# 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

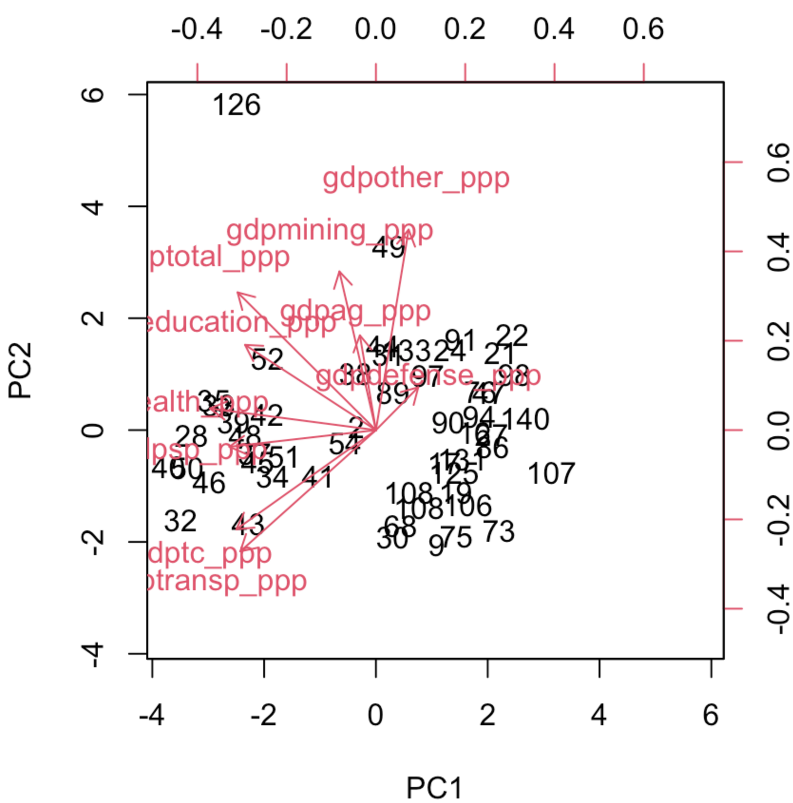
Remove non-numeric columns:

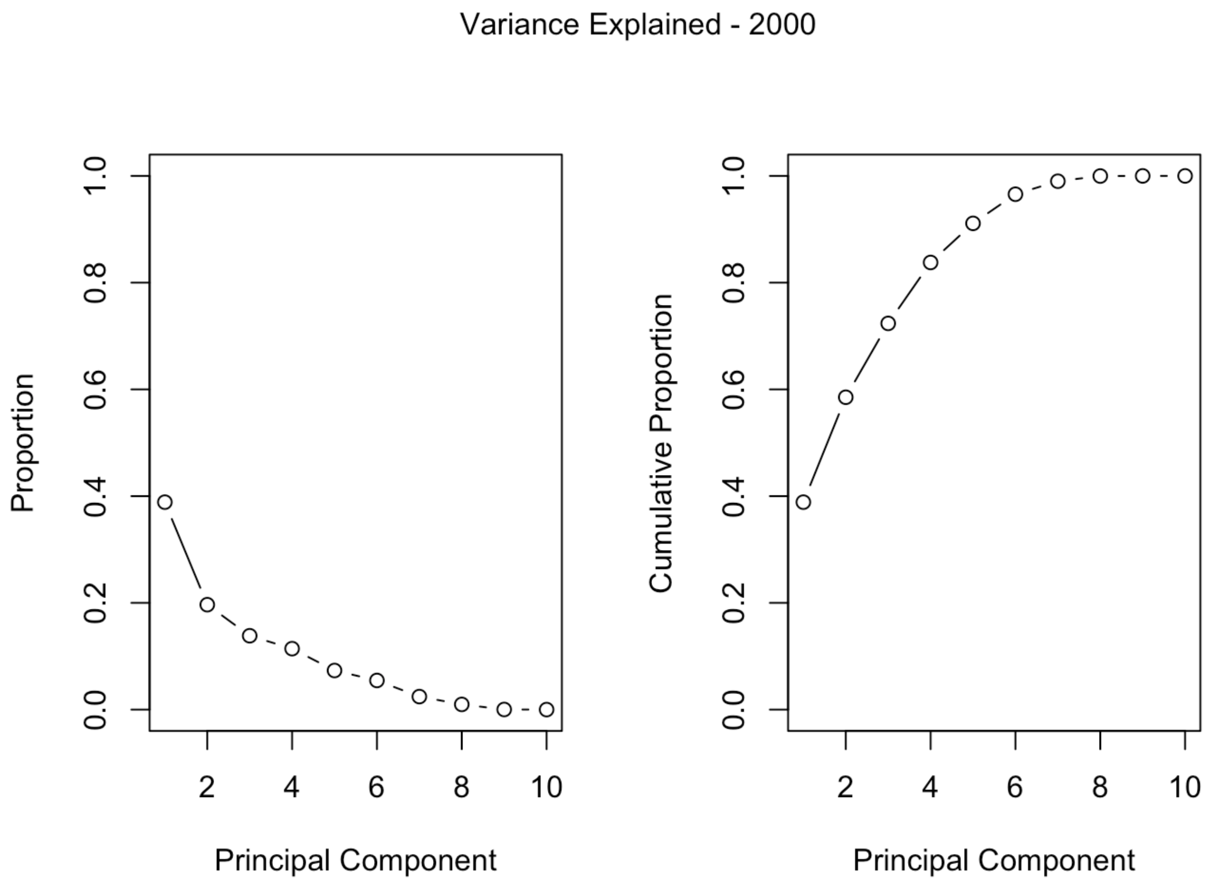
```
df2000num <- subset(data2000, select = -c(country, region))
df2010num <- subset(data2010, select = -c(country, region))
```

```
# 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")
```

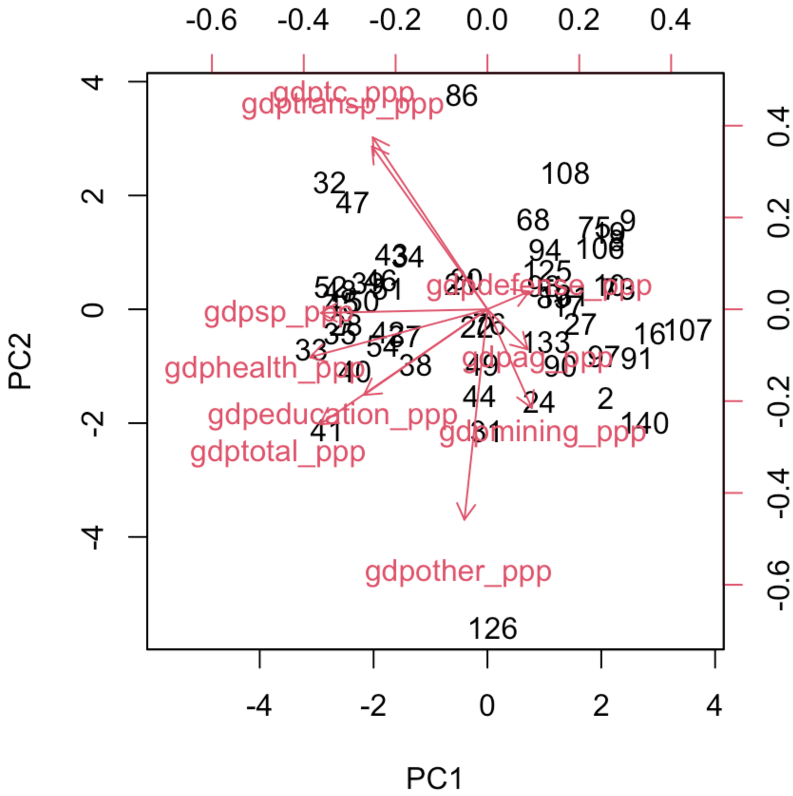
```
##                         PC1         PC2         PC3         PC4         PC5
## gdpag_ppp       -0.04495740  0.26508496 -0.62247186 -0.03819963 -0.60911264
## gdpeducation_ppp -0.36648794  0.23850465  0.01045602 -0.35559938 -0.29368451
## gdphealth_ppp   -0.46441588  0.06037977  0.18624913  0.11573209 -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
```



