their public expenditure patterns follow similar patterns. Countries in cluster 1 also fell mainly under EuroZone regions. Similarly, cluster 2 consisted of mainly developing nations and included countries from South Asia and ECA regions. While there were discrepancies for MENA countries (included in clusters 1, 2, and 3) and SSA countries (included in clusters 2, 3, and 4), this reflects different economies within these regions (Figure 8). Clusterings for the 2000 dataset followed similar patterns as well. This indicates that behaviors of countries within a cluster did not have any drastic changes between the year 2000 and 2010.

Hierarchical Clustering

As noted, hierarchical clustering using the complete-linkage method resulted in four clusters for each dataset, 2000 and 2010. However it was difficult to find patterns for differences in clustering for these two datasets and how countries were separated for each cluster. This could be because hierarchical clustering performs better when data and observations are nested and while expenditure habits of the countries could follow patterns, they are not nested. Therefore, the k-means clustering performed better for this dataset compared to hierarchical clustering.

Decision Trees

The decision tree models performed slightly better for the 2010 data than for the 2000 data, but performance was similar between the two. In 2000 health expenditure, transportation expenditure, and social protection expenditure showed up as important variables in multiple models. In 2010 total expenditure, education expenditure, and health all showed up as important variables in multiple models. These variables might make sense as countries that have higher GDPmight have more money to spend on healthcare and education as opposed to infrastructure or other more 'basic' needs. The decision trees were generally best at accurately classifying countries in the EuroZone, and worst at classifying countries in Sub-Saharan Africa (SSA) and South Asia. This could be because there is much more data for EuroZone countries than other regions.

Similar to the PCA results, education and healthcare were top predictors for the decision trees, but unlike PCA agriculture did not show up as a top predictor for most of the models. However, the decision trees did not perform overwhelmingly well and it is possible that limiting the predictors considered to just the top PCA components could increase the accuracy.

Similar to clustering the decision trees tend to classify Euro Zone countries the same- with a few exceptions. For example, in 2000 Croatia was incorrectly predicted as EAP instead of Euro Zone and Jordan was incorrectly predicted as Euro Zone instead of MENA. These two countries have similar GDPs but Croatia is much smaller in population.

Conclusion

Because governments have so much spending power and influence over their citizens' lives, it is important to understand how they are spending money. If governments in one region are spending money in certain sectors more than in other regions it could indicate differing priorities, or an imbalance in what governments need to spend on. For example, countries with higher GDPs might feel more free to spend more on defense instead of social services like healthcare. In contrast, developing nations might be burdened by higher infrastructure costs for example.

The report found clear patterns in how groups of countries prioritize their spending, with some sectors consistently having higher expenditures than others. It also found that the groups of countries in each cluster, with similar expenditure patterns, remained consistent over time. Finally the proportion of spending on sectors like education and healthcare was a determining factor in what group a country fell into.

All this indicates that regions of the world have drastically different priorities when it comes to expenditure spending. An important follow-up question, then, is what impact these spending decisions have on the lives of the people impacted by them.

References

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (1st ed.) [PDF]. Springer.

Appendix A: Code

Homework 4

McKenzie Maidl, Samikshya Pandey, Anna Tsai

2023-06-05

Import Libraries

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0     ✔ purrr   1.0.0
## ✔ tibble  3.2.1     ✔ dplyr   1.1.2
## ✔ tidyr   1.2.1     ✔ stringr 1.5.0
## ✔ readr   2.1.3     ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ggplot2)
library(ggrepel)
```

Load Data

```
load("Data/data2000.RData")
load("Data/data2010.RData")

df2000 <- data2000
df2010 <- data2010
```

Unsupervised Learning:

PCA

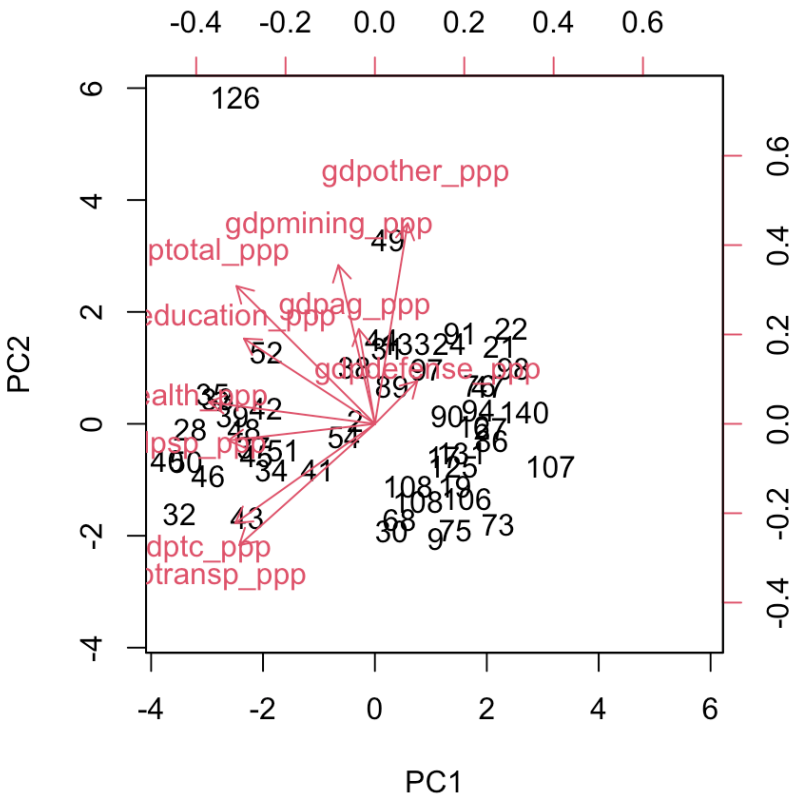
Remove non-numeric columns:

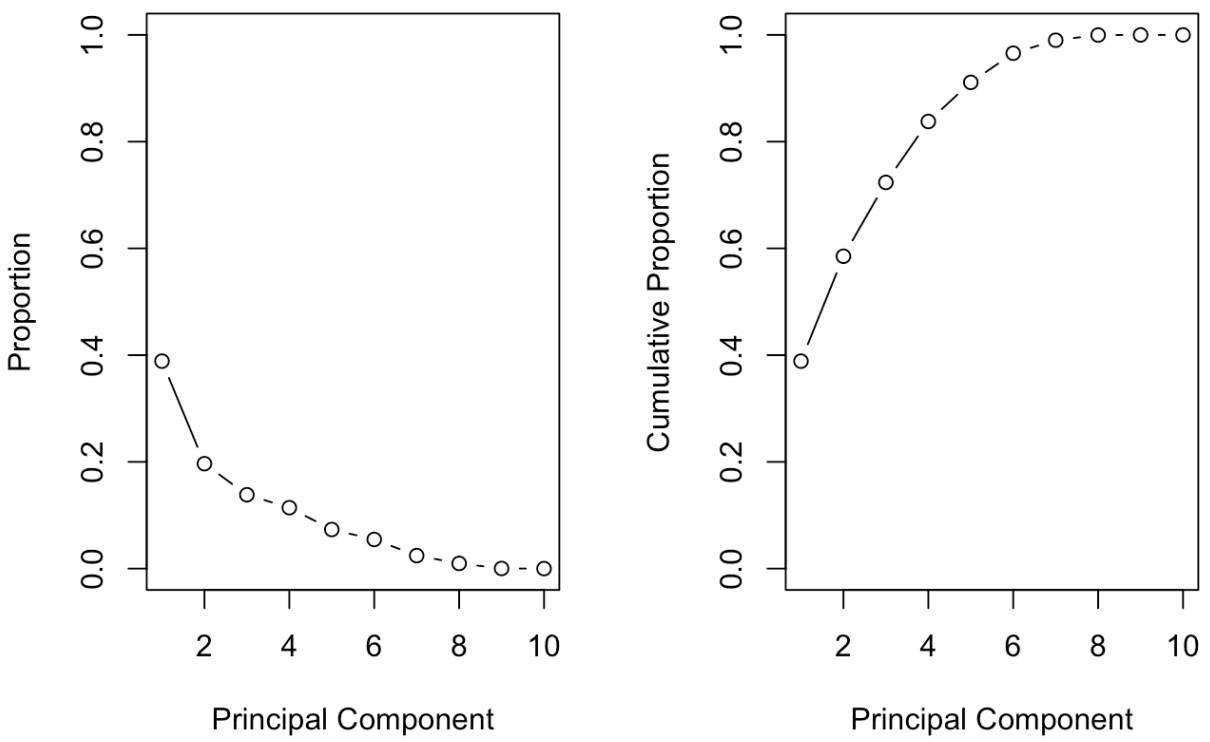
```
df2000num <- subset(data2000, select = -c(country, region))
df2010num <- subset(data2010, select = -c(country, region))
```

```
# Function to perform PCA on each data set (2000 and 2010)
perform_pca <- function(df, year) {

  # perform PCA (scale the data)
  pr.out <- prcomp(df, scale = TRUE)

  # get loadings of each principal component
  print(data.frame(pr.out$rotation))

  # plot the first two principal components (arrows scaled to represent loadings)
  biplot(pr.out, scale = 0)

  # get variance explained by each principal component
  pr.var <- pr.out$sdev^2     # variance
  pve <- pr.var / sum(pr.var) # proportion of total variance

  # plot these
  par(mfrow = c(1, 2))
  plot(pve, xlab = "Principal Component", ylab = "Proportion", ylim = c(0, 1), type = "b")
  plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion", ylim = c
(0, 1), type = "b")
  mtext(paste("Variance Explained", "-", year), side = 3, line = -1, outer = TRUE)
}
```

```
# 2000 data
perform_pca(df2000num, "2000")
```

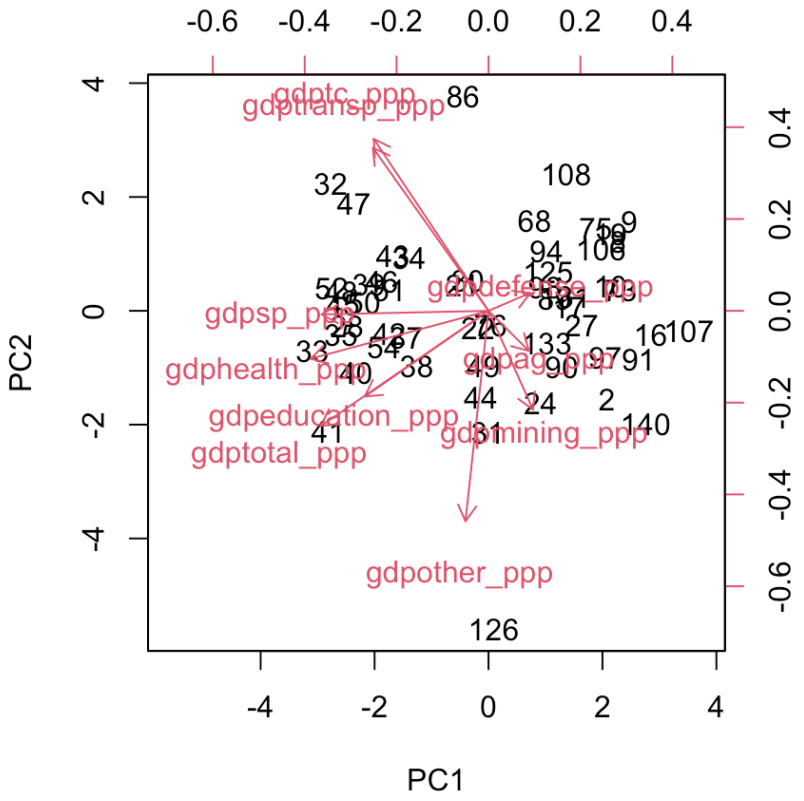
```
##                          PC1         PC2         PC3         PC4         PC5
## gdpag_ppp         -0.04495740  0.26508496 -0.62247186 -0.03819963 -0.60911264
## gdpeducation_ppp -0.36648794  0.23850465  0.01045602 -0.35559938 -0.29368451
## gdphealth_ppp     -0.46441588  0.06037977  0.18624913  0.11573209 -0.17055446
## gdpdefense_ppp     0.11850962  0.11975898  0.41119012 -0.64888879 -0.15183180
## gdptransp_ppp     -0.37959184 -0.33944587 -0.28852887 -0.08423308  0.19778519
## gdptc_ppp         -0.39074022 -0.27769830 -0.32011059 -0.14613223  0.24976475
## gdpsp_ppp         -0.41019813 -0.04666363  0.37306967  0.22878972 -0.04678655
## gdpmining_ppp     -0.10228976  0.44388938 -0.18031629 -0.41332852  0.57446898
## gdpother_ppp       0.09156904  0.56104484 -0.11242831  0.38632053  0.23803651
## gdptotal_ppp      -0.38737619  0.38539738  0.19646536  0.19173081  0.01493660
##                          PC6         PC7         PC8          PC9          PC10
## gdpag_ppp          0.11789265 -0.38762340 -0.04352326 -0.006204541 -0.040948807
## gdpeducation_ppp -0.36092431  0.54760355  0.38224770 -0.035202431 -0.136024846
## gdphealth_ppp     -0.03113686  0.15027577 -0.81009001  0.007824047 -0.153757440
## gdpdefense_ppp     0.57916467 -0.09704796 -0.04509606 -0.001069859 -0.107431693
## gdptransp_ppp      0.32735223  0.07574327  0.06636720 -0.702788522  0.003255406
## gdptc_ppp          0.27225030  0.06099062  0.07574096  0.704650825 -0.062356506
## gdpsp_ppp         -0.07664327 -0.54627616  0.34844258 -0.013448827 -0.458214322
## gdpmining_ppp     -0.33227076 -0.33314540 -0.17176220 -0.080991720 -0.040967746
## gdpother_ppp       0.41916625  0.30424183  0.08958530 -0.019078166 -0.426199253
## gdptotal_ppp       0.21488517 -0.08033582  0.15694469  0.033330693  0.739901387
```



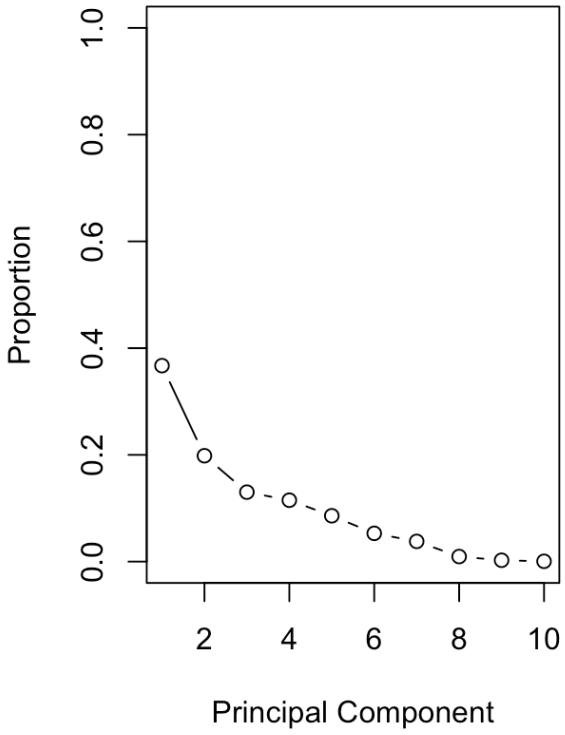


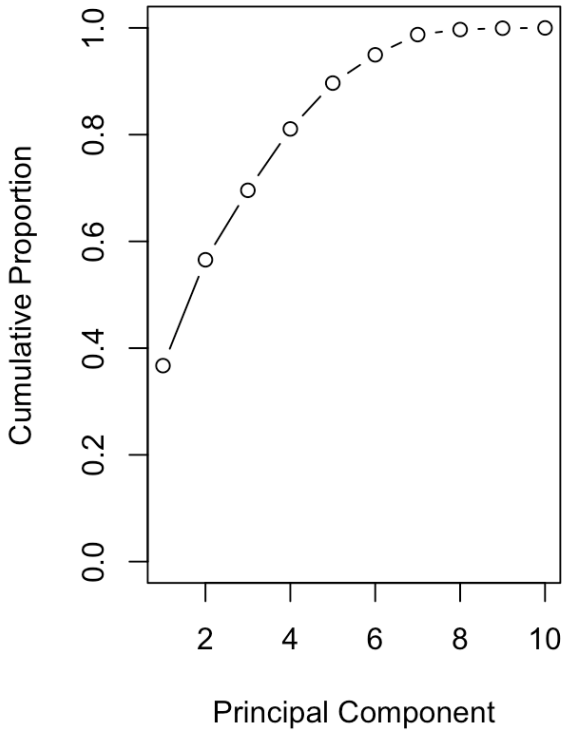