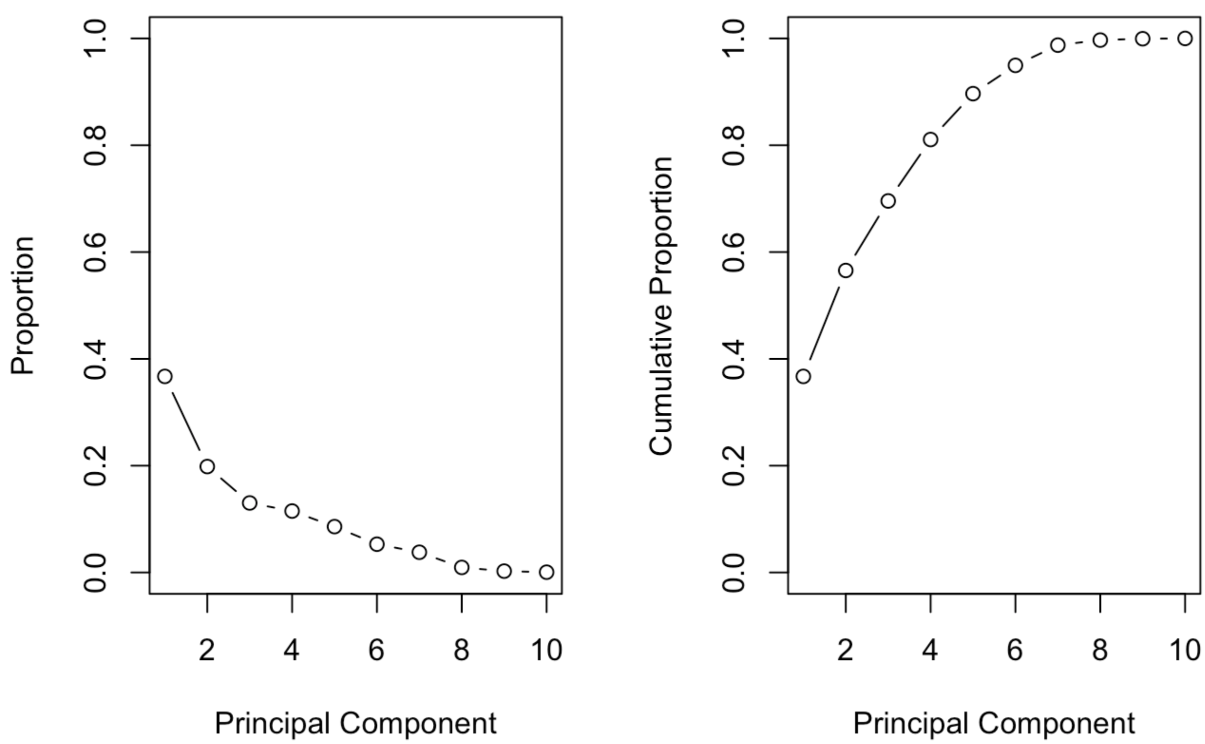



```
# 2010 data
perform_pca(df2010num, "2010")
```

```
##                        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

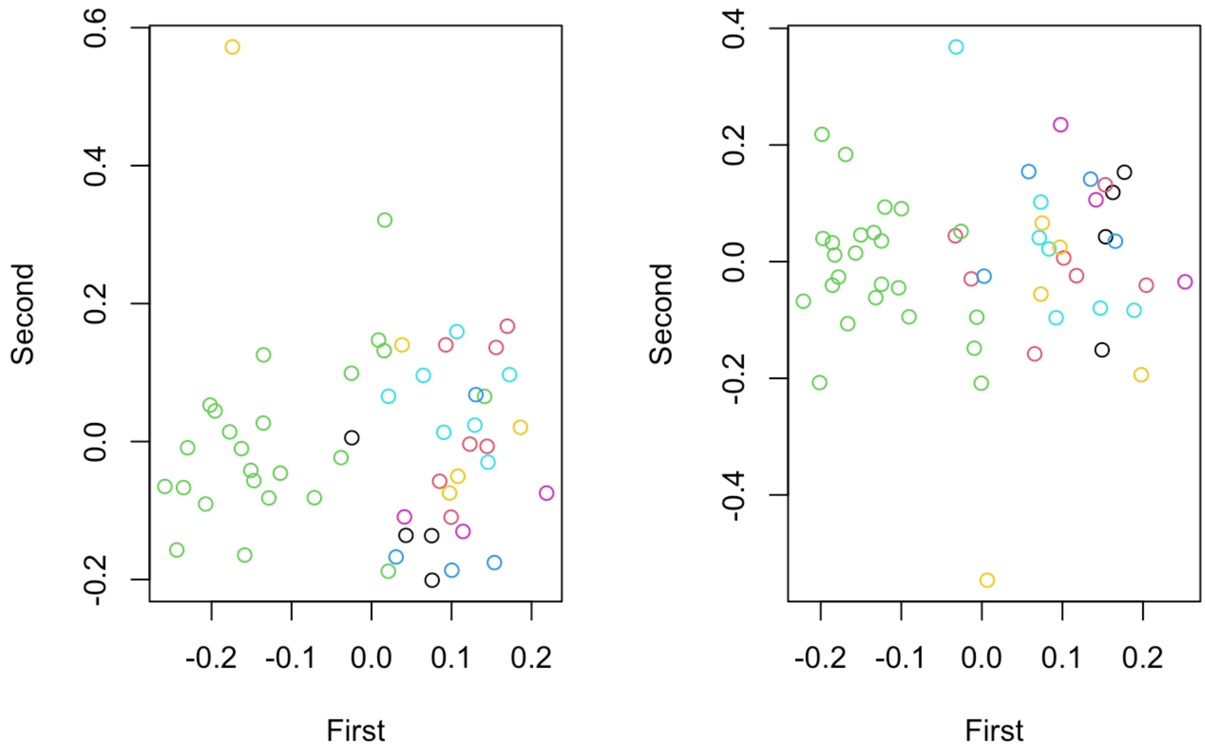
```
# 2000
dfscale00 <-  scale(df2000num) # scale data
s00 <- svd(dfscale00)          # perform SVD
data.frame(s00$v)              # loadings (same as PCA)
```

```
##             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")
```

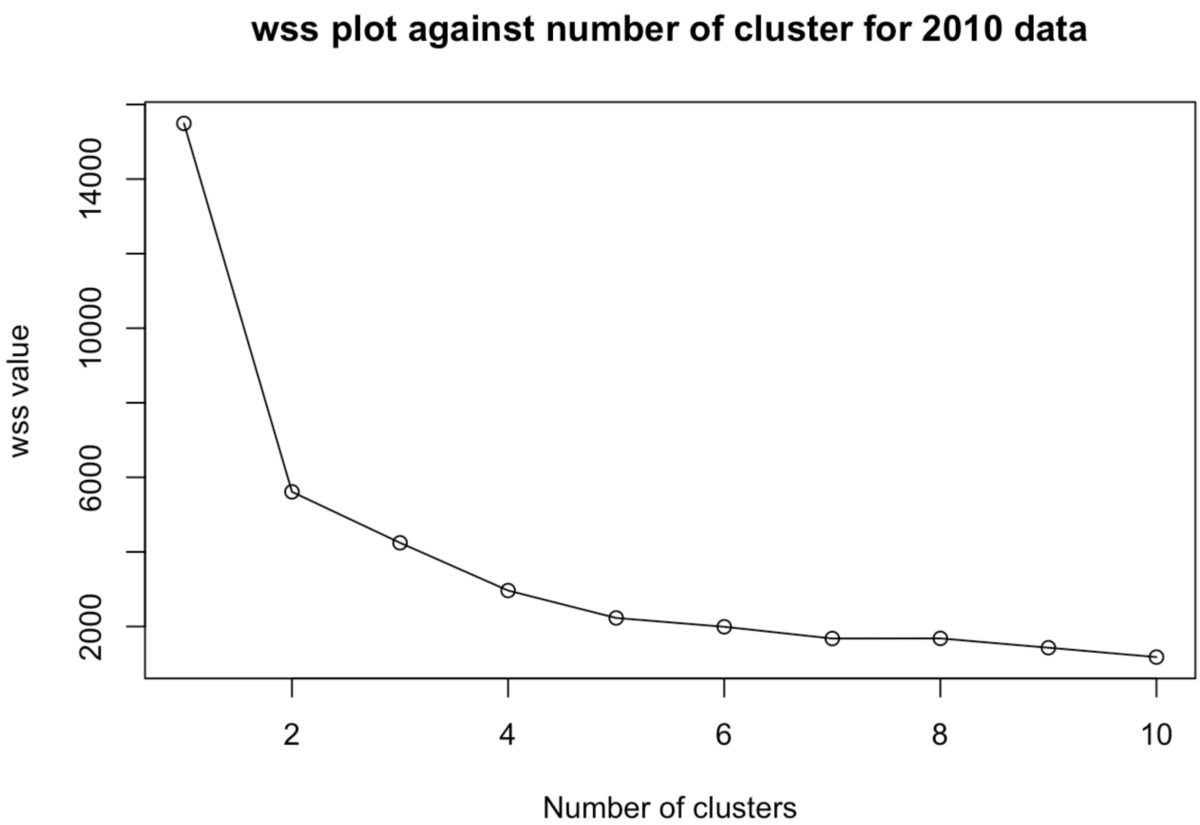



###Clustering

## K-means Clustering- 2010

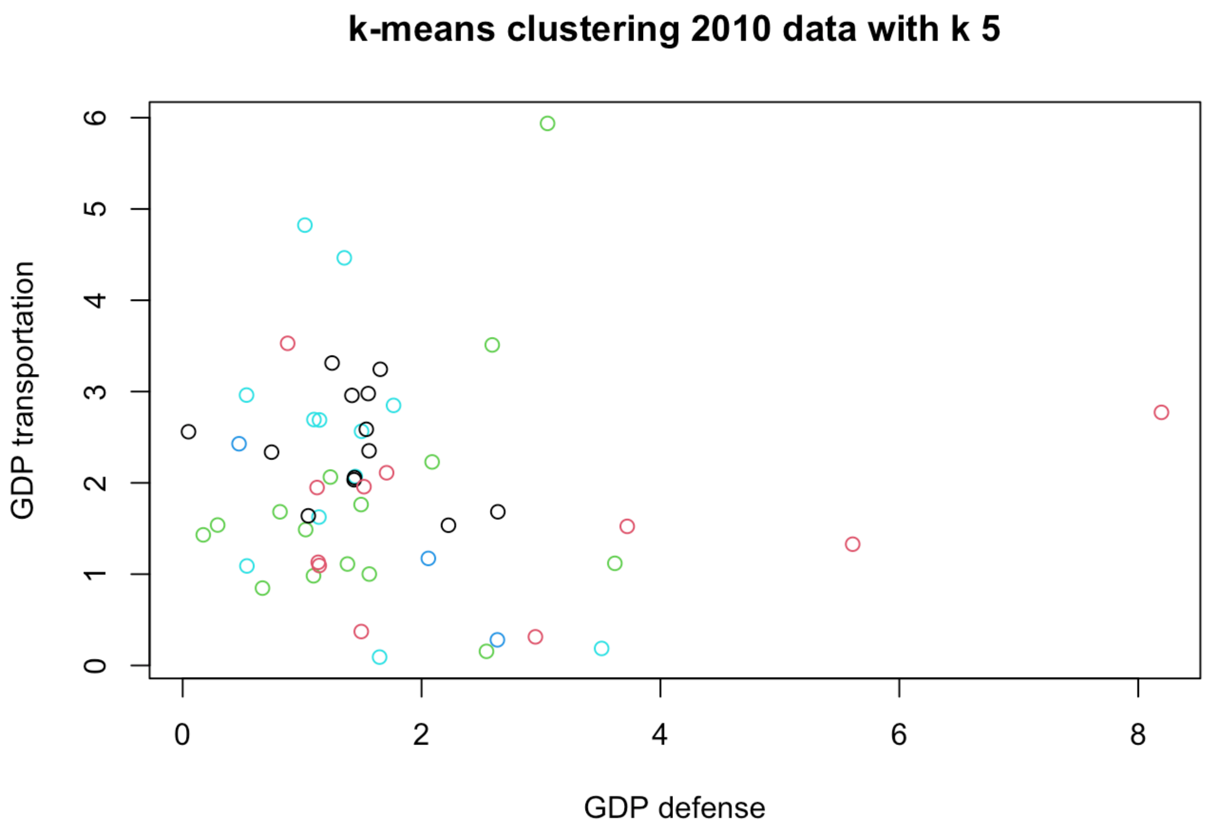
Wss means against k cluster to decide how many cluster to use for analysis

```
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")
```



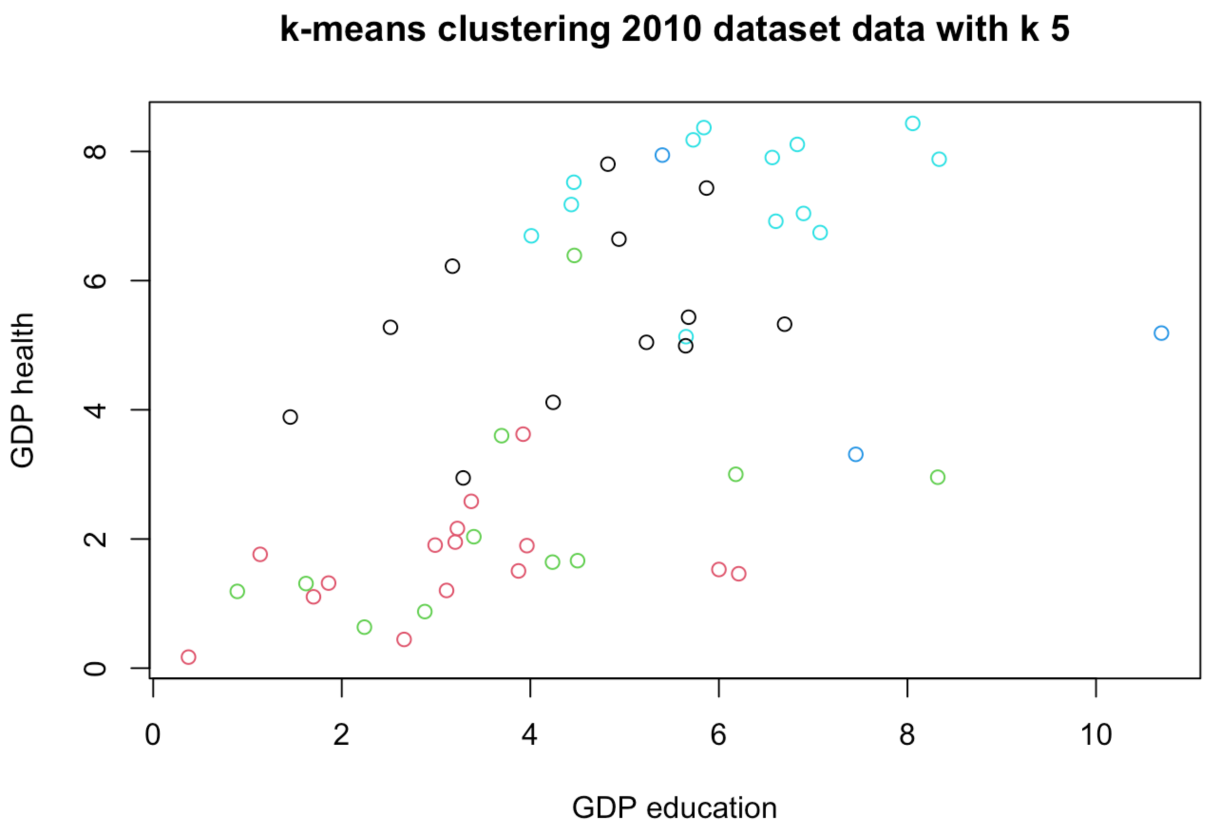

Use 5 clusters for both as the wss is reasonable at that point before dropping off. Since wss will keep on dropping till each observation has its own cluster; 5 clusters seems a reasonable wss distance

```
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.

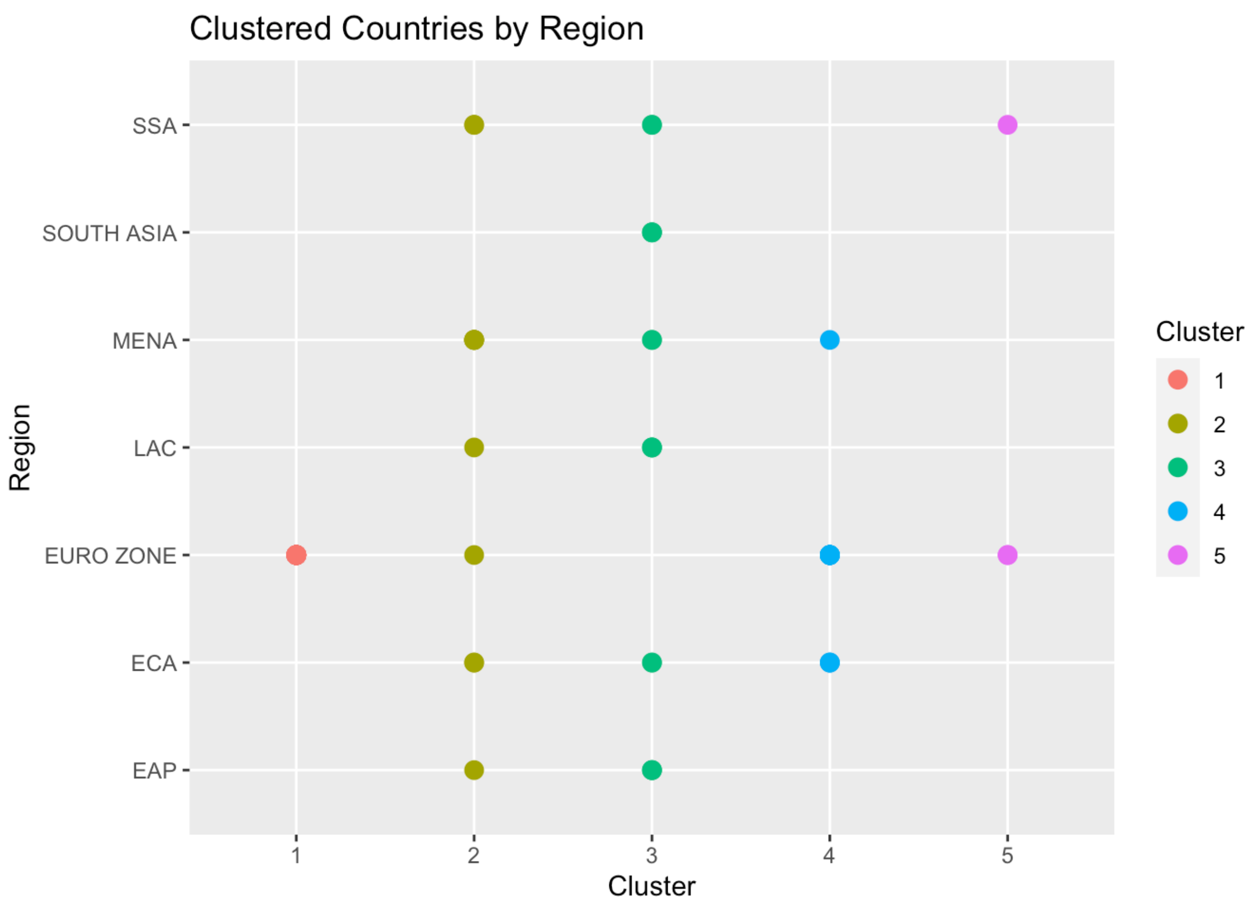
```
country_cluster <- data.frame(country = data2010$country, cluster = km_df2010num$cluster)

# 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…
```

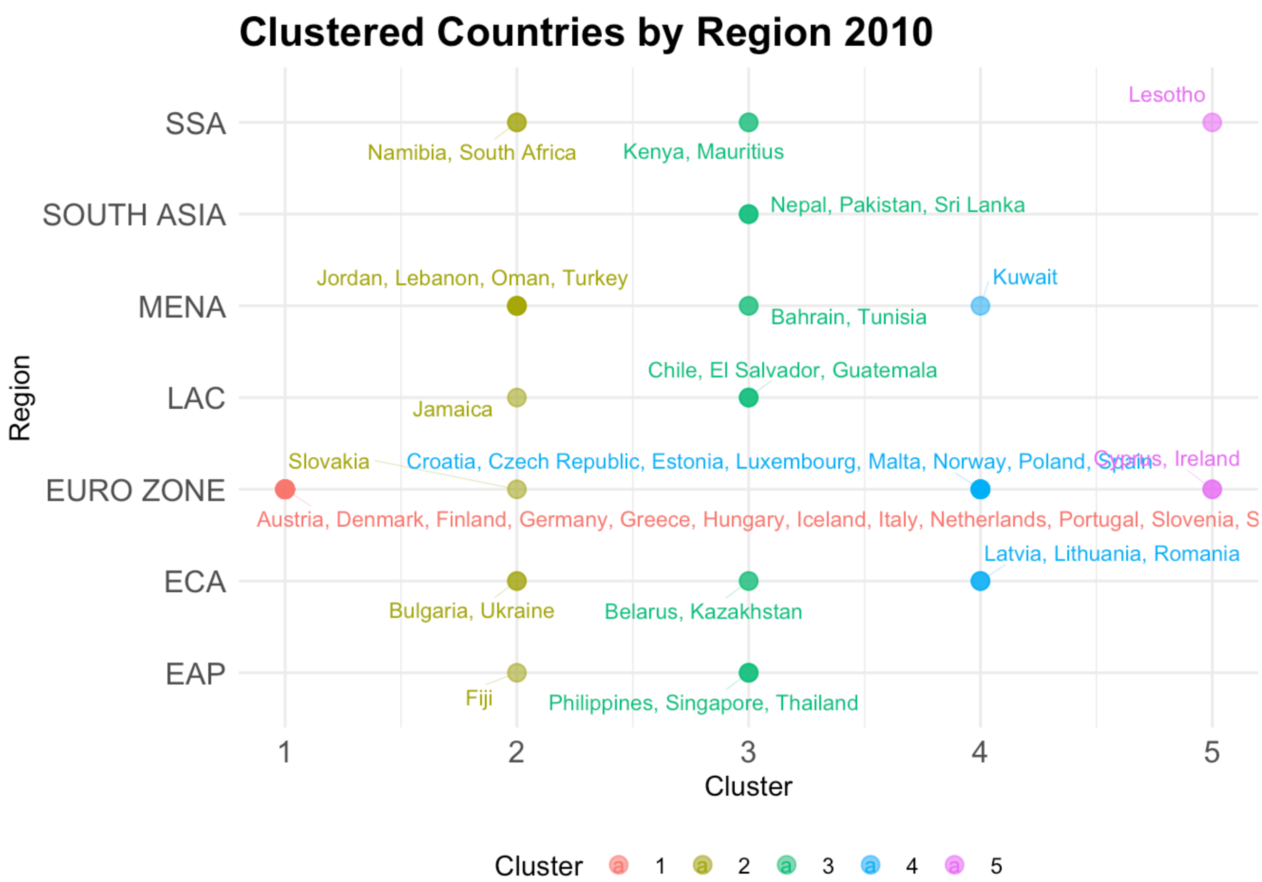
```
km_df2010num <- kmeans(df2010num, 5, nstart = 20)

# 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 = ", "))
```