```
# 2010 data
perform_pca(df2010num, "2010")
```

```
##                         PC1          PC2         PC3         PC4          PC5
## gdpag_ppp          0.10972372 -0.109055378 -0.49515728 -0.53607576 -0.488100326
## gdpeducation_ppp -0.33615675 -0.233101618 -0.28217704  0.03937934 -0.419982856
## gdphealth_ppp    -0.48364466 -0.130447266  0.11924545 -0.07042513 -0.004853905
## gdpdefense_ppp     0.11328947  0.048522176 -0.01703001  0.76996505 -0.501969310
## gdptransp_ppp    -0.31428501  0.443532812 -0.36905500  0.10450852  0.116526791
## gdptc_ppp        -0.31225694  0.468079548 -0.35082600  0.09966103  0.103690849
## gdpsp_ppp        -0.45355101 -0.009834882  0.31282093 -0.11325888  0.051083264
## gdpmining_ppp      0.12107358 -0.268779543 -0.52527126  0.18089371  0.532300599
## gdpother_ppp     -0.06275342 -0.572844149 -0.16438885  0.19084144  0.151689024
## gdptotal_ppp     -0.45566204 -0.313193654  0.02479692  0.10218248 -0.009080241
##                         PC6         PC7          PC8         PC9         PC10
## gdpag_ppp          0.06921543 -0.44590944  0.004983596 -0.02502362 -0.031475954
## gdpeducation_ppp -0.32272039  0.65519254 -0.163900671  0.02357205 -0.119403864
## gdphealth_ppp    -0.15386746 -0.13394583  0.819996397 -0.02997951 -0.122307286
## gdpdefense_ppp   -0.08630724 -0.34538418  0.052136691 -0.03617766 -0.094527669
## gdptransp_ppp      0.24813615  0.04784193 -0.022213544 -0.69242023  0.001356799
## gdptc_ppp          0.15566487 -0.04686195 -0.010762451  0.71381631 -0.059640566
## gdpsp_ppp        -0.20279143 -0.37504180 -0.499471102 -0.05919904 -0.496129137
## gdpmining_ppp    -0.52802147 -0.19891995 -0.014730401 -0.03284184 -0.038091906
## gdpother_ppp       0.66813303  0.03841706 -0.007737304  0.04511998 -0.362608037
## gdptotal_ppp       0.08832194 -0.22015797 -0.218270253  0.03181431  0.760407054
```



Variance Explained - 2010





SVD

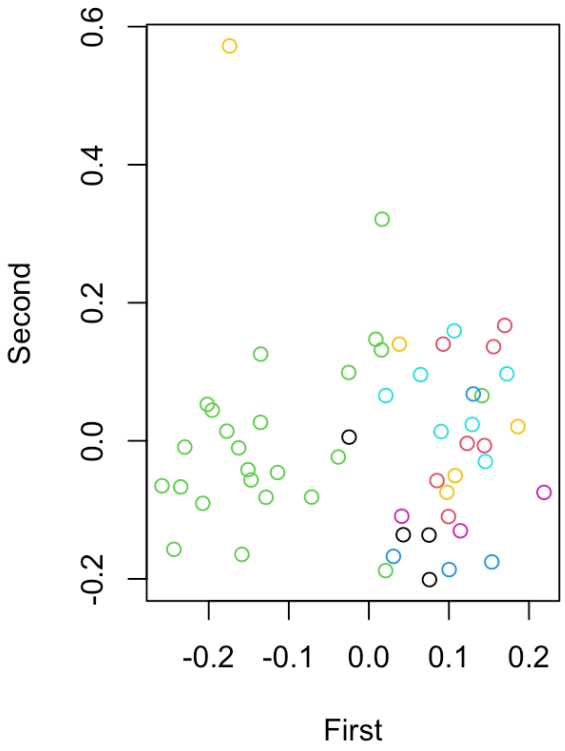
```
# 2000
dfscale00 <-  scale(df2000num) # scale data
s00 <- svd(dfscale00)          # perform SVD
data.frame(s00$v)              # loadings (same as PCA)
```

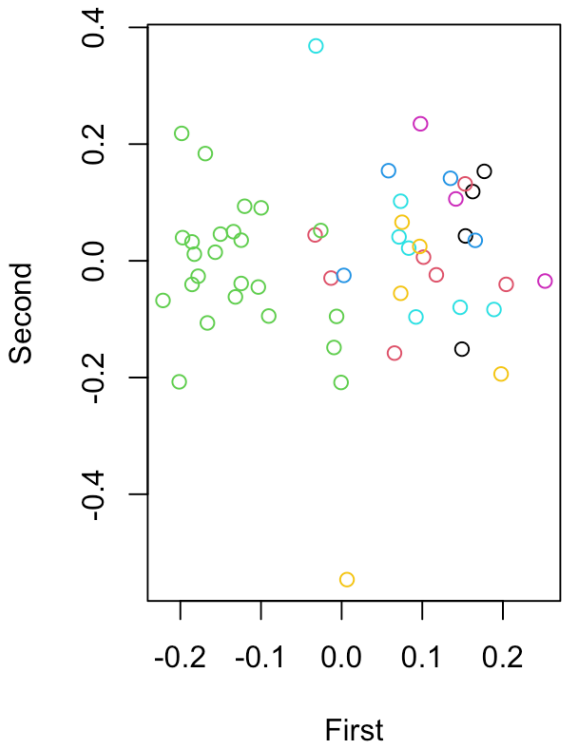
```
##             X1          X2          X3          X4          X5          X6
## 1  -0.04495740  0.26508496 -0.62247186 -0.03819963 -0.60911264  0.11789265
## 2  -0.36648794  0.23850465  0.01045602 -0.35559938 -0.29368451 -0.36092431
## 3  -0.46441588  0.06037977  0.18624913  0.11573209 -0.17055446 -0.03113686
## 4   0.11850962  0.11975898  0.41119012 -0.64888879 -0.15183180  0.57916467
## 5  -0.37959184 -0.33944587 -0.28852887 -0.08423308  0.19778519  0.32735223
## 6  -0.39074022 -0.27769830 -0.32011059 -0.14613223  0.24976475  0.27225030
## 7  -0.41019813 -0.04666363  0.37306967  0.22878972 -0.04678655 -0.07664327
## 8  -0.10228976  0.44388938 -0.18031629 -0.41332852  0.57446898 -0.33227076
## 9   0.09156904  0.56104484 -0.11242831  0.38632053  0.23803651  0.41916625
## 10 -0.38737619  0.38539738  0.19646536  0.19173081  0.01493660  0.21488517
##             X7          X8           X9          X10
## 1  -0.38762340 -0.04352326 -0.006204541 -0.040948807
## 2   0.54760355  0.38224770 -0.035202431 -0.136024846
## 3   0.15027577 -0.81009001  0.007824047 -0.153757440
## 4  -0.09704796 -0.04509606 -0.001069859 -0.107431693
## 5   0.07574327  0.06636720 -0.702788522  0.003255406
## 6   0.06099062  0.07574096  0.704650825 -0.062356506
## 7  -0.54627616  0.34844258 -0.013448827 -0.458214322
## 8  -0.33314540 -0.17176220 -0.080991720 -0.040967746
## 9   0.30424183  0.08958530 -0.019078166 -0.426199253
## 10 -0.08033582  0.15694469  0.033330693  0.739901387
```

```
# 2010
dfscale10 <-  scale(df2010num) # scale data
s10 <- svd(dfscale10)          # perform SVD
data.frame(s10$v)              # loadings (same as PCA)
```

```
##             X1           X2          X3          X4           X5          X6
## 1   0.10972372 -0.109055378 -0.49515728 -0.53607576 -0.488100326  0.06921543
## 2  -0.33615675 -0.233101618 -0.28217704  0.03937934 -0.419982856 -0.32272039
## 3  -0.48364466 -0.130447266  0.11924545 -0.07042513 -0.004853905 -0.15386746
## 4   0.11328947  0.048522176 -0.01703001  0.76996505 -0.501969310 -0.08630724
## 5  -0.31428501  0.443532812 -0.36905500  0.10450852  0.116526791  0.24813615
## 6  -0.31225694  0.468079548 -0.35082600  0.09966103  0.103690849  0.15566487
## 7  -0.45355101 -0.009834882  0.31282093 -0.11325888  0.051083264 -0.20279143
## 8   0.12107358 -0.268779543 -0.52527126  0.18089371  0.532300599 -0.52802147
## 9  -0.06275342 -0.572844149 -0.16438885  0.19084144  0.151689024  0.66813303
## 10 -0.45566204 -0.313193654  0.02479692  0.10218248 -0.009080241  0.08832194
##             X7           X8          X9          X10
## 1  -0.44590944  0.004983596 -0.02502362 -0.031475954
## 2   0.65519254 -0.163900671  0.02357205 -0.119403864
## 3  -0.13394583  0.819996397 -0.02997951 -0.122307286
## 4  -0.34538418  0.052136691 -0.03617766 -0.094527669
## 5   0.04784193 -0.022213544 -0.69242023  0.001356799
## 6  -0.04686195 -0.010762451  0.71381631 -0.059640566
## 7  -0.37504180 -0.499471102 -0.05919904 -0.496129137
## 8  -0.19891995 -0.014730401 -0.03284184 -0.038091906
## 9   0.03841706 -0.007737304  0.04511998 -0.362608037
## 10 -0.22015797 -0.218270253  0.03181431  0.760407054
```

```
# first two singular vectors
par(mfrow = c(1, 2))
plot(s00$u[,1], s00$u[,2], col = as.factor(data2000$region), xlab = "First", ylab = "Secon
d")
plot(s10$u[,1], s10$u[,2], col = as.factor(data2010$region), xlab = "First", ylab = "Secon
d")
```



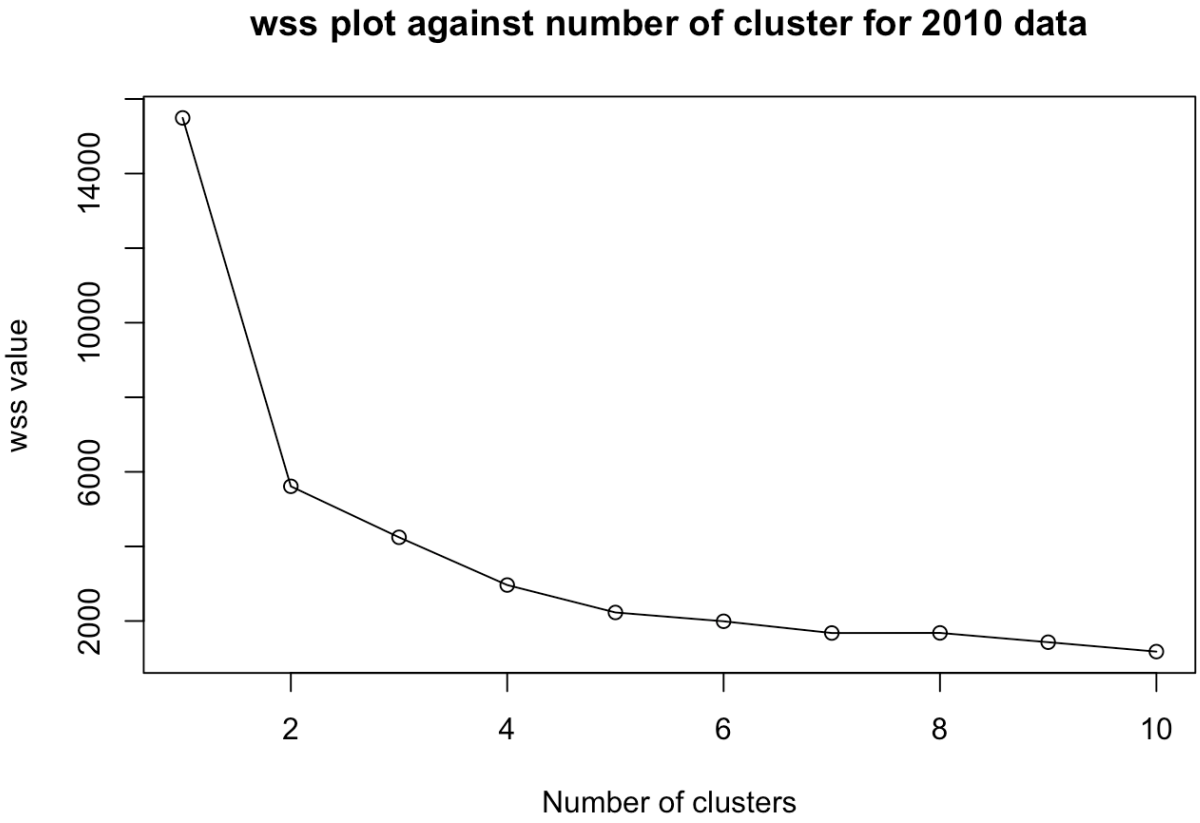


###Clustering

K-means Clustering- 2010

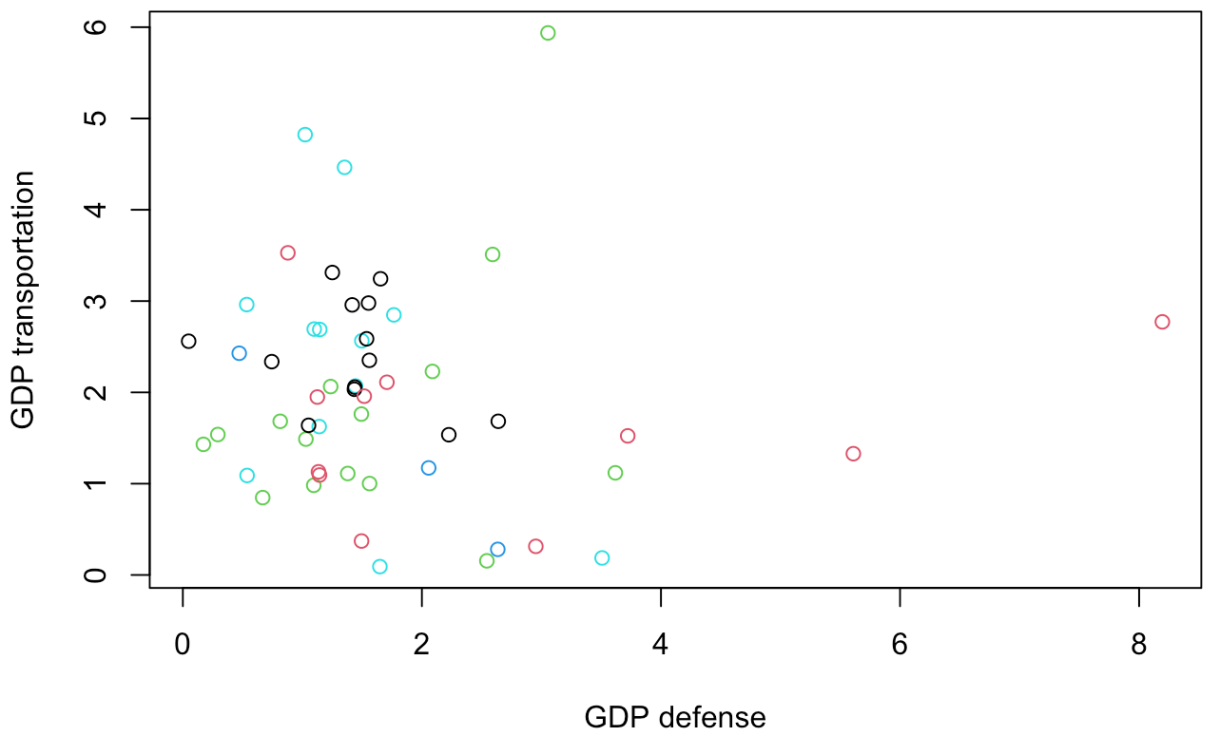
Wss means against k cluster to decide how many cluster to use for analysis

```
set.seed(1)
wss<- NULL
for (i in 1:10){
  fit = kmeans(df2010num,centers = i)
  wss = c(wss, fit$tot.withinss)
}
plot(1:10, wss, type = "o", main = 'wss plot against number of cluster for 2010 data', xla
b = "Number of clusters", ylab = "wss value")
```



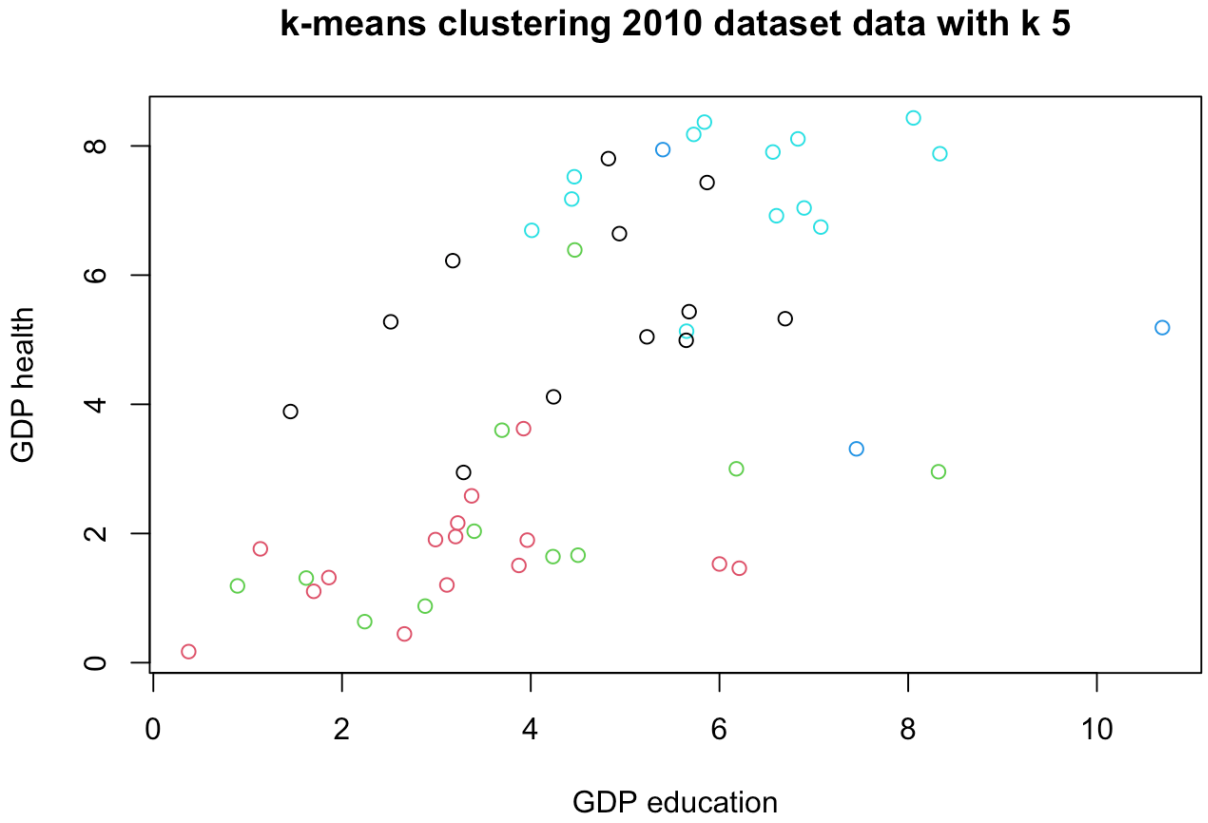
Use 5 clusters for both as the wss is reasonable at that point before dropping off. Since wss will keep on dropping till each observation has its own cluster; 5 clusters seems a reasonable wss distance

```
km_df2010num <- kmeans(df2010num, 5, nstart = 20) #iteration for initializing is 20

plot(df2010num[, c("gdpdefense_ppp", "gdptransp_ppp")],
     col = km_df2010num$cluster,
     main = paste("k-means clustering 2010 data with k 5"),
     xlab = "GDP defense", ylab = "GDP transportation")
```




```
    #xlim = c(min(df2010$country), max(df2010$country)))
```

```
km_df2010num <- kmeans(df2010num, 5, nstart = 20) #iteration for initializing is 20

plot(df2010num[, c("gdpeducation_ppp", "gdphealth_ppp")],
     col = km_df2010num$cluster,
     main = paste("k-means clustering 2010 dataset data with k 5"),
     xlab = "GDP education", ylab = "GDP health")
```



wss distance:

```
km_df2010num$tot.withinss
```

```
## [1] 2228.917
```

The within cluster distance is 2228.18.

```
country_cluster2010 <- data.frame(country = data2010$country, cluster = km_df2010num$clust
er)

# Print the data frame
print(country_cluster2010)
```

```
##            country cluster
## 2             Fiji       3
## 8      Philippines       2
## 9        Singapore       2
## 10        Thailand       2
## 16         Belarus       2
## 17        Bulgaria       3
## 19      Kazakhstan       2
## 21          Latvia       1
## 22       Lithuania       1
## 24         Romania       1
## 27         Ukraine       3
## 28         Austria       5
## 30         Croatia       1
## 31          Cyprus       4
## 32  Czech Republic       1
## 33         Denmark       5
## 34         Estonia       1
## 35         Finland       5
## 37         Germany       5
## 38          Greece       5
## 39         Hungary       5
## 40         Iceland       5
## 41         Ireland       4
## 42           Italy       5
## 43      Luxembourg       1
## 44           Malta       1
## 45     Netherlands       5
## 46          Norway       1
## 47          Poland       1
## 48        Portugal       5
## 49        Slovakia       3
## 50        Slovenia       5
## 51           Spain       1
## 52          Sweden       5
## 54  United Kingdom       5
## 68           Chile       2
## 73     El Salvador       2
## 75       Guatemala       2
## 76         Jamaica       3
## 86         Bahrain       2
## 89          Jordan       3
## 90          Kuwait       1
## 91         Lebanon       3
## 94            Oman       3
## 97         Tunisia       2
## 98          Turkey       3
## 106          Nepal       2
## 107       Pakistan       2
## 108      Sri Lanka       2
## 125          Kenya       2
## 126        Lesotho       4
## 131      Mauritius       2
```

```
## 133        Namibia       3
## 140   South Africa       3
```

creating dataframe with country's name and cluster for regional comparision and see which countries fell into what cluster.

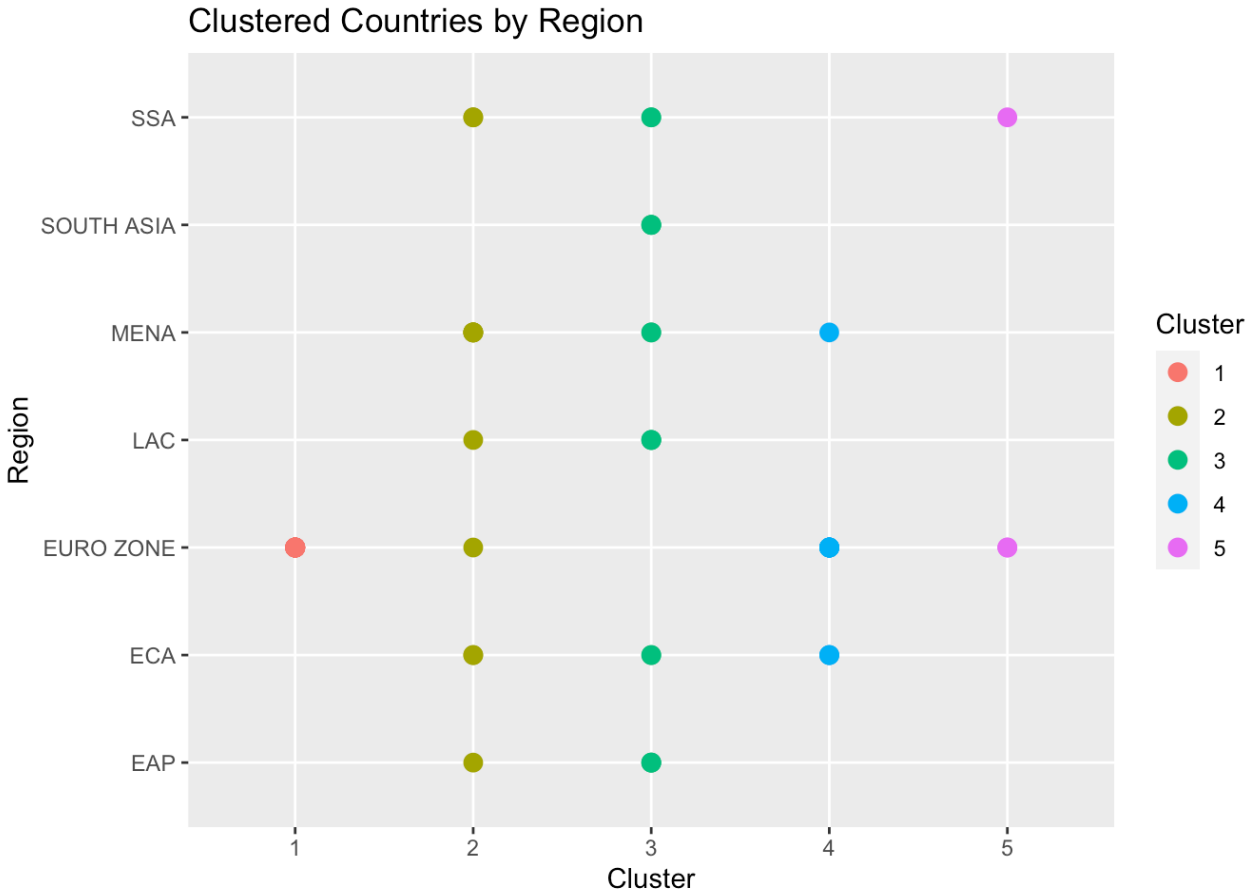
```
country_cluster <- data.frame(country = data2010$country, cluster = km_df2010num$cluster)

# Group the data frame by cluster and list the countries in each cluster
grouped_countries <- country_cluster %>%
  group_by(cluster) %>%
  summarise(countries = paste(country, collapse = ", "))

print(grouped_countries)
```

```
## # A tibble: 5 × 2
##   cluster countries
##     <int> <chr>
## 1       1 Latvia, Lithuania, Romania, Croatia, Czech Republic, Estonia, Luxembo…
## 2       2 Philippines, Singapore, Thailand, Belarus, Kazakhstan, Chile, El Salv…
## 3       3 Fiji, Bulgaria, Ukraine, Slovakia, Jamaica, Jordan, Lebanon, Oman, Tu…
## 4       4 Cyprus, Ireland, Lesotho
## 5       5 Austria, Denmark, Finland, Germany, Greece, Hungary, Iceland, Italy, …
```

```
region_cluster <- data.frame(country = data2010$region, cluster = km_df2010num$cluster)