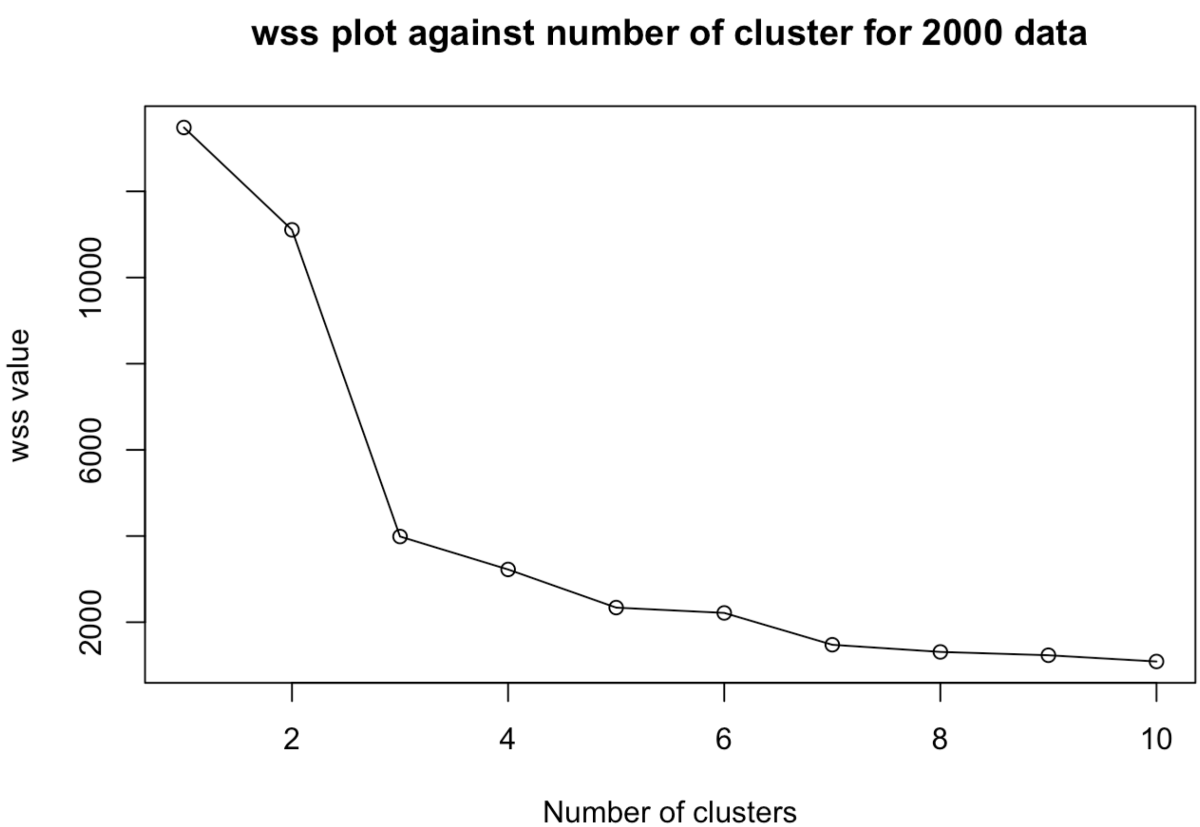
```
## `summarise()` has grouped output by 'cluster'. You can override using the
## `.groups` argument.
```

```
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

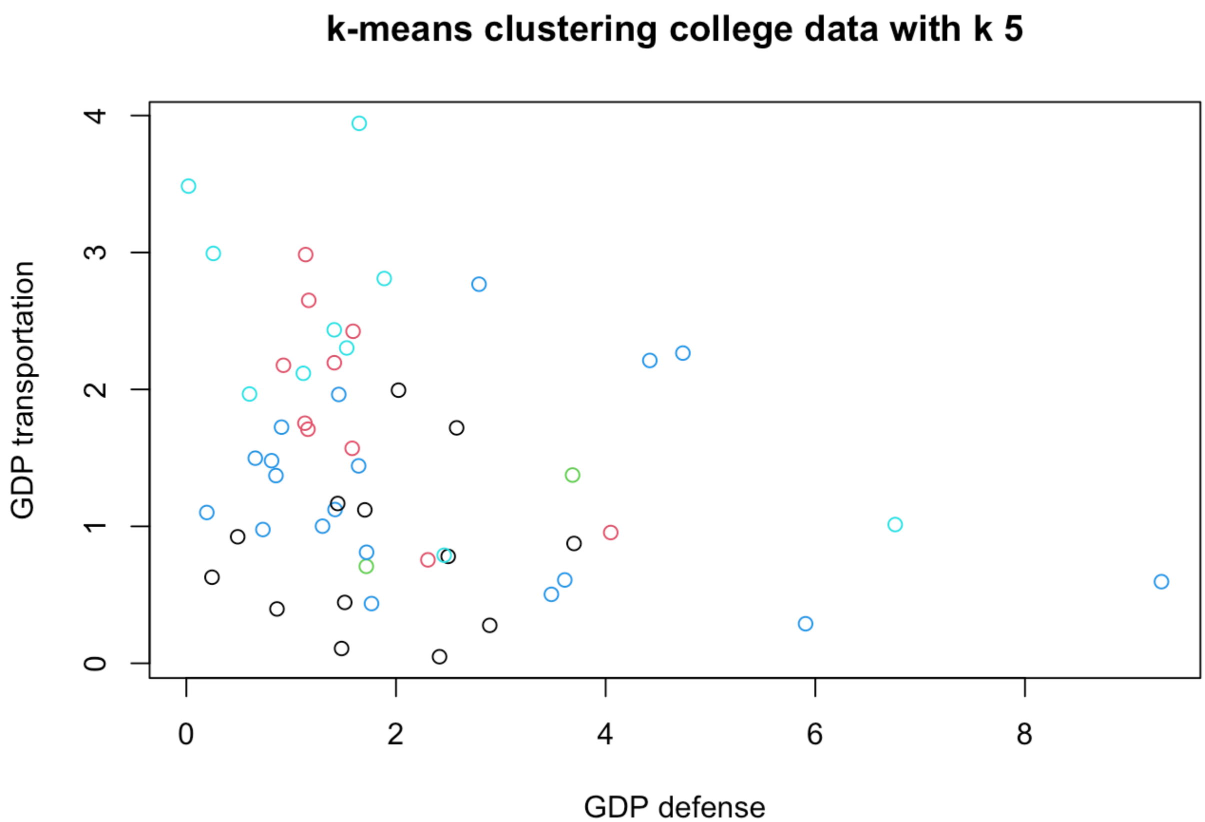
```
set.seed(1)
wss<- NULL
for (i in 1:10){
  fit = kmeans(df2000num,centers = i)
  wss = c(wss, fit$tot.withinss)
}
plot(1:10, wss, type = "o", main = 'wss plot against number of cluster for 2000 data', xla
b = "Number of clusters", ylab = "wss value")
```




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")
```



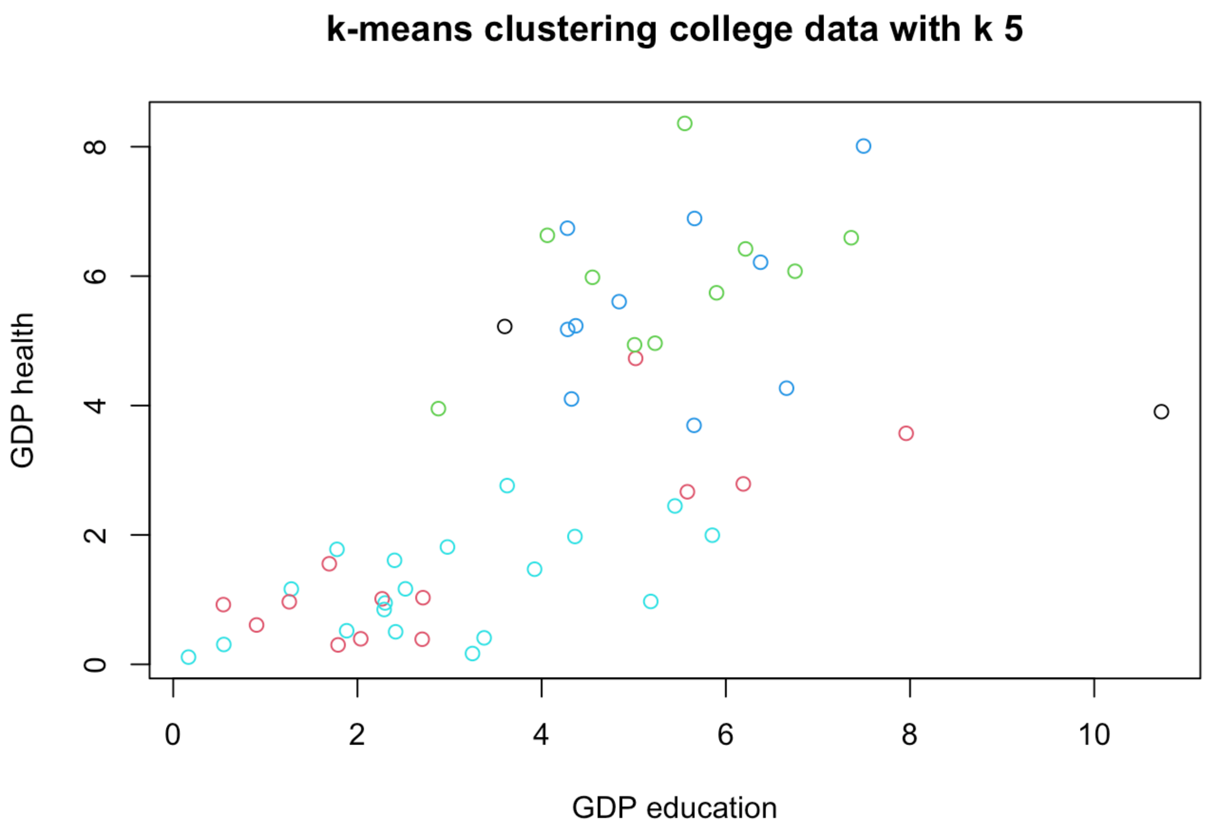

```
    #xlim = c(min(df2010$country), max(df2010$country)))
```

```
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)
```