# Group the data frame by cluster and list the countries in each cluster
grouped_region <- region_cluster %>%
  group_by(cluster) %>%
  summarise(countries = paste(country, collapse = ", "))

print(grouped_region)
```

```
## # A tibble: 5 × 2
##   cluster countries
##     <int> <chr>
## 1       1 ECA, ECA, ECA, EURO ZONE, EURO ZONE, EURO ZONE, EURO ZONE, EURO ZONE,…
## 2       2 EAP, EAP, EAP, ECA, ECA, LAC, LAC, LAC, MENA, MENA, SOUTH ASIA, SOUTH…
## 3       3 EAP, ECA, ECA, EURO ZONE, LAC, MENA, MENA, MENA, MENA, SSA, SSA
## 4       4 EURO ZONE, EURO ZONE, SSA
## 5       5 EURO ZONE, EURO ZONE, EURO ZONE, EURO ZONE, EURO ZONE, EURO ZONE, EUR…
```

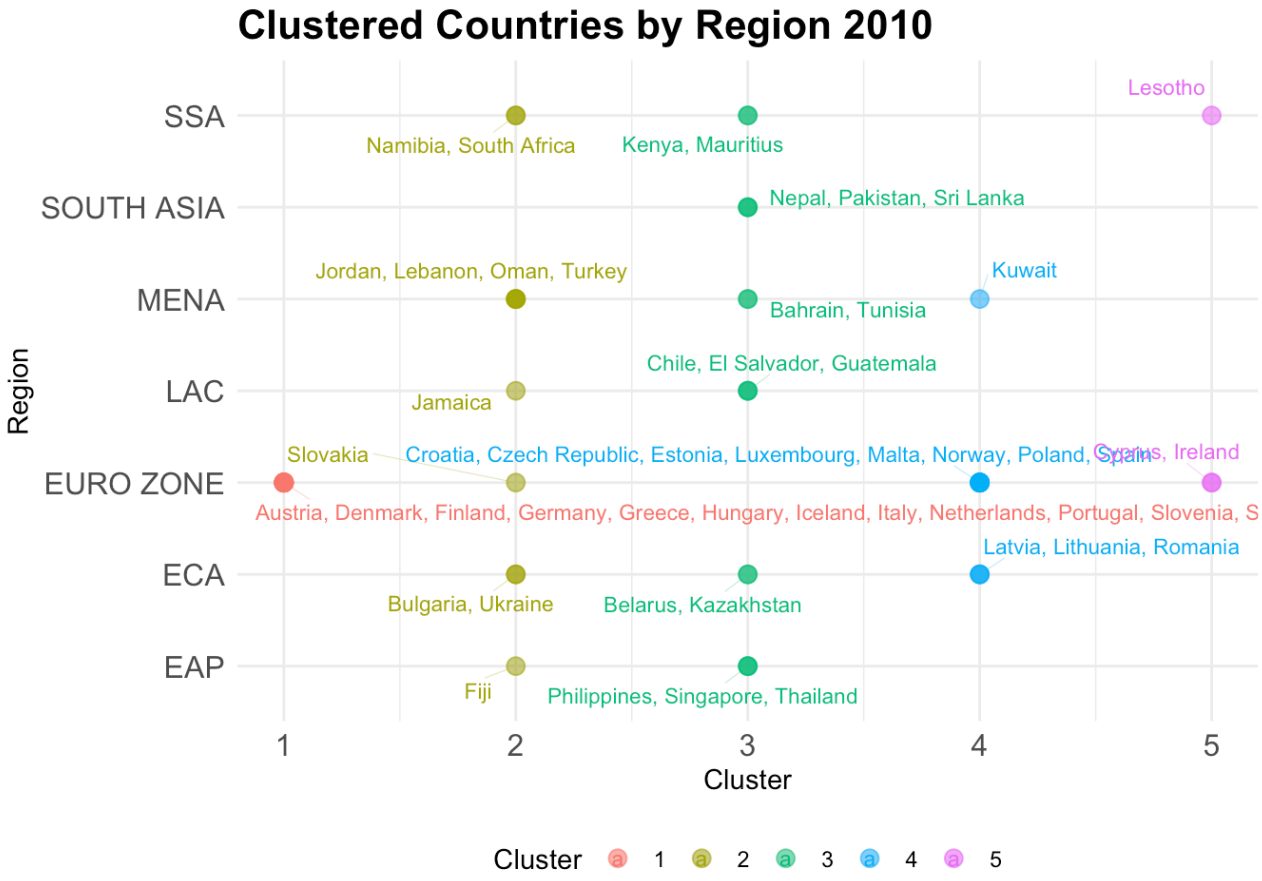
```
km_df2010num <- kmeans(df2010num, 5, nstart = 20)

# Create a data frame with country, region, and cluster columns
country_region_cluster <- data.frame(country = df2010$country, region = df2010$region, clu
ster = km_df2010num$cluster)

# Create a scatter plot
ggplot(country_region_cluster, aes(x = factor(cluster), y = region, color = factor(cluste
r))) +
  geom_point(size = 3) +
  labs(x = "Cluster", y = "Region", title = "Clustered Countries by Region") +
  scale_color_discrete(name = "Cluster")
```




```
country_region_cluster2010 <- data.frame(
  country = df2010$country,
  region = df2010$region,
  cluster = km_df2010num$cluster
)

# Group the countries by cluster and region
grouped_countries2010 <- country_region_cluster2010 %>%
  group_by(cluster, region) %>%
  summarise(countries = paste(country, collapse = ", "))
```

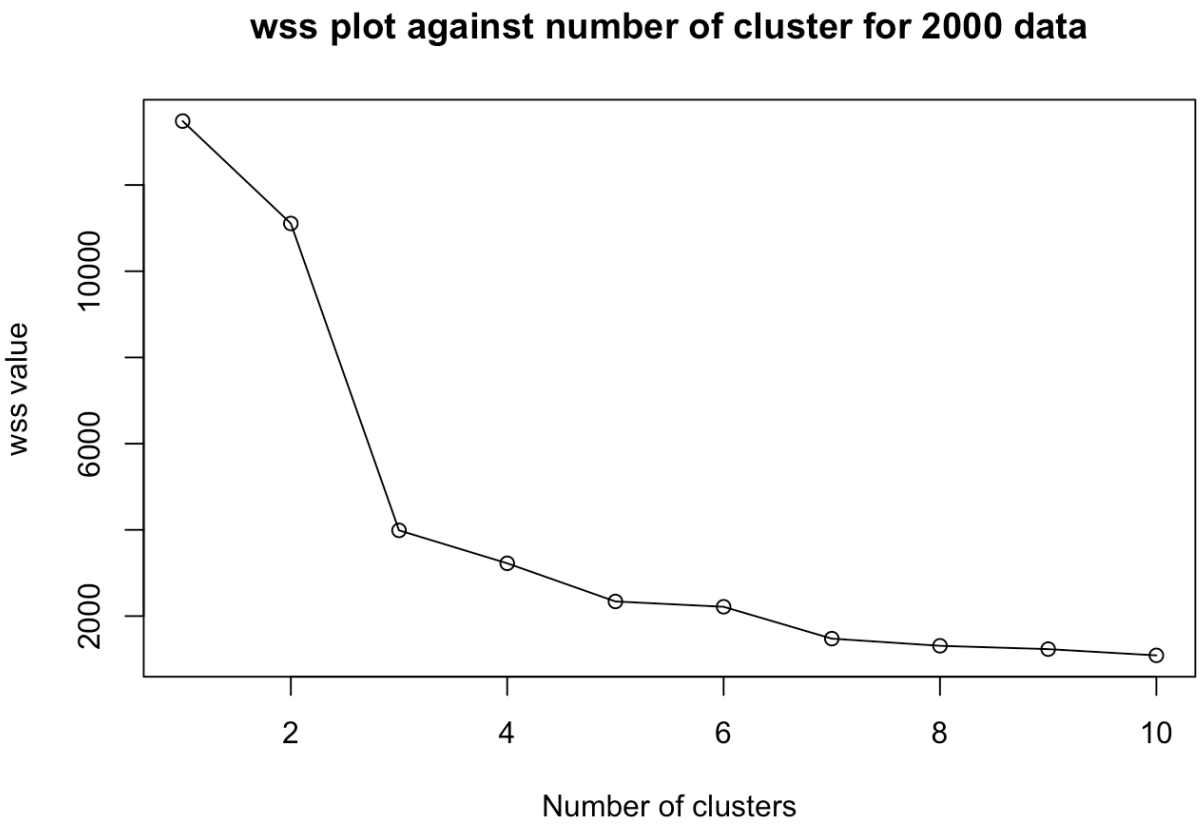
```
## `summarise()` has grouped output by 'cluster'. You can override using the
## `.groups` argument.
```

```
ggplot(country_region_cluster2010, aes(x = cluster, y = region, color = factor(cluster)))
+
  geom_point(size = 3, alpha = 0.6) +
  geom_text_repel(data = grouped_countries2010, aes(label = countries), size = 3, box.padd
ing = 0.5,
                  point.padding = 0.3, force = 10, segment.alpha = 0.3, segment.size = 0.
2) +
  labs(x = "Cluster", y = "Region", title = "Clustered Countries by Region 2010") +
  scale_color_discrete(name = "Cluster") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.text = element_text(size = 12),
        legend.position = "bottom")
```



K-means Clustering- 2010

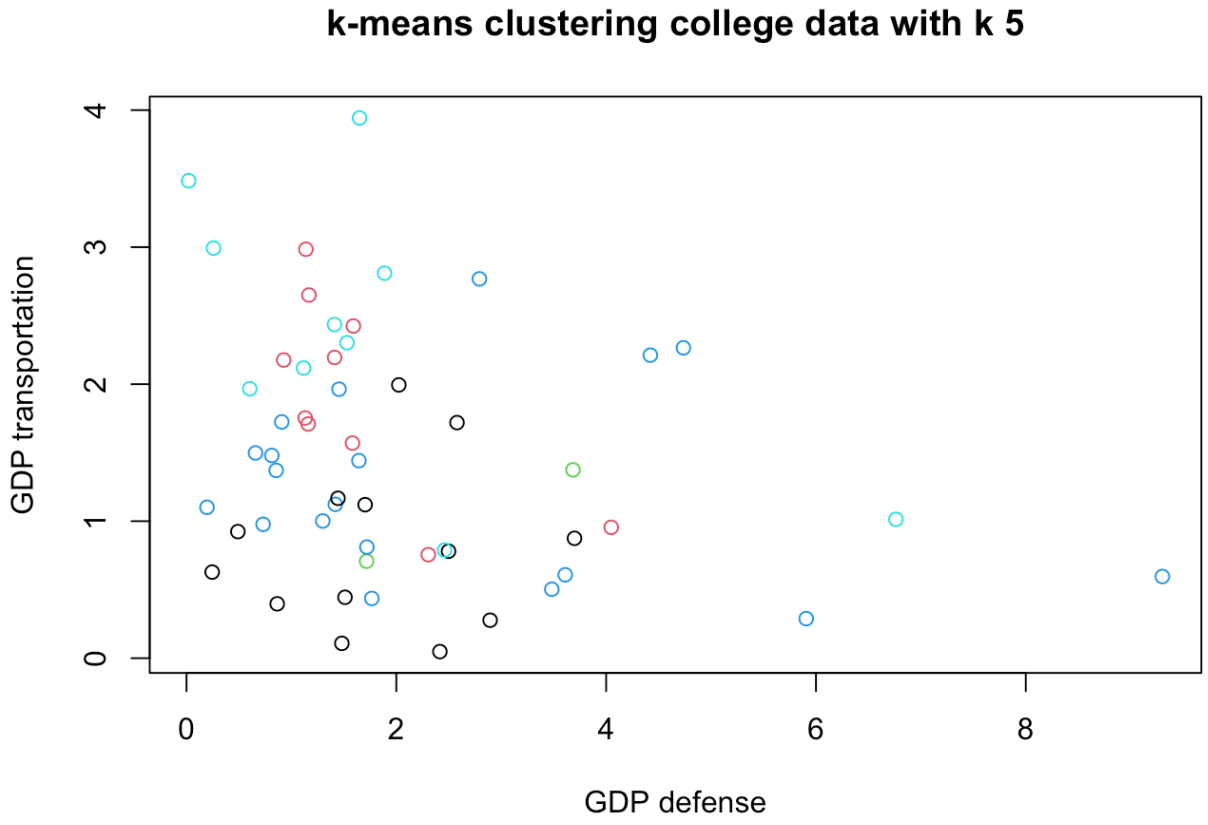
```
set.seed(1)
wss<- NULL
for (i in 1:10){
  fit = kmeans(df2000num,centers = i)
  wss = c(wss, fit$tot.withinss)
}
plot(1:10, wss, type = "o", main = 'wss plot against number of cluster for 2000 data', xla
b = "Number of clusters", ylab = "wss value")
```



The wss seems fairly stabilized after cluster number 5, so choosing to make 5 clusters moving forward:

k cluster with 5 clusters and plotting to observe the relationship with other variables.

```
km_df2000num <- kmeans(df2000num, 5, nstart = 20) #iteration for initializing is 20

plot(df2000num[, c("gdpdefense_ppp", "gdptransp_ppp")],
     col = km_df2000num$cluster,
     main = paste("k-means clustering college data with k 5"),
     xlab = "GDP defense", ylab = "GDP transportation")
```



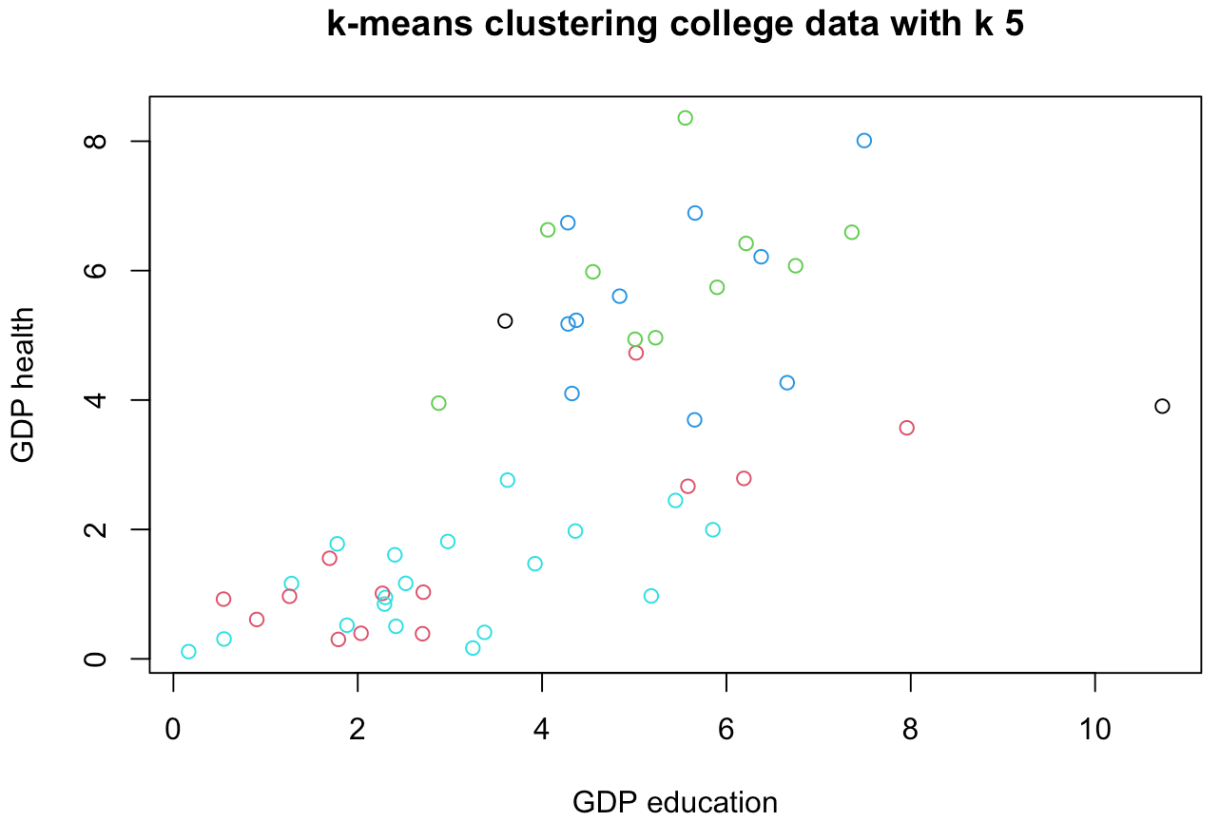

```
    #xlim = c(min(df2010$country), max(df2010$country)))
```

```
km_df2000num$tot.withinss
```

```
## [1] 2281.719
```

The within the cluster distance was 2281.72

```
km_df2000num <- kmeans(df2000num, 5, nstart = 20) #iteration for initializing is 20

plot(df2000num[, c("gdpeducation_ppp", "gdphealth_ppp")],
     col = km_df2000num$cluster,
     main = paste("k-means clustering college data with k 5"),
     xlab = "GDP education", ylab = "GDP health")
```




```
country_cluster2000 <- data.frame(country = df2000$country, cluster = km_df2000num$cluste
r)