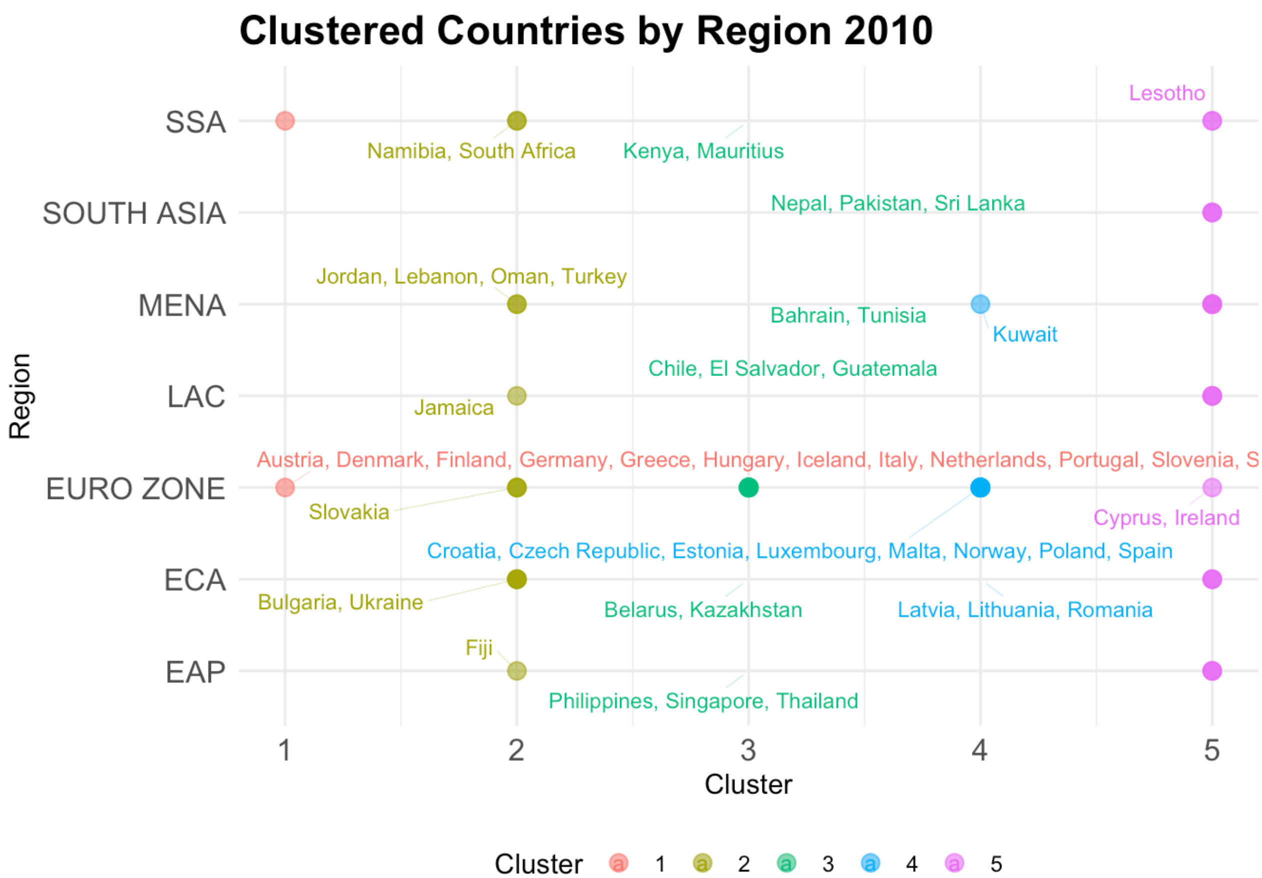
```
##             country cluster
## 2              Fiji       2
## 8       Philippines       5
## 9         Singapore       5
## 10         Thailand       5
## 16          Belarus       5
## 17         Bulgaria       2
## 19       Kazakhstan       5
## 21           Latvia       2
## 22        Lithuania       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")
```

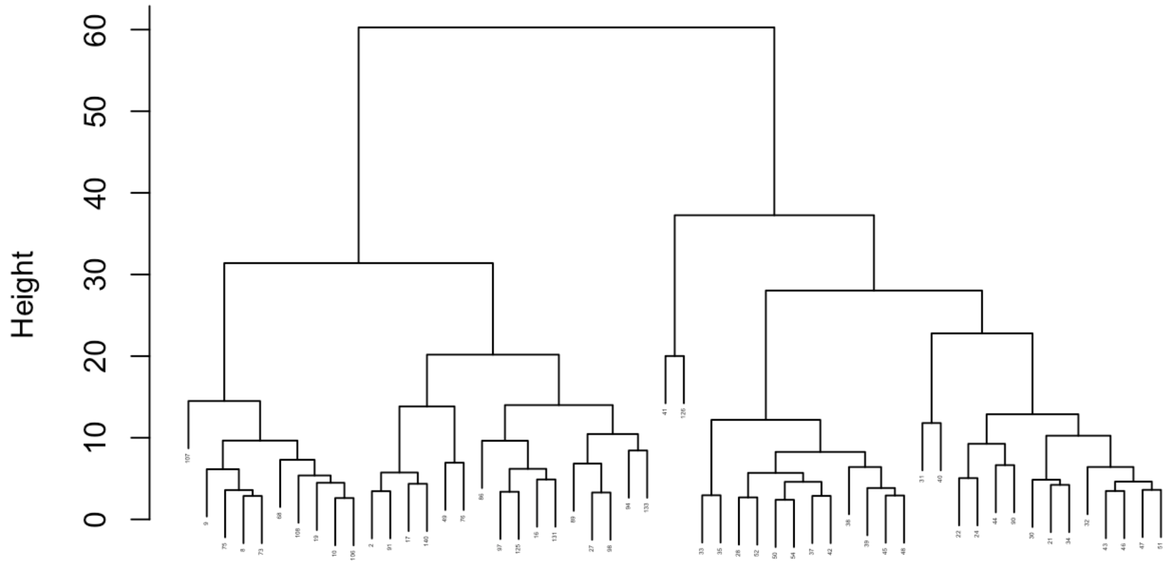



## Hierarchical Clustering- 2010

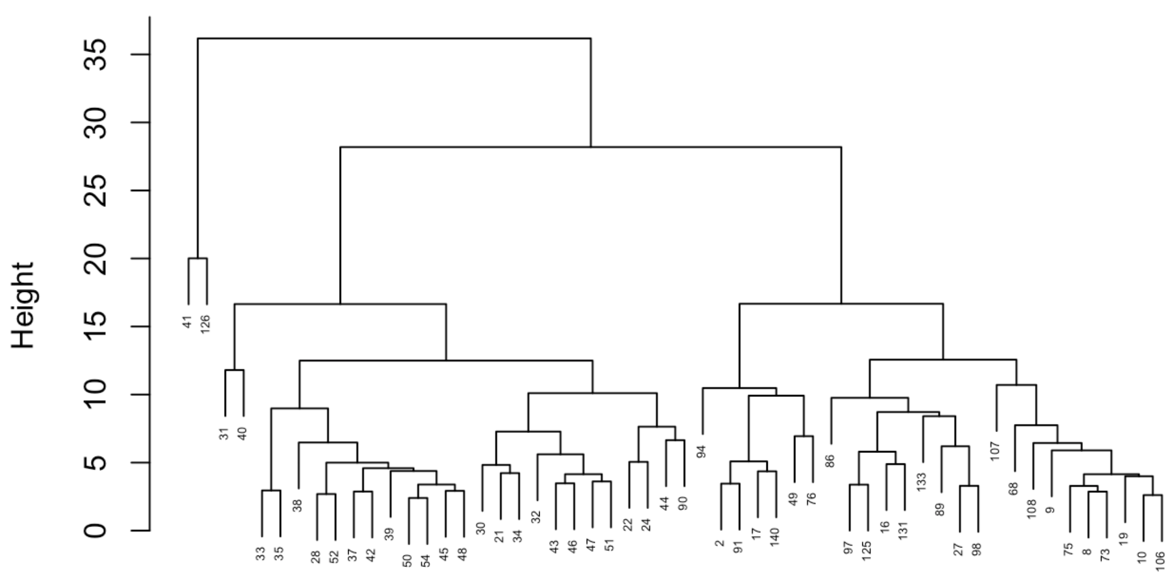
```
hcluster <- function(df){

    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)
}
```

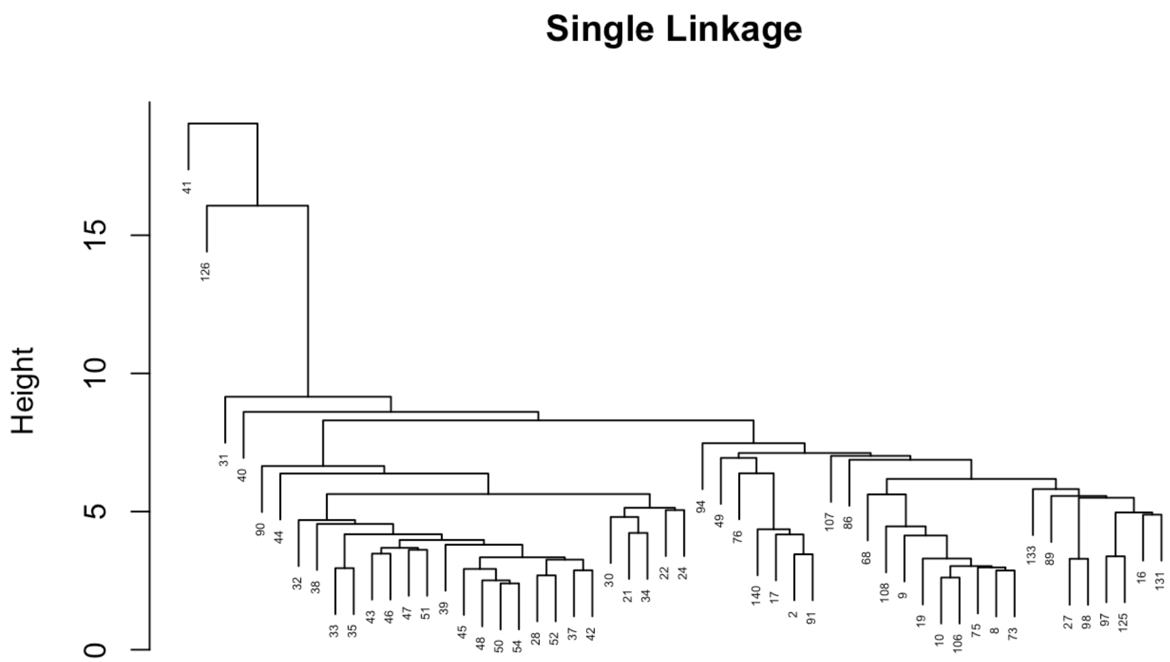
```
hcluster(df2010num)   #call the function
```