# Print the data frame
print(country_cluster2000)
```

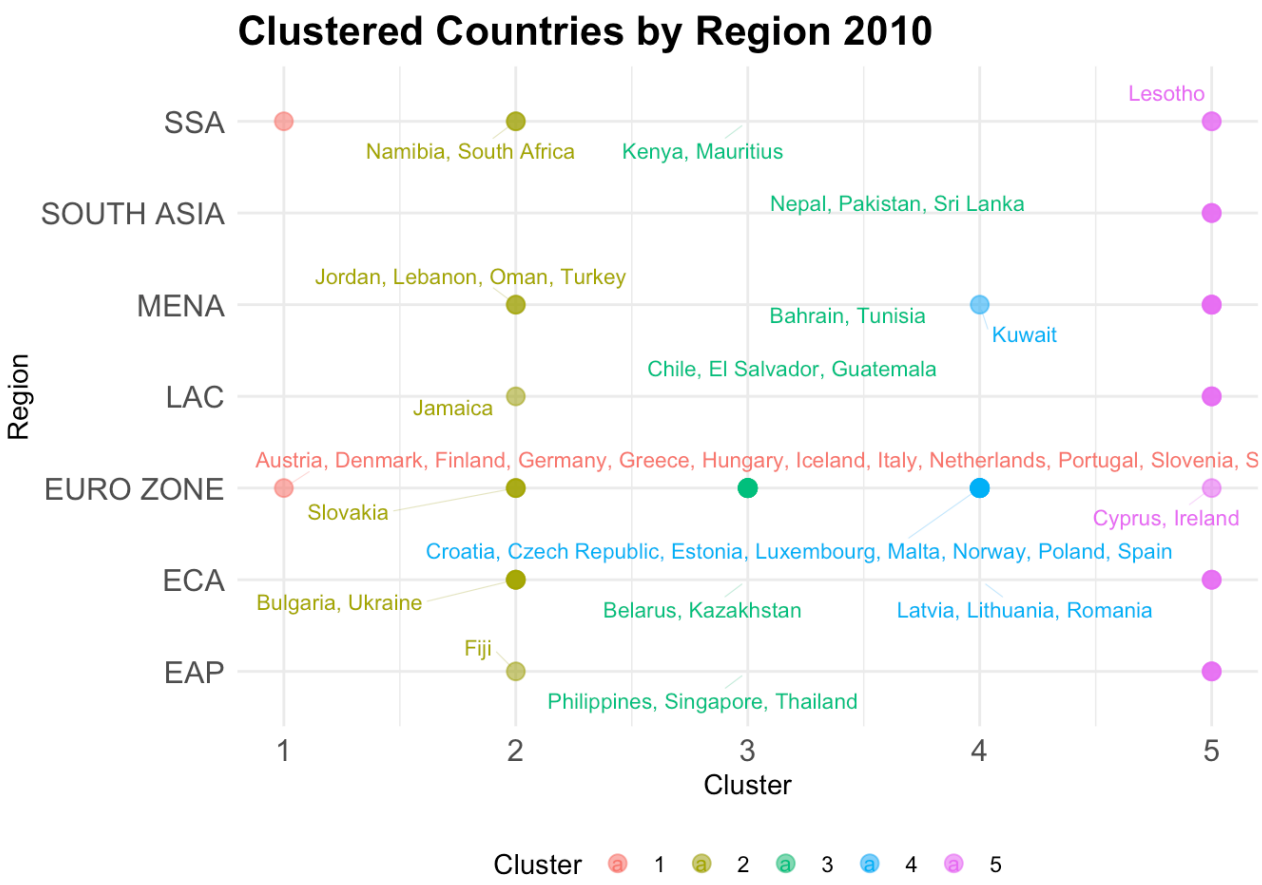
```
##            country cluster
## 2             Fiji       2
## 8      Philippines       5
## 9        Singapore       5
## 10        Thailand       5
## 16         Belarus       5
## 17        Bulgaria       2
## 19      Kazakhstan       5
## 21          Latvia       2
## 22       Lithuania       2
## 24         Romania       2
## 27         Ukraine       5
## 28         Austria       3
## 30         Croatia       5
## 31          Cyprus       2
## 32  Czech Republic       4
## 33         Denmark       3
## 34         Estonia       4
## 35         Finland       3
## 37         Germany       3
## 38          Greece       3
## 39         Hungary       3
## 40         Iceland       4
## 41         Ireland       4
## 42           Italy       3
## 43      Luxembourg       4
## 44           Malta       2
## 45     Netherlands       3
## 46          Norway       4
## 47          Poland       2
## 48        Portugal       4
## 49        Slovakia       1
## 50        Slovenia       3
## 51           Spain       4
## 52          Sweden       3
## 54  United Kingdom       4
## 68           Chile       5
## 73     El Salvador       5
## 75       Guatemala       5
## 76         Jamaica       2
## 86         Bahrain       5
## 89          Jordan       4
## 90          Kuwait       5
## 91         Lebanon       2
## 94            Oman       5
## 97         Tunisia       5
## 98          Turkey       2
## 106          Nepal       5
## 107       Pakistan       5
## 108      Sri Lanka       5
## 125          Kenya       5
## 126        Lesotho       1
## 131      Mauritius       5
```

```
## 133        Namibia       2
## 140   South Africa       2
```

```
country_region_cluster2000 <- data.frame(
  country = df2000$country,
  region = df2000$region,
  cluster = km_df2000num$cluster
)

# Group the countries by cluster and region
grouped_countries2000 <- country_region_cluster2000 %>%
  group_by(cluster, region) %>%
  summarise(countries = paste(country, collapse = ", "))
```

```
## `summarise()` has grouped output by 'cluster'. You can override using the
## `.groups` argument.
```

```
library(ggrepel)
ggplot(country_region_cluster2000, aes(x = cluster, y = region, color = factor(cluster)))
+
  geom_point(size = 3, alpha = 0.6) +
  geom_text_repel(data = grouped_countries2010, aes(label = countries), size = 3, box.padd
ing = 0.5,
                  point.padding = 0.3, force = 10, segment.alpha = 0.3, segment.size = 0.
2) +
  labs(x = "Cluster", y = "Region", title = "Clustered Countries by Region 2010") +
  scale_color_discrete(name = "Cluster") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.text = element_text(size = 12),
        legend.position = "bottom")
```

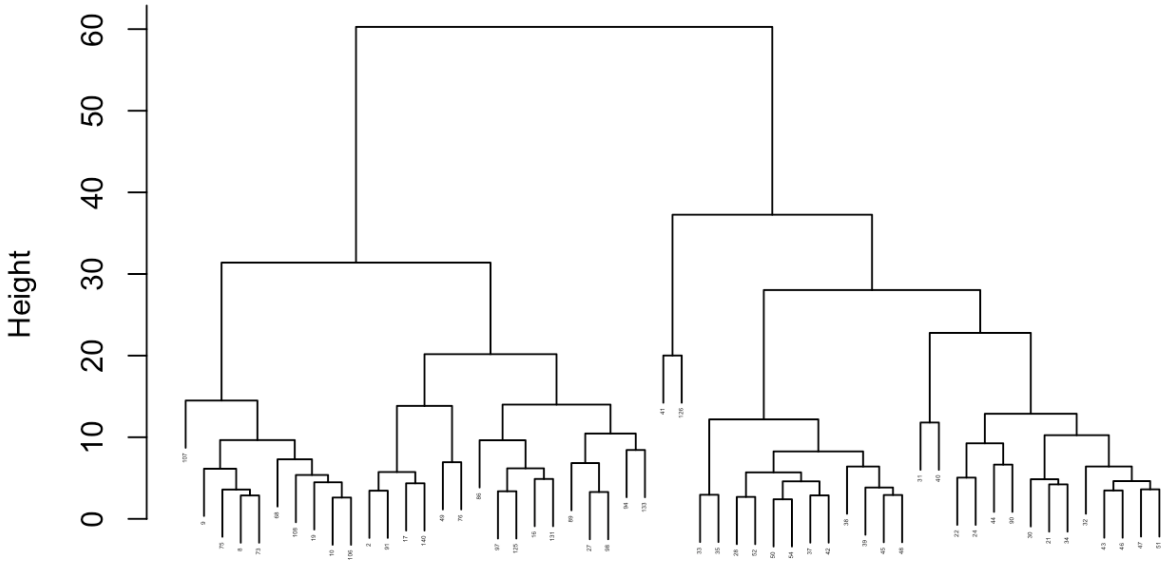


Hierarchical Clustering- 2010

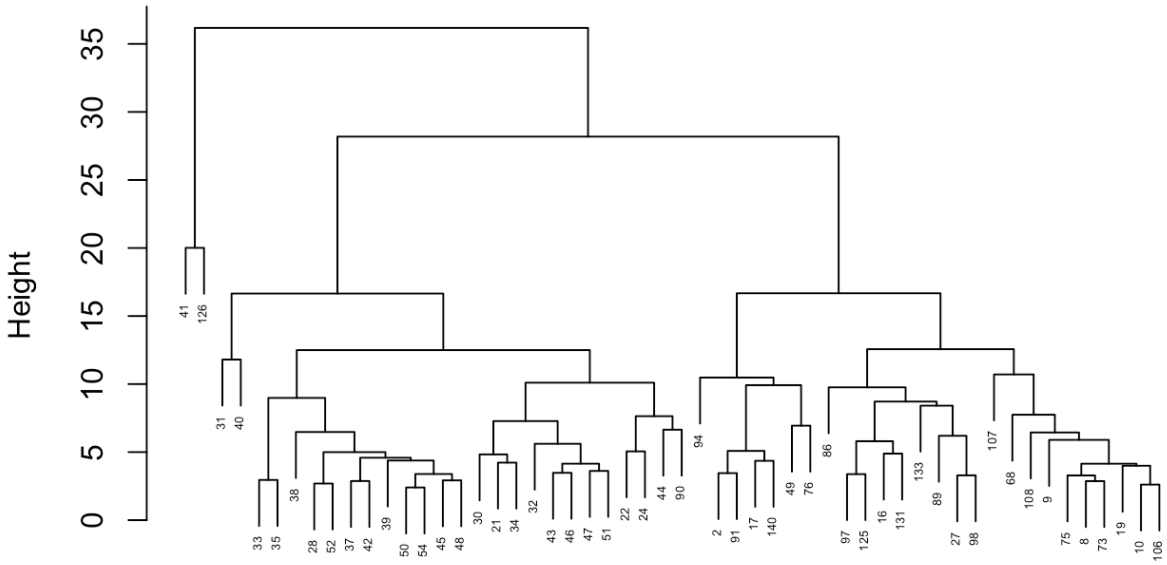
```
hcluster <- function(df){

    df_complete <- hclust(dist(df), method = "complete")
    df_average <- hclust(dist(df), method = "average")
    df_single <- hclust(dist(df), method = "single")
    df_centroid <- hclust(dist(df), method = "centroid")

    plot(df_complete, main = "Complete Linkage",
    xlab = "", sub = "", cex = .2)
    plot(df_average, main = "Average Linkage",
    xlab = "", sub = "", cex = .4)
  plot(df_single, main = "Single Linkage",
    xlab = "", sub = "", cex = .4)
  plot(df_centroid, main = "Centroid Linkage",
    xlab = "", sub = "", cex = .4)
}
```