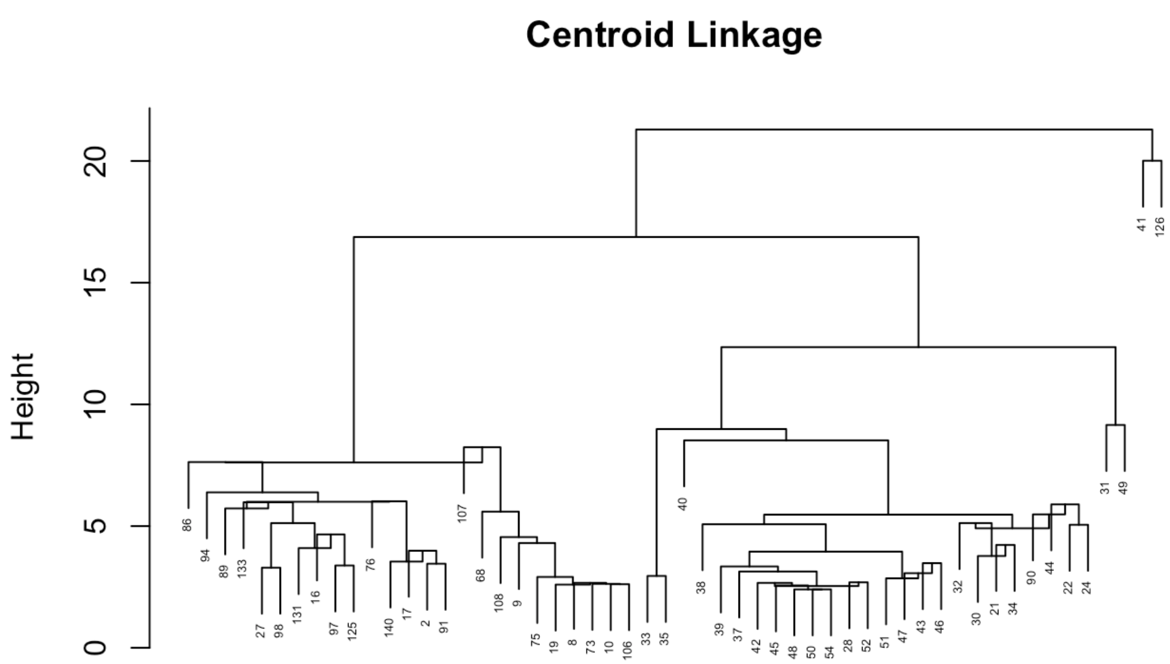
## Complete Linkage




## Average Linkage



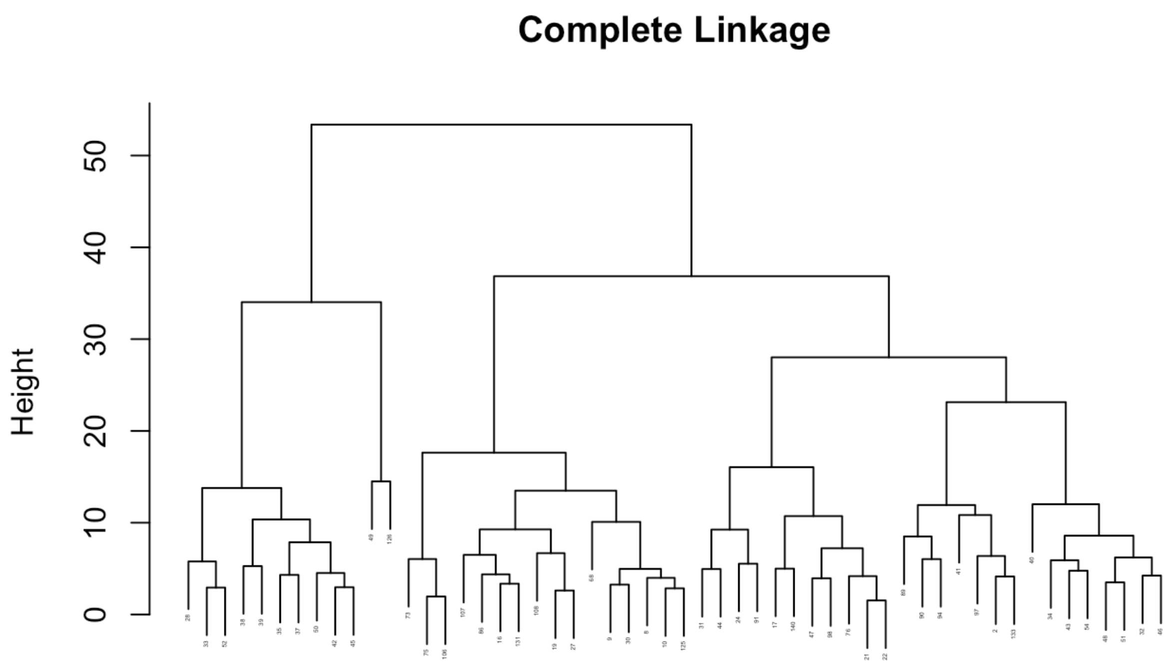


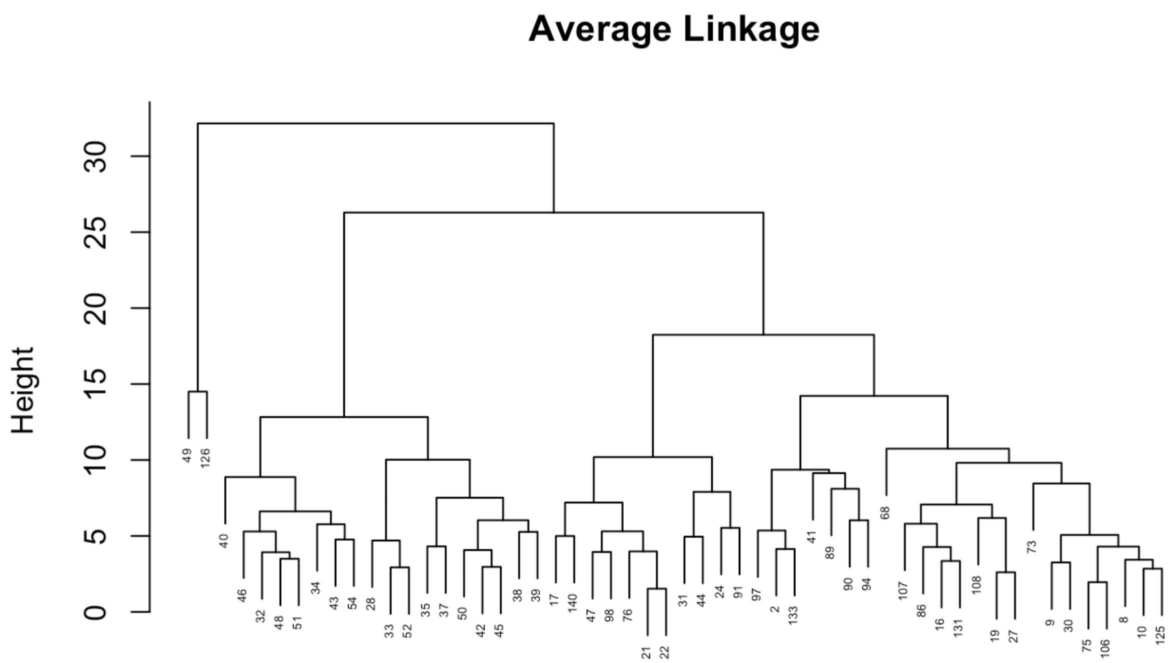



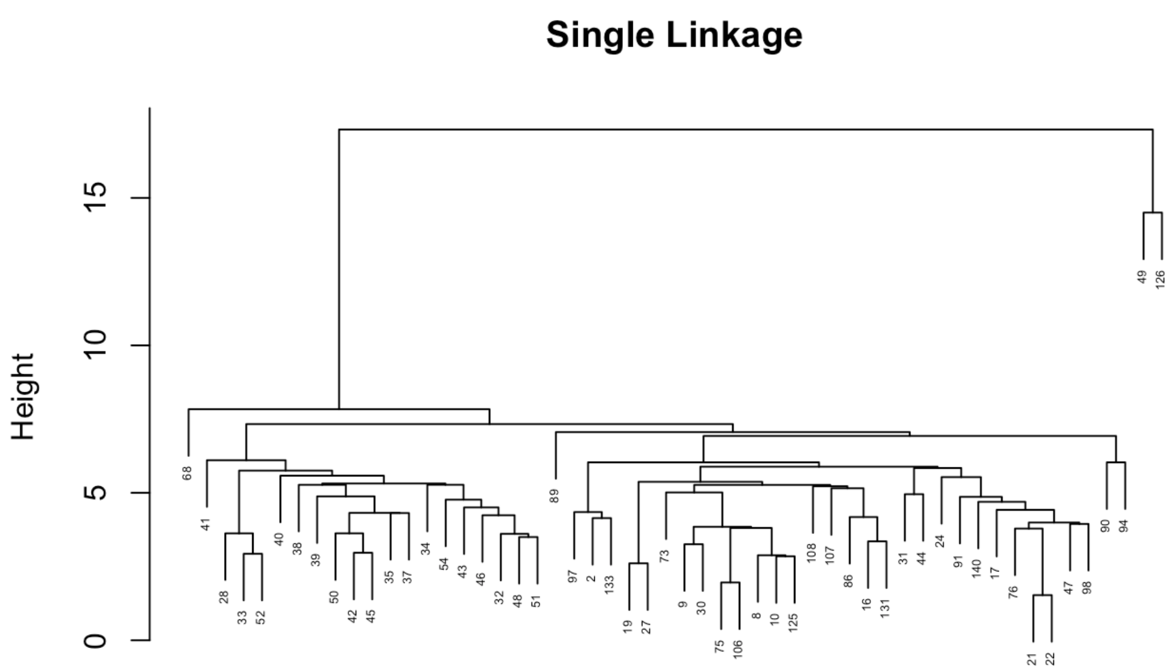

Among all options for the htree, complete linkage had the most even split so using complete linkage with 5 as the cut, we have 4 clusters for 2010 data.