```
hcluster(df2010num)   #call the function
```

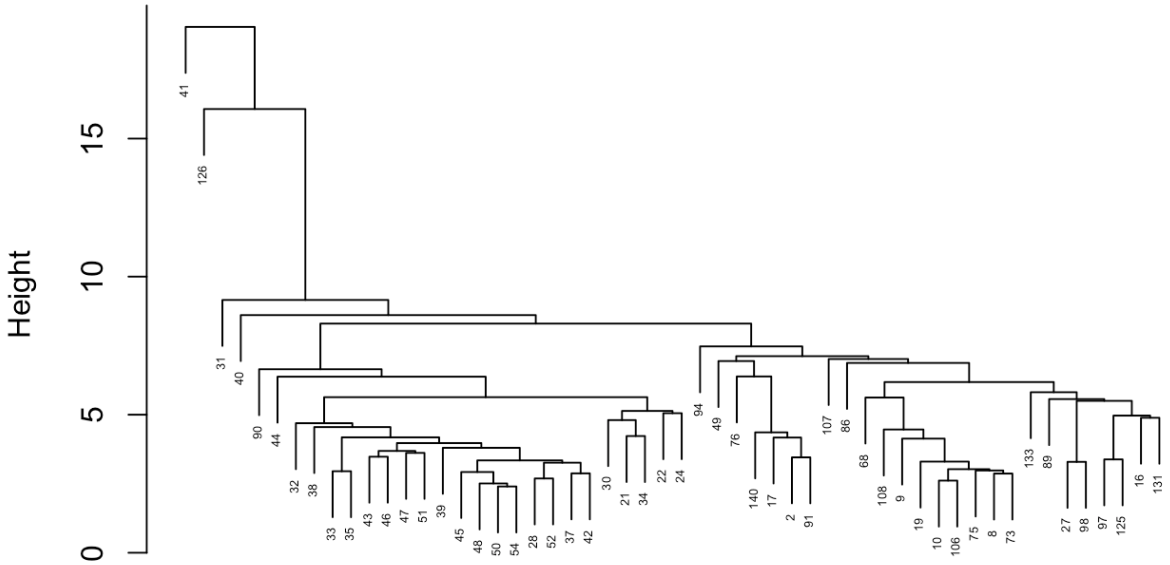
Complete Linkage



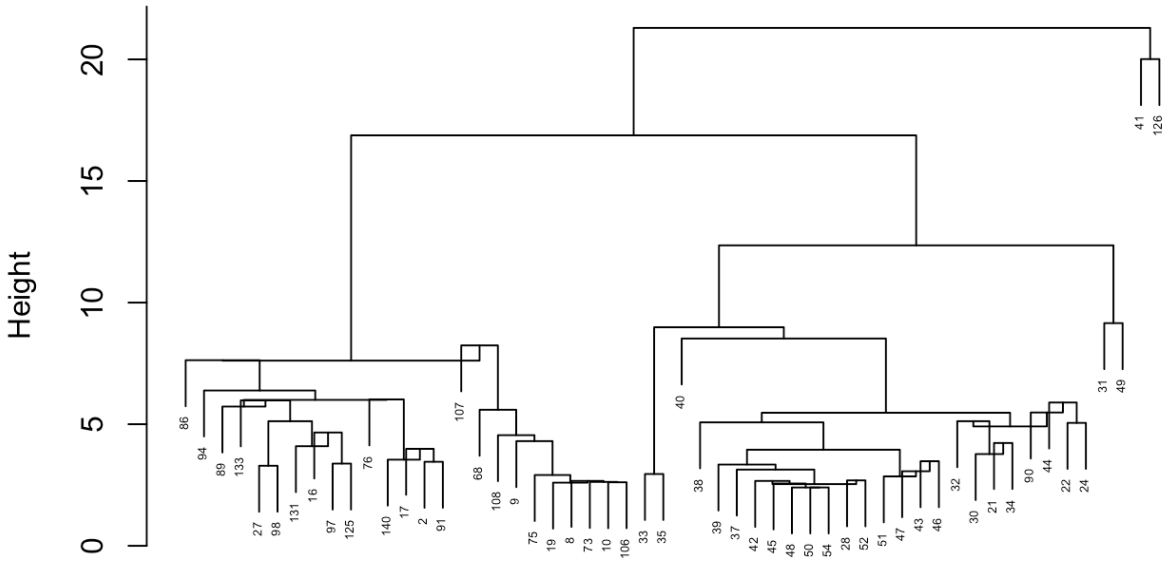
Average Linkage



Single Linkage



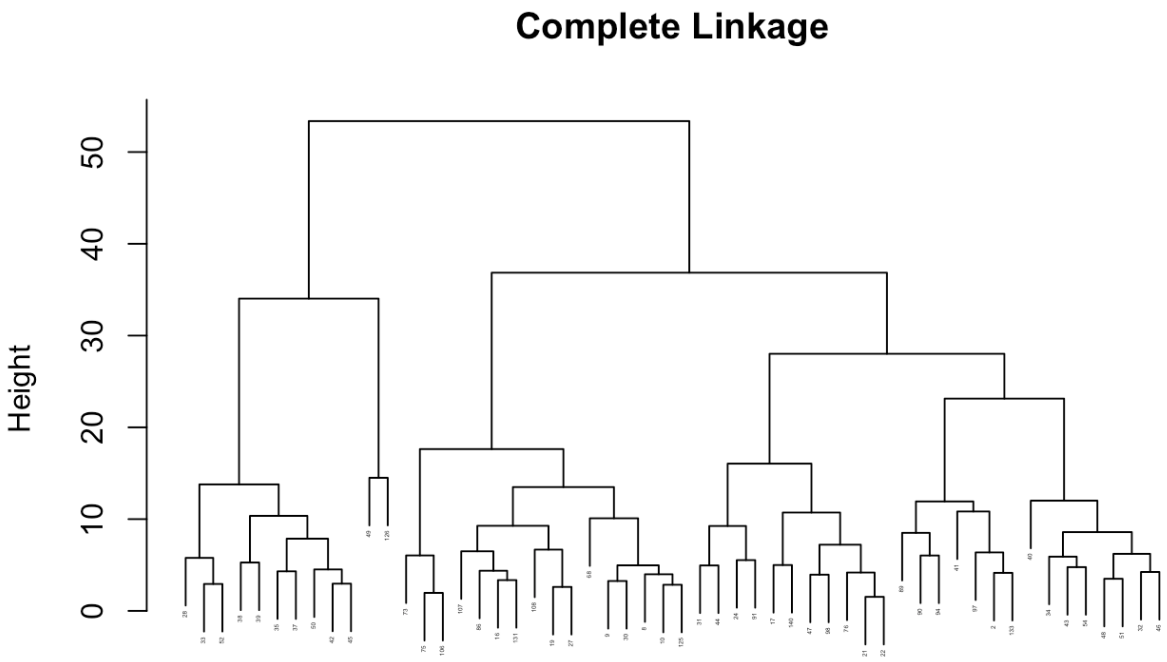
Centroid Linkage



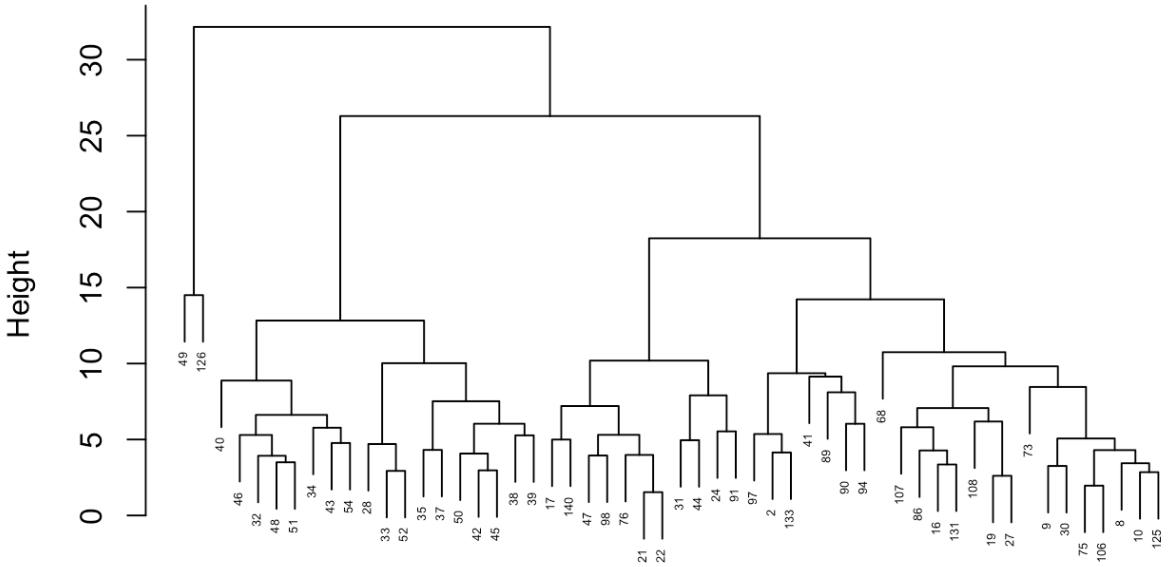
Among all options for the htree, complete linkage had the most even split so using complete linkage with 5 as the cut, we have 4 clusters for 2010 data.

Hierarchical Clustering - 200

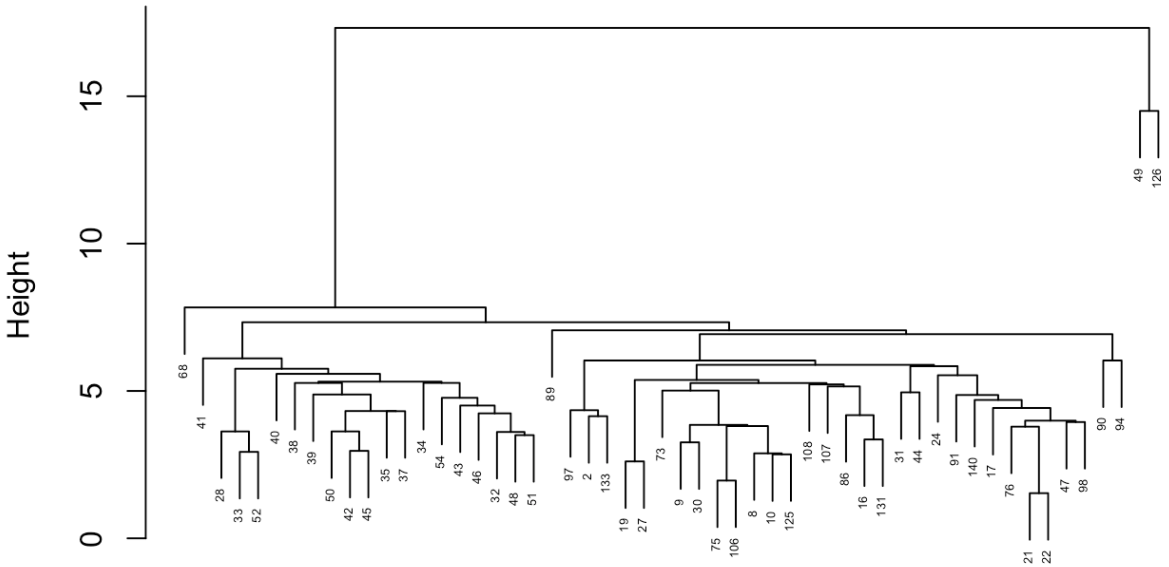
```
hcluster(df2000num)  #call the function
```

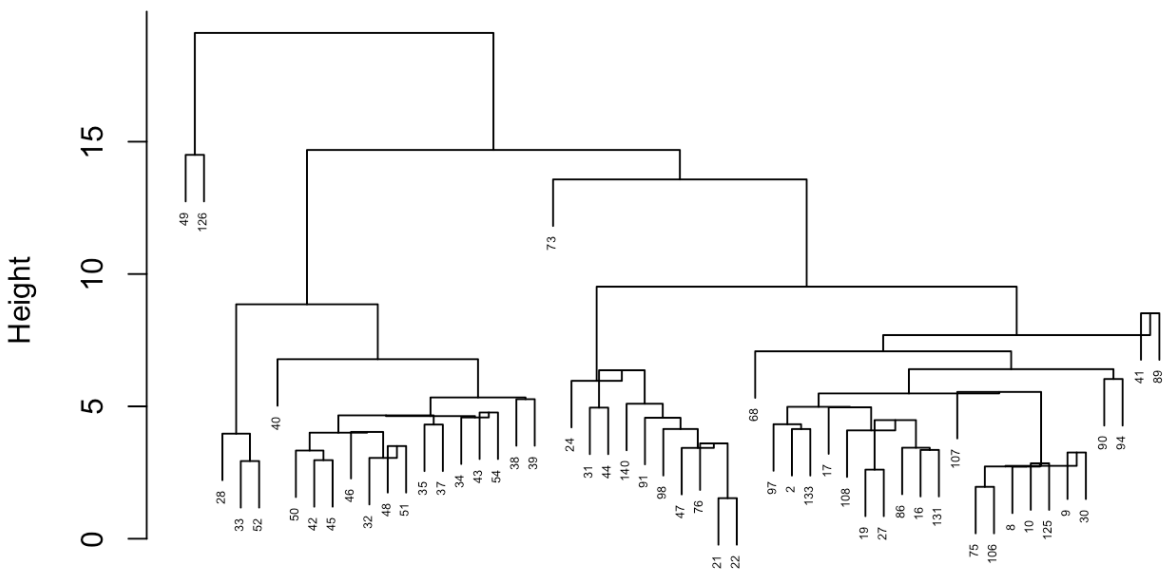


Average Linkage



Single Linkage





Compared to all graphs, complete works for both 2000 and 2010 datasets.

FOr both the average linkage with 4 clusters had the best and balanced cluster division.

Print how many clusters there are and what are included in the cluster

2010 dataset

```
hcut_df2000 <- hclust(dist(df2000num), method = "complete")
cluster_2000 <- cutree(hcut_df2000, 4)
cluster_df <- data.frame(country = df2000$country, cluster = cluster_2000)

for (i in unique(cluster_df$cluster)) {
  cat("Cluster", i, ":\n")
  countries_in_cluster <- cluster_df$country[cluster_df$cluster == i]
  cat(paste(countries_in_cluster, collapse = ", "), "\n\n")

num_countries_in_cluster <- length(countries_in_cluster)
  cat("Number of countries in Cluster", i, ":", num_countries_in_cluster, "\n\n")

}
```

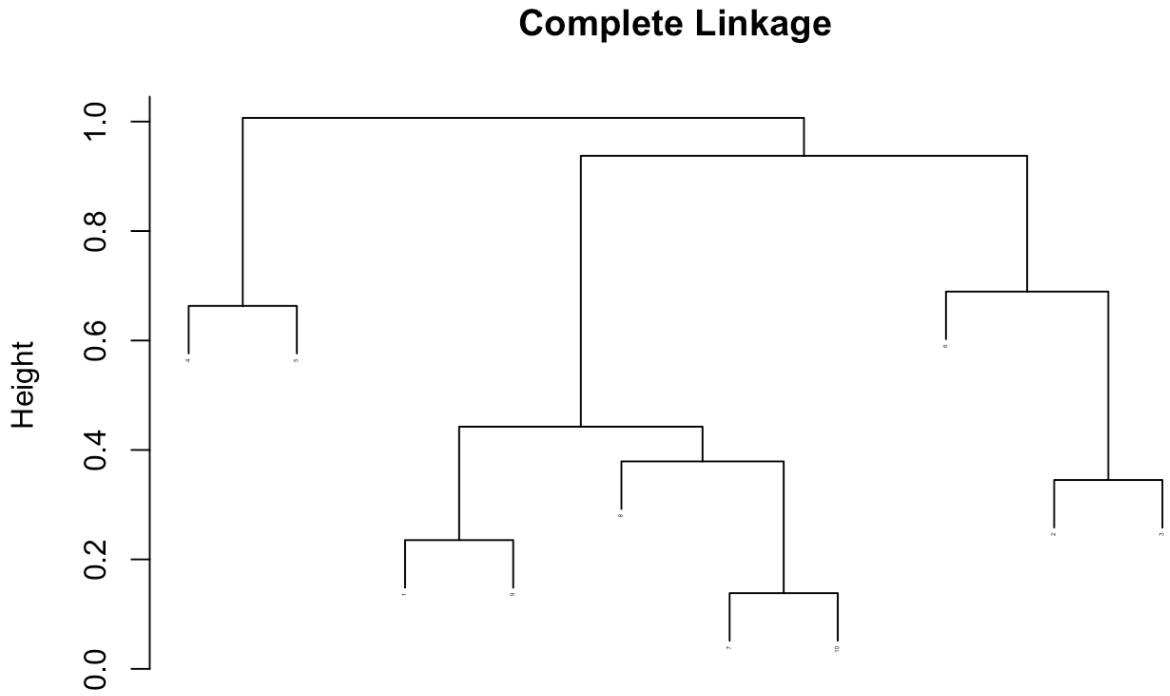
```
## Cluster 1 :
## Fiji, Bulgaria, Latvia, Lithuania, Romania, Cyprus, Czech Republic, Estonia, Iceland, I
reland, Luxembourg, Malta, Norway, Poland, Portugal, Spain, United Kingdom, Jamaica, Jorda
n, Kuwait, Lebanon, Oman, Tunisia, Turkey, Namibia, South Africa
##
## Number of countries in Cluster 1 : 26
##
## Cluster 2 :
## Philippines, Singapore, Thailand, Belarus, Kazakhstan, Ukraine, Croatia, Chile, El Salv
ador, Guatemala, Bahrain, Nepal, Pakistan, Sri Lanka, Kenya, Mauritius
##
## Number of countries in Cluster 2 : 16
##
## Cluster 3 :
## Austria, Denmark, Finland, Germany, Greece, Hungary, Italy, Netherlands, Slovenia, Swed
en
##
## Number of countries in Cluster 3 : 10
##
## Cluster 4 :
## Slovakia, Lesotho
##
## Number of countries in Cluster 4 : 2
```

```
hcut_df2010 <- hclust(dist(df2010num), method = "complete")
cluster_2010 <- cutree(hcut_df2010, 4)
cluster_df <- data.frame(country = df2010$country, cluster = cluster_2010)

for (i in unique(cluster_df$cluster)) {
  cat("Cluster", i, ":\n")
  countries_in_cluster <- cluster_df$country[cluster_df$cluster == i]
  cat(paste(countries_in_cluster, collapse = ", "), "\n\n")
  num_countries_in_cluster <- length(countries_in_cluster)
  cat("Number of countries in Cluster", i, ":", num_countries_in_cluster, "\n\n")
}
```

```
## Cluster 1 :
## Fiji, Belarus, Bulgaria, Ukraine, Slovakia, Jamaica, Bahrain, Jordan, Lebanon, Oman, Tu
nisia, Turkey, Kenya, Mauritius, Namibia, South Africa
##
## Number of countries in Cluster 1 : 16
##
## Cluster 2 :
## Philippines, Singapore, Thailand, Kazakhstan, Chile, El Salvador, Guatemala, Nepal, Pak
istan, Sri Lanka
##
## Number of countries in Cluster 2 : 10
##
## Cluster 3 :
## Latvia, Lithuania, Romania, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia,
Finland, Germany, Greece, Hungary, Iceland, Italy, Luxembourg, Malta, Netherlands, Norway,
Poland, Portugal, Slovenia, Spain, Sweden, United Kingdom, Kuwait
##
## Number of countries in Cluster 3 : 26
##
## Cluster 4 :
## Ireland, Lesotho
##
## Number of countries in Cluster 4 : 2
```

code for the theorotical section of clustering:

```
library(random)
X <- matrix(runif(30), nrow = 10, ncol = 3)

df_tree <- hclust(dist(X), method = "complete")
plot(df_tree, main = "Complete Linkage",
    xlab = "", sub = "", cex = .2)
```



Supervised Learning: Decision Trees

Predictor is region so remove country

```
df2000 <- subset(data2000, select = -country)
df2010 <- subset(data2010, select = -country)
```

Make region a factor

```
df2000$region <- factor(data2000$region)
df2010$region <- factor(data2010$region)
```

2000

Basic tree

```
set.seed(1)
train <- sample(1:nrow(df2000), nrow(df2000)*0.6)
df2000.test <- df2000[-train, ]
region2000.test <- df2000$region[-train]
```

```
tree2000 <- tree(region ~ . - region , df2000, subset = train)
summary(tree2000)
```

```
##
## Classification tree:
## tree(formula = region ~ . - region, data = df2000, subset = train)
## Variables actually used in tree construction:
## [1] "gdphealth_ppp"  "gdptransp_ppp"  "gdpdefense_ppp"
## Number of terminal nodes:  4
## Residual mean deviance:  1.507 = 42.18 / 28
## Misclassification error rate: 0.3125 = 10 / 32
```

```
tree.pred <- predict(tree2000, df2000.test,
    type = "class")
mean(tree.pred == region2000.test)
```

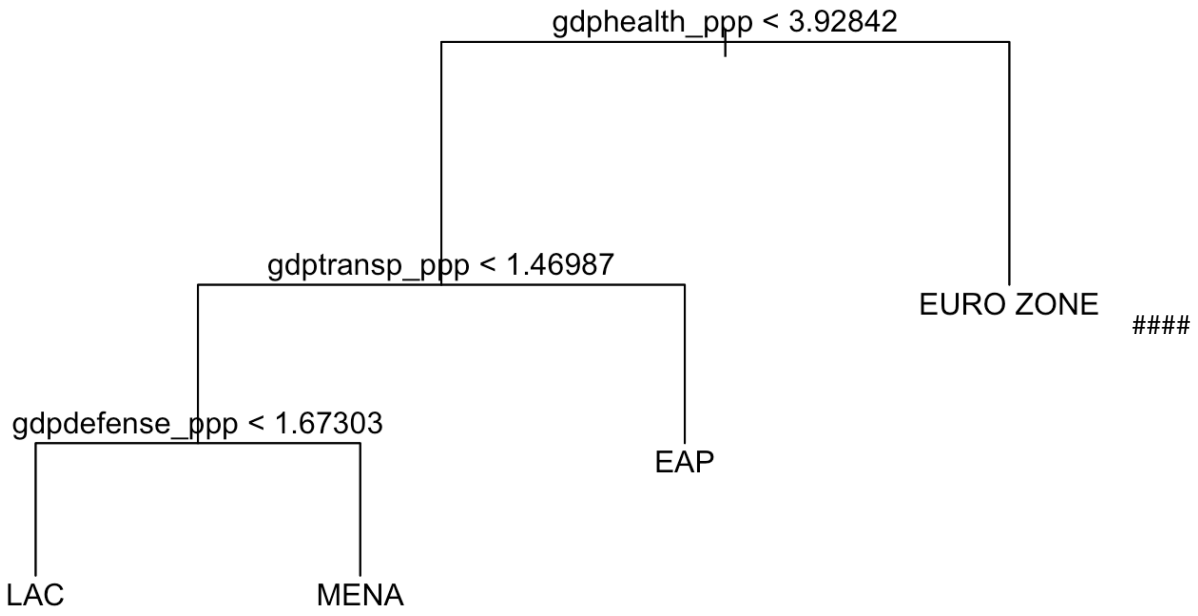
```
## [1] 0.5909091
```

```
table(tree.pred, region2000.test)
```

```
##             region2000.test
## tree.pred    EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          1   0         1   1    0          0   0
##   ECA          0   0         0   0    0          0   0
##   EURO ZONE    0   0        10   0    0          0   0
##   LAC          0   2         0   0    0          1   2
##   MENA         0   1         0   0    2          1   0
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

Prediction accuracy of 59%, seems to be best at predicting Eurozone (or there are just more obs for eurozone).

```
plot(tree2000)
text(tree2000, pretty = 0)
```

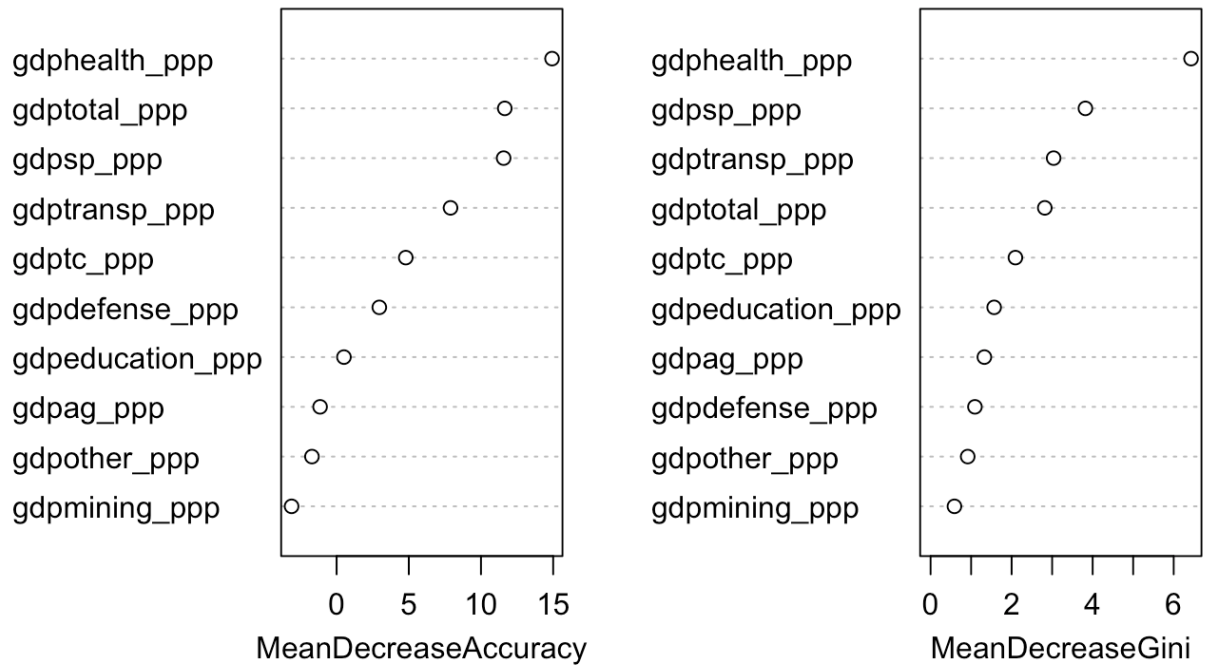


Bagging

```
bag2000 <- randomForest(region ~ . , data = df2000,
    subset = train, mtry = 10, importance = TRUE)
bag2000
```

```
##
## Call:
##  randomForest(formula = region ~ ., data = df2000, mtry = 10,      importance = TRUE, s
ubset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 53.12%
## Confusion matrix:
##             EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA class.error
## EAP           1   0         0   1    0          0   1   0.6666667
## ECA           1   0         1   1    1          0   0   1.0000000
## EURO ZONE     0   0        11   0    1          0   1   0.1538462
## LAC           0   2         1   0    0          0   0   1.0000000
## MENA          0   0         0   1    3          0   1   0.4000000
## SOUTH ASIA    1   0         0   0    0          0   0   1.0000000
## SSA           0   0         2   1    0          0   0   1.0000000
```

```
varImpPlot(bag2000)
```




```
bag.pred <- predict(bag2000, df2000.test, type = 'class')
mean(bag.pred == region2000.test)
```

```
## [1] 0.6363636
```

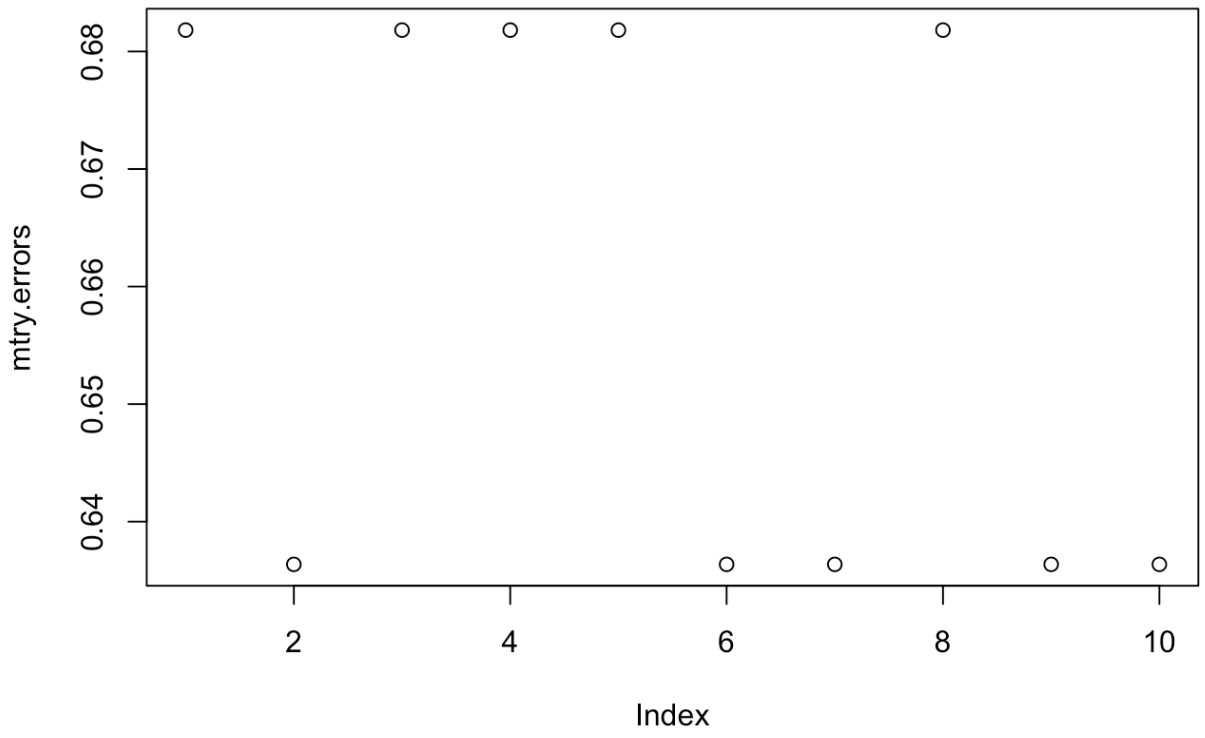
```
table(bag.pred, region2000.test)
```

```
##             region2000.test
## bag.pred     EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          0   0         1   0    0          0   0
##   ECA          1   2         0   0    0          0   0
##   EURO ZONE    0   0        10   0    1          0   0
##   LAC          0   0         0   1    0          1   1
##   MENA         0   1         0   0    1          1   1
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

Random Forest

```
set.seed(1)
mtry.errors<- c()
for (i in 1:10){
  mod <- randomForest(region ~ ., data = df2000,
                      subset = train, mtry = i)
  pred <- predict(mod, df2000.test, type = 'class')
  mtry.errors[i] <- mean(pred == region2000.test)
}
```

```
plot(mtry.errors)
```



2, 6, 7, 9 or 10 are the best number of trees. 10 is boosting so try 2, 3, 4 trees with different ntree?

```
set.seed(1)
tune <- function (m){
  errors <- c()
  ntree <- c(100, 250, 500)
  n <- length(ntree)
  for (i in 1:n){
    mod <- randomForest(region ~ ., data = df2000,
                        subset = train, mtry = m, ntrees = ntree[i])
    pred <- predict(mod, df2000.test, type = 'class')
    errors[i] <- mean(pred == region2000.test)
    }
  return(errors)
}
```

```
errors2 <- tune(2)
errors2
```

```
## [1] 0.6818182 0.6818182 0.6818182
```

```
errors3 <- tune(3)
errors3
```

```
## [1] 0.6818182 0.6818182 0.6818182
```

```
errors4 <- tune(4)
errors4
```

```
## [1] 0.6818182 0.6818182 0.6818182
```

Not changing much but even if you made ntree something really big or something really large it's not changing much.

```
set.seed(1)
bestmod <- randomForest(region ~., data = df2000,
                        subset= train, mtry = 3, ntree= 250,
                        importance = TRUE)
bestmod
```

```
##
## Call:
##  randomForest(formula = region ~ ., data = df2000, mtry = 3, ntree = 250,      importan
ce = TRUE, subset = train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 53.12%
## Confusion matrix:
##             EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA class.error
## EAP           1   0         1   0    0          1   0   0.6666667
## ECA           1   0         1   1    1          0   0   1.0000000
## EURO ZONE     0   0        11   0    1          0   1   0.1538462
## LAC           0   1         1   1    0          0   0   0.6666667
## MENA          0   2         0   0    2          0   1   0.6000000
## SOUTH ASIA    1   0         0   0    0          0   0   1.0000000
## SSA           0   1         2   0    0          0   0   1.0000000
```

```
bestpred <- predict(bestmod, df2000.test, type = 'class')
mean(bestpred == region2000.test)
```

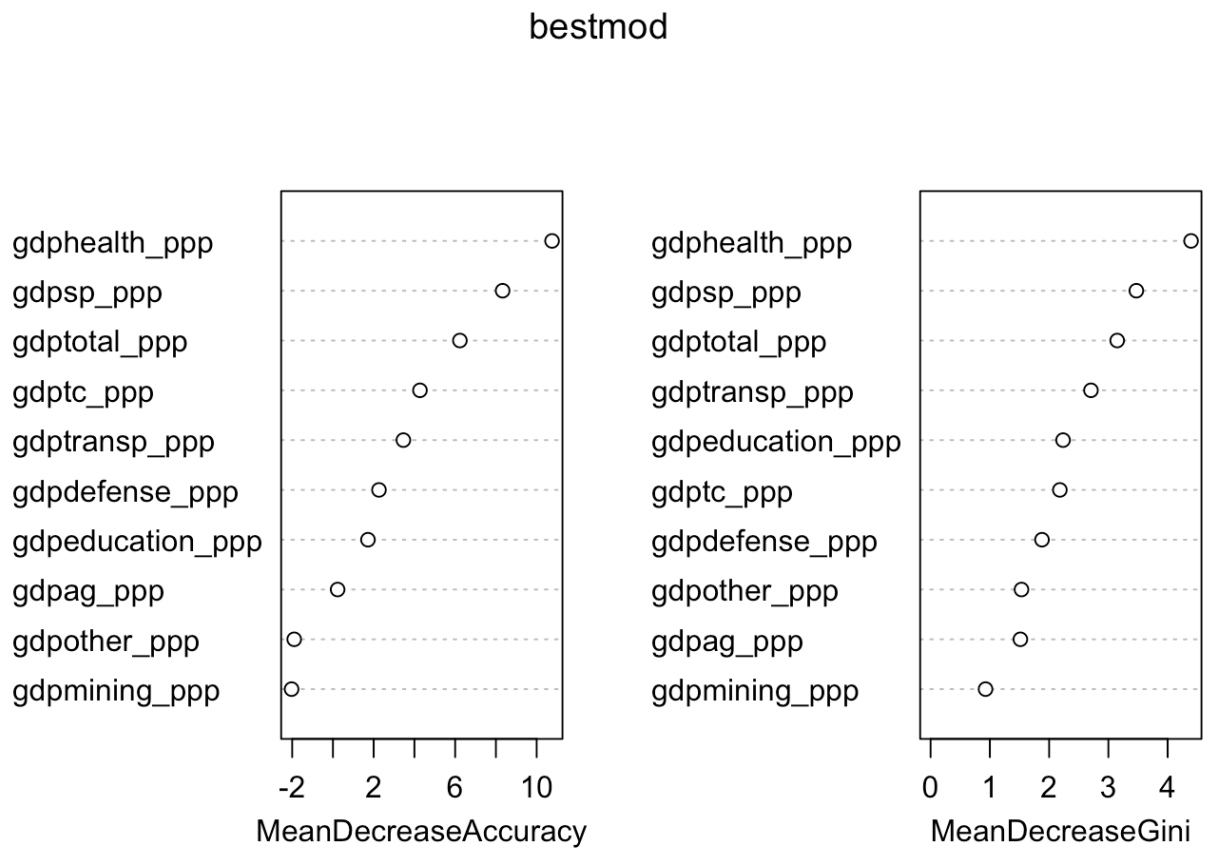
```
## [1] 0.6818182
```