## Hierarchical Clustering - 200

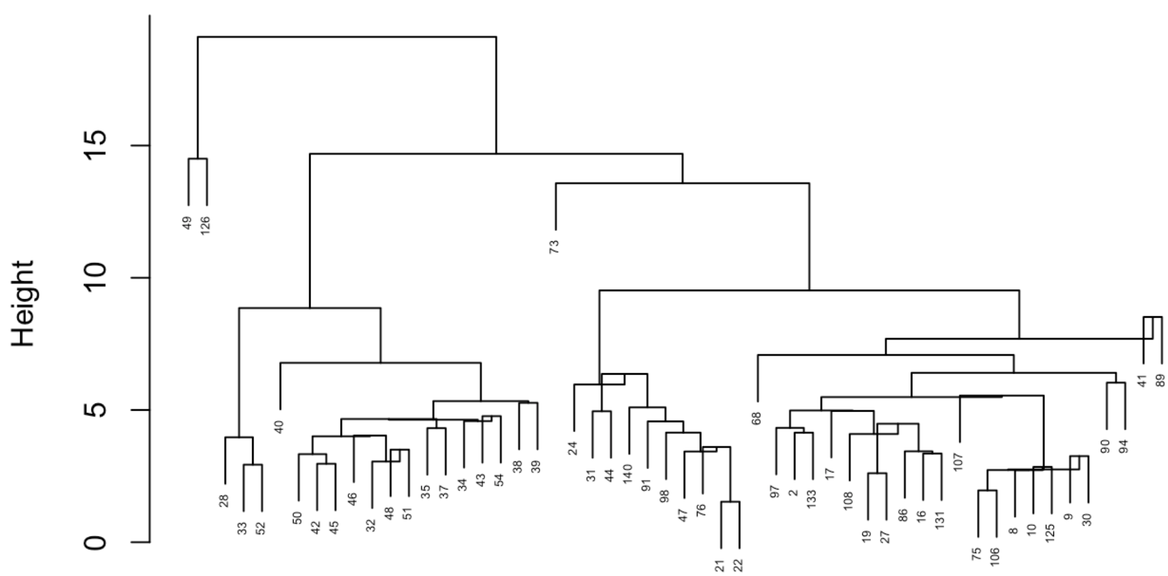
```
hcluster(df2000num)  #call the function
```




Average Linkage
Height
0
5
10
15
20
25
30
49 126 40 46 32 48 51 34 43 54 28 33 52 35 37 50 42 45 38 39 17 140 47 98 76 21 22 31 44 24 91 97 2 133 41 89 90 94 68 107 86 16 131 108 19 27 73 9 30 75 106 8 10 125


Single Linkage
Height
0
5
10
15
68 41 28 33 52 40 38 39 50 42 45 35 37 34 54 43 46 32 48 51 89 97 2 133 19 27 73 9 30 75 106 8 10 125 108 107 86 16 131 31 44 24 91 140 17 76 21 22 47 98 90 94 49 126

## Centroid 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")

}
```

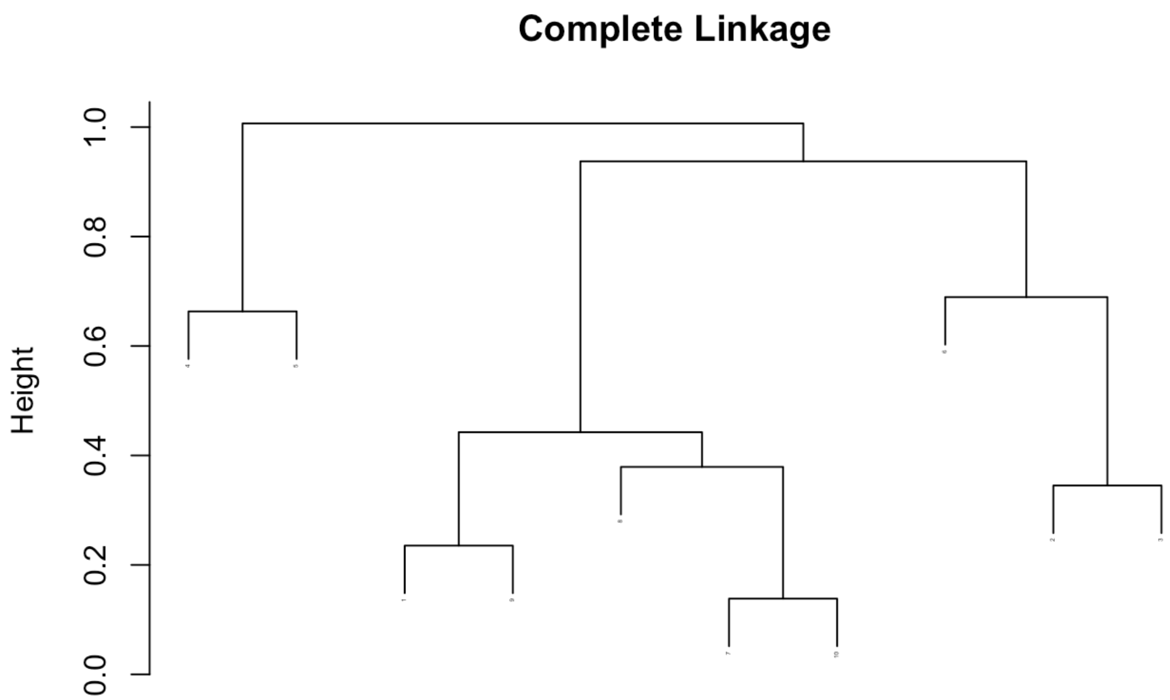
```
## Cluster 1 :
## Fiji, Bulgaria, Latvia, Lithuania, Romania, Cyprus, Czech Republic, Estonia, Iceland, I
reland, Luxembourg, Malta, Norway, Poland, Portugal, Spain, United Kingdom, Jamaica, Jorda
n, Kuwait, Lebanon, Oman, Tunisia, Turkey, Namibia, South Africa
##
## Number of countries in Cluster 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
```

```r
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)
```

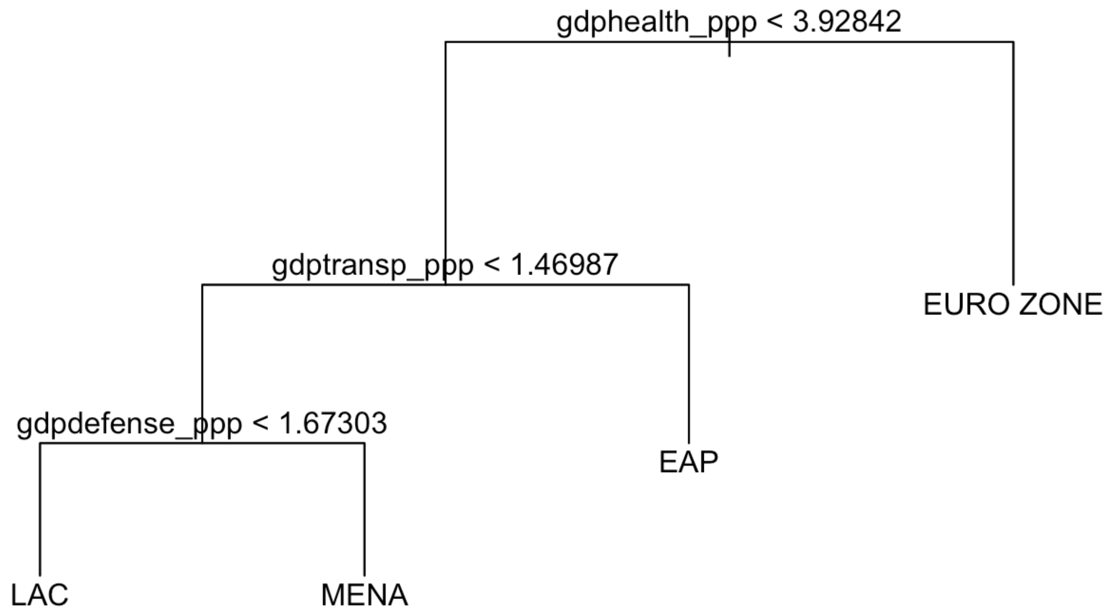
```
## [1] 0.5909091
```

```
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)
```



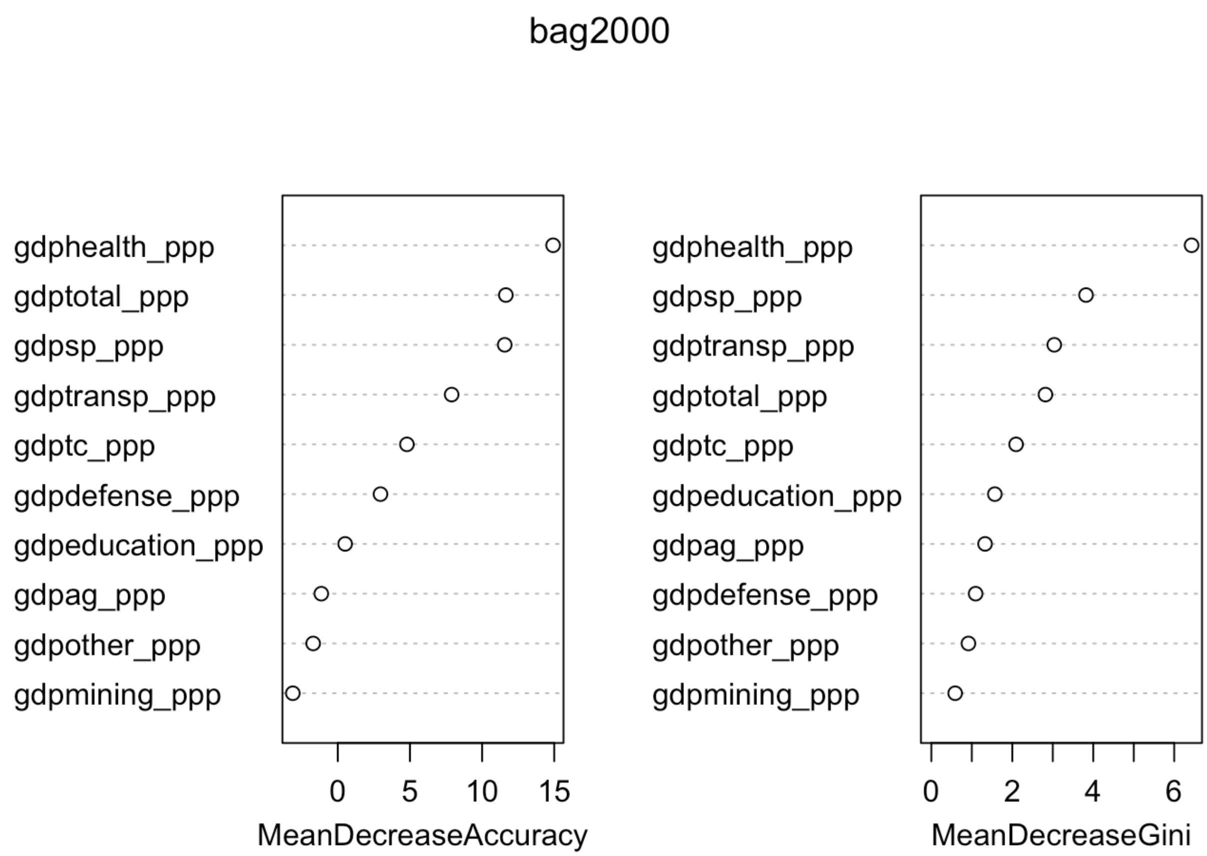

####

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
```