```
table(bestpred, region2000.test)
```

```
##             region2000.test
## bestpred     EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          1   0         1   0    0          0   0
##   ECA          0   2         0   0    0          0   0
##   EURO ZONE    0   0        10   0    1          0   0
##   LAC          0   0         0   1    0          1   1
##   MENA         0   1         0   0    1          1   1
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

```
countries <- data2000$country[-train]
treelabs2000 <- data.frame(countries, bestpred, region2000.test)
```

```
varImpPlot(bestmod)
```



2010

Basic tree

```
set.seed(1)
train <- sample(1:nrow(df2010), nrow(df2010)*0.6)
df2010.test <- df2010[-train, ]
region2010.test <- df2010$region[-train]
```

```
tree2010 <- tree(region ~ . - region , df2010, subset = train)
summary(tree2010)
```

```
##
## Classification tree:
## tree(formula = region ~ . - region, data = df2010, subset = train)
## Variables actually used in tree construction:
## [1] "gdptotal_ppp"     "gdpeducation_ppp" "gdpdefense_ppp"   "gdpag_ppp"
## Number of terminal nodes:  5
## Residual mean deviance:  1.5 = 40.51 / 27
## Misclassification error rate: 0.2812 = 9 / 32
```

```
tree.pred <- predict(tree2010, df2010.test,
    type = "class")
mean(tree.pred == region2010.test)
```

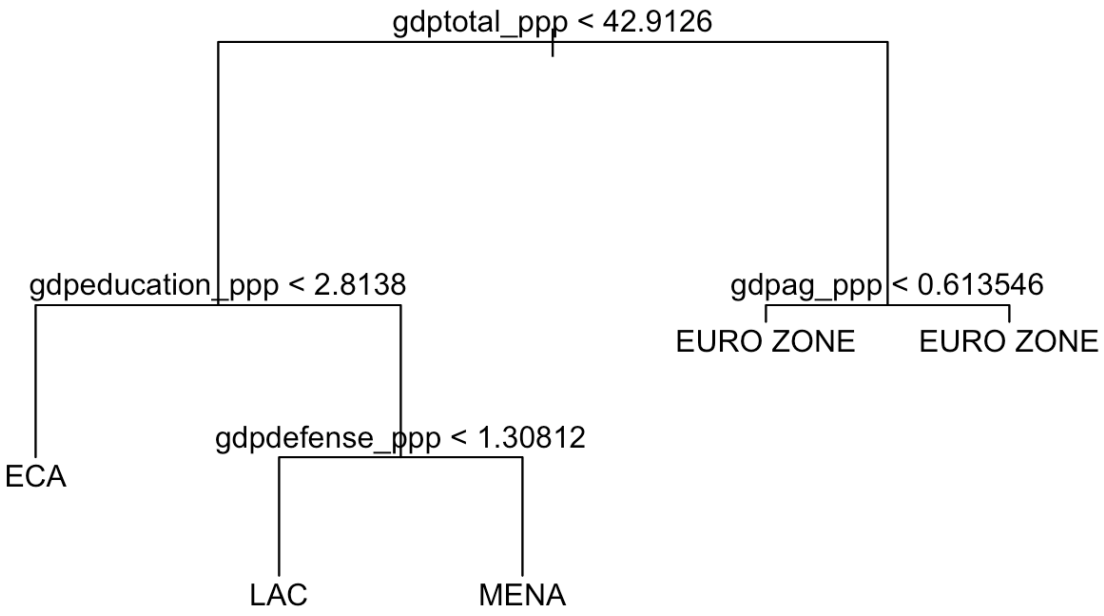
```
## [1] 0.5454545
```

```
table(tree.pred, region2010.test)
```

```
##             region2010.test
## tree.pred    EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          0   0         0   0    0          0   0
##   ECA          1   1         0   0    0          1   1
##   EURO ZONE    0   0         8   0    0          0   0
##   LAC          0   2         1   1    0          0   0
##   MENA         0   0         2   0    2          1   1
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

Similar to 2000, different predictor variables seem important

```
plot(tree2010)
text(tree2010, pretty = 0)
```

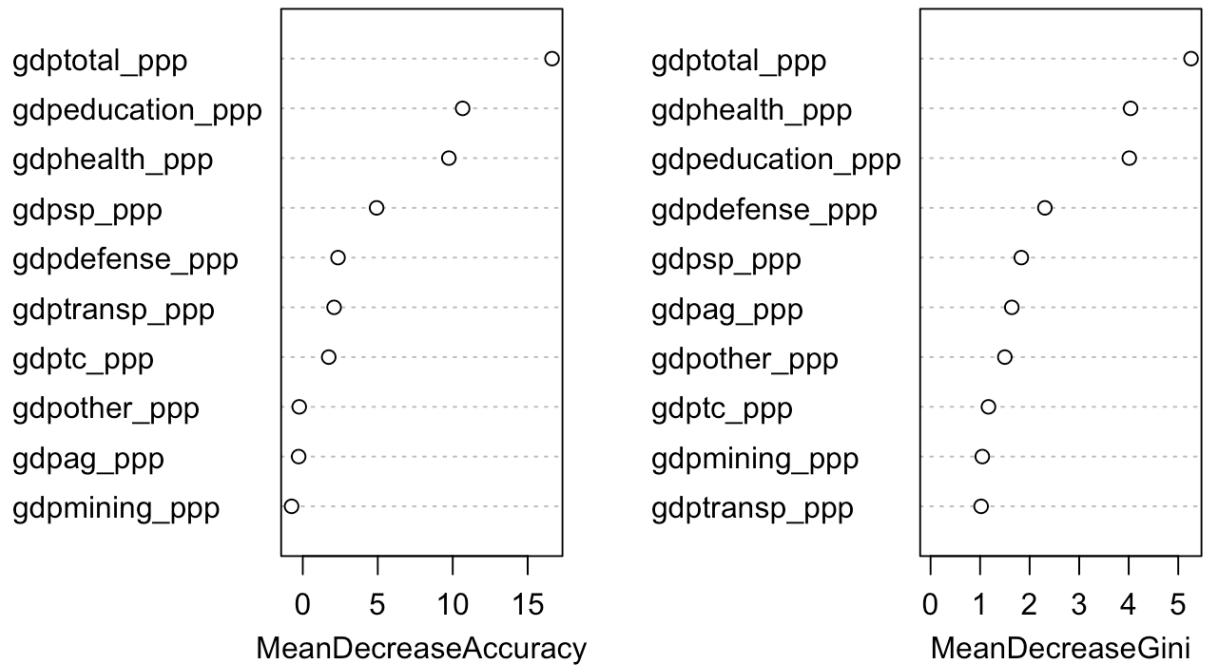


Bagging

```
bag2010 <- randomForest(region ~ . , data = df2010,
    subset = train, mtry = 10, importance = TRUE)
bag2010
```

```
##
## Call:
##  randomForest(formula = region ~ ., data = df2010, mtry = 10,      importance = TRUE, s
ubset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 50%
## Confusion matrix:
##            EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA class.error
## EAP          0   0         0   0    3          0   0  1.00000000
## ECA          0   2         1   0    1          0   0  0.50000000
## EURO ZONE    0   0        12   1    0          0   0  0.07692308
## LAC          2   0         0   0    1          0   0  1.00000000
## MENA         2   1         0   0    2          0   0  0.60000000
## SOUTH ASIA   0   1         0   0    0          0   0  1.00000000
## SSA          0   0         1   1    1          0   0  1.00000000
```

```
varImpPlot(bag2010)
```




```
bag.pred <- predict(bag2010, df2010.test, type = 'class')
mean(bag.pred == region2010.test)
```

```
## [1] 0.6363636
```

```
table(bag.pred, region2010.test)
```

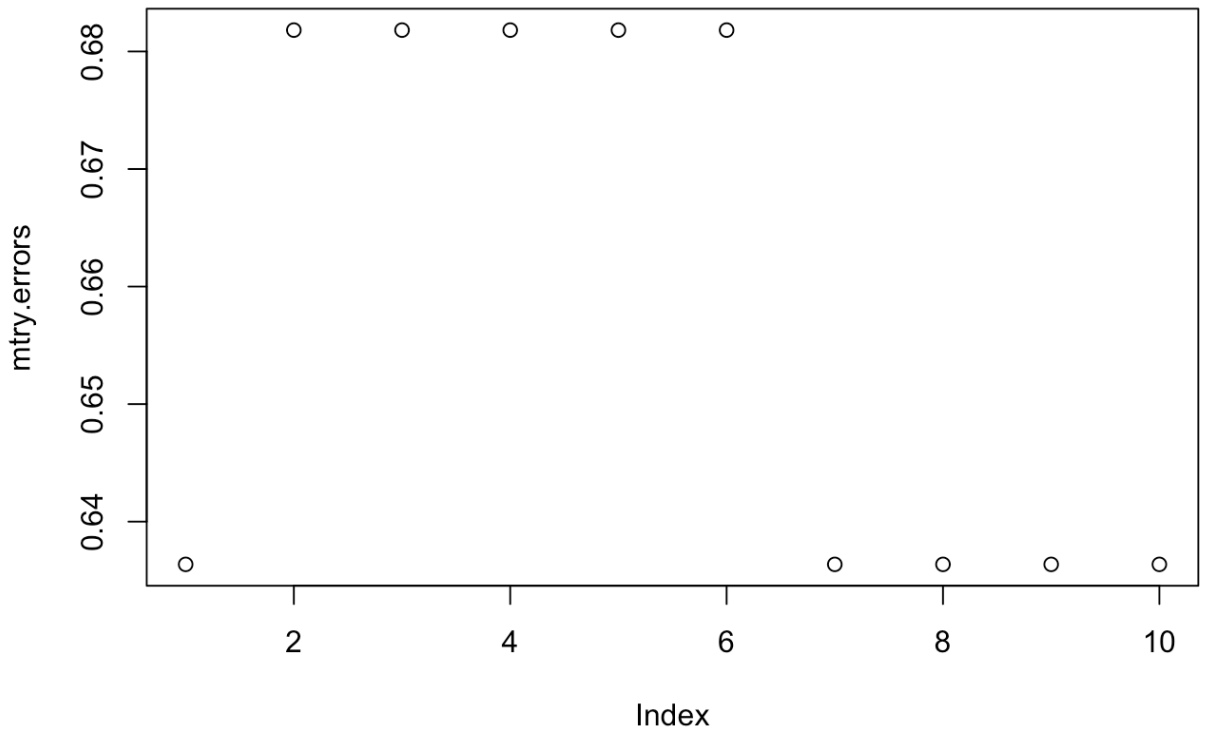
```
##             region2010.test
## bag.pred     EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          0   0         0   0    1          1   0
##   ECA          1   1         0   0    0          1   1
##   EURO ZONE    0   1        11   0    0          0   0
##   LAC          0   0         0   1    0          0   0
##   MENA         0   1         0   0    1          0   1
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

Slightly better than the basic tree

Random Forest

```
set.seed(1)
mtry.errors<- c()
for (i in 1:10){
  mod <- randomForest(region ~ ., data = df2010,
                      subset = train, mtry = i)
  pred <- predict(mod, df2010.test, type = 'class')
  mtry.errors[i] <- mean(pred == region2010.test)
}
```

```
plot(mtry.errors)
```



1, 7, 8, 9 or 10 are the best number of trees. 10 is boosting so try 7-9 trees with different ntree?

```
set.seed(1)
tune <- function (m){
  errors <- c()
  ntree <- c(100, 250, 500)
  n <- length(ntree)
  for (i in 1:n){
    mod <- randomForest(region ~ ., data = df2010,
                        subset = train, mtry = m, ntrees = ntree[i])
    pred <- predict(mod, df2010.test, type = 'class')
    errors[i] <- mean(pred == region2010.test)
    }
  return(errors)
}
```

```
errors2 <- tune(7)
errors2
```

```
## [1] 0.6363636 0.6363636 0.6363636
```

```
errors3 <- tune(8)
errors3
```

```
## [1] 0.6818182 0.6363636 0.6363636
```

```
errors4 <- tune(9)
errors4
```

```
## [1] 0.6363636 0.6818182 0.6363636
```

```
set.seed(1)
bestmod <- randomForest(region ~., data = df2010,
                        subset= train, mtry = 8, ntree= 250,
                        importance = TRUE)
bestmod
```

```
##
## Call:
##  randomForest(formula = region ~ ., data = df2010, mtry = 8, ntree = 250,      importan
ce = TRUE, subset = train)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 46.88%
## Confusion matrix:
##             EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA class.error
## EAP           0   0         0   0    3          0   0  1.00000000
## ECA           0   3         1   0    0          0   0  0.25000000
## EURO ZONE     0   0        12   1    0          0   0  0.07692308
## LAC           1   0         1   0    1          0   0  1.00000000
## MENA          2   1         0   0    2          0   0  0.60000000
## SOUTH ASIA    0   1         0   0    0          0   0  1.00000000
## SSA           0   0         1   1    1          0   0  1.00000000
```

```
bestpred <- predict(bestmod, df2010.test, type = 'class')
mean(bestpred == region2010.test)
```

```
## [1] 0.6363636
```

```
table(bestpred, region2010.test)
```

```
##             region2010.test
## bestpred     EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP          0   0         0   0    1          1   0
##   ECA          1   1         0   0    0          1   1
##   EURO ZONE    0   1        11   0    0          0   0
##   LAC          0   0         0   1    0          0   0
##   MENA         0   1         0   0    1          0   1
##   SOUTH ASIA   0   0         0   0    0          0   0
##   SSA          0   0         0   0    0          0   0
```

```
countries <- data2010$country[-train]
treelabs2010 <- data.frame(countries, bestpred, region2010.test)
```

```
varImpPlot(bestmod)
```