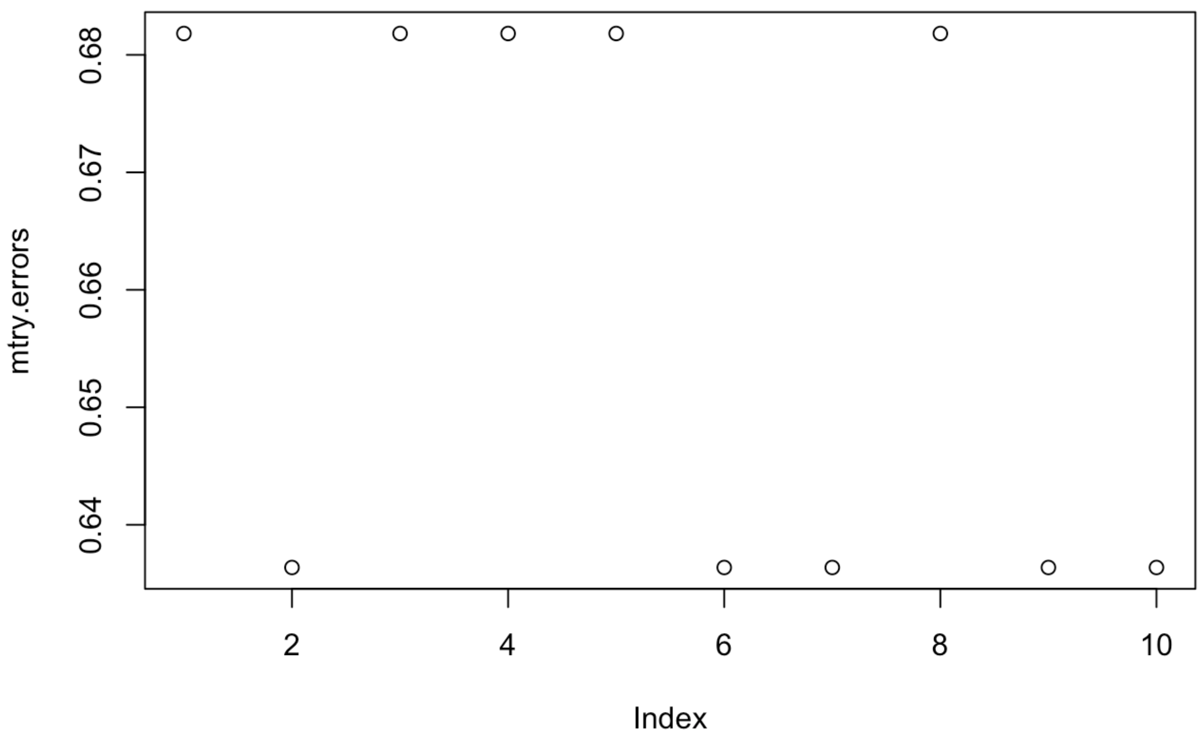
```
table(bag.pred, region2000.test)
```

```
##               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

```r
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)
}
```

```r
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?

```r
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)
```

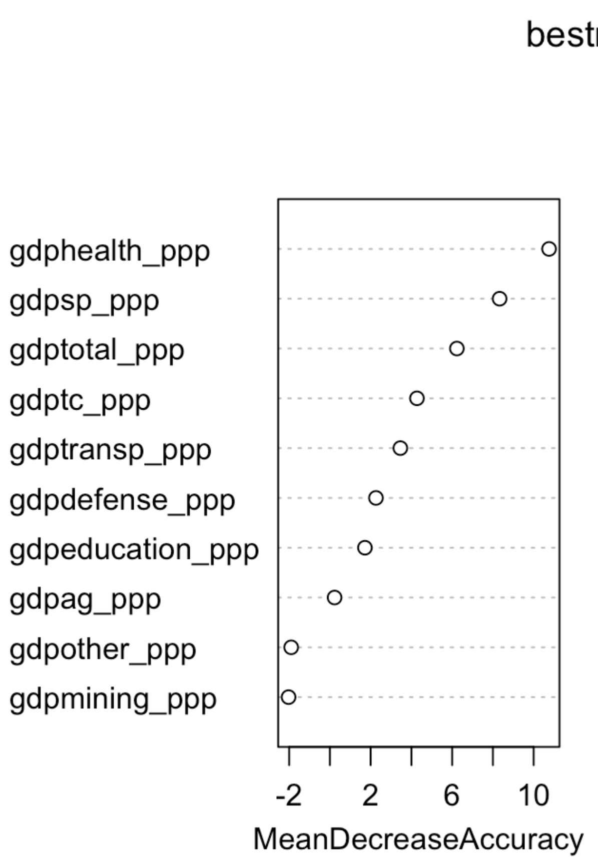
```
## [1] 0.6818182
```

```
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)
```

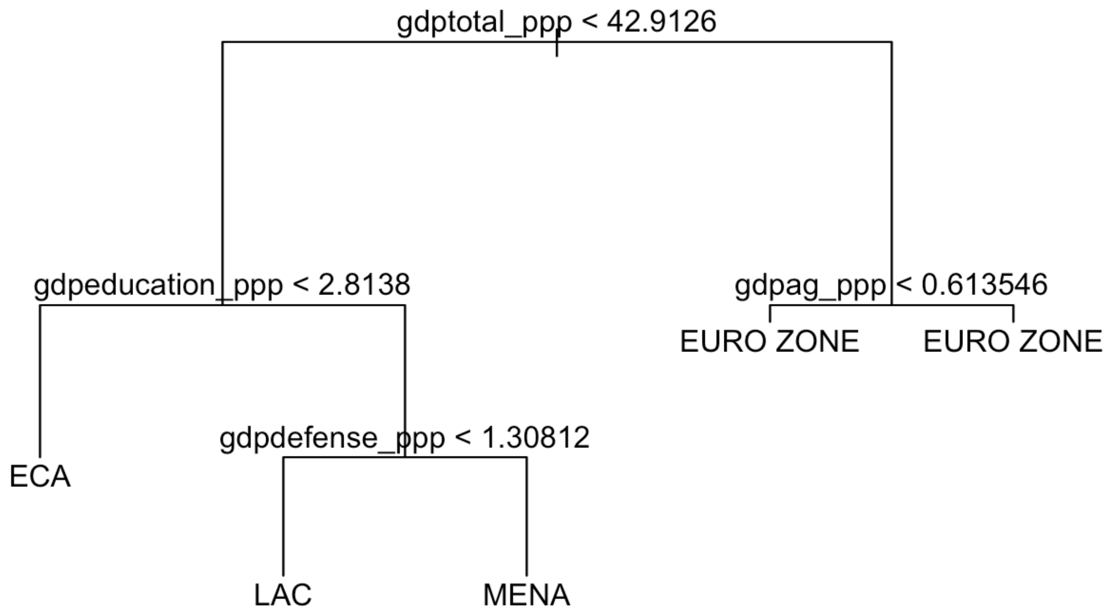
```
## [1] 0.5454545
```

```
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)
```

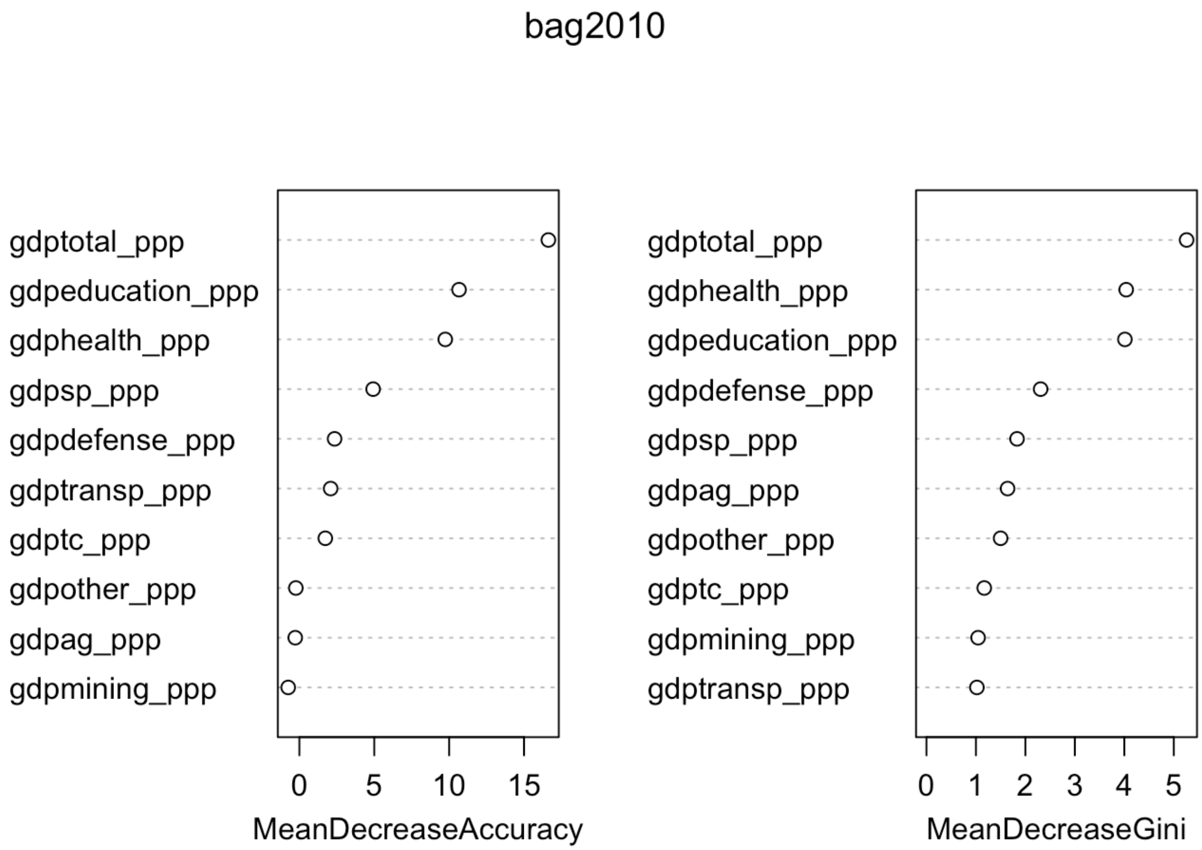



## Bagging

```
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)
```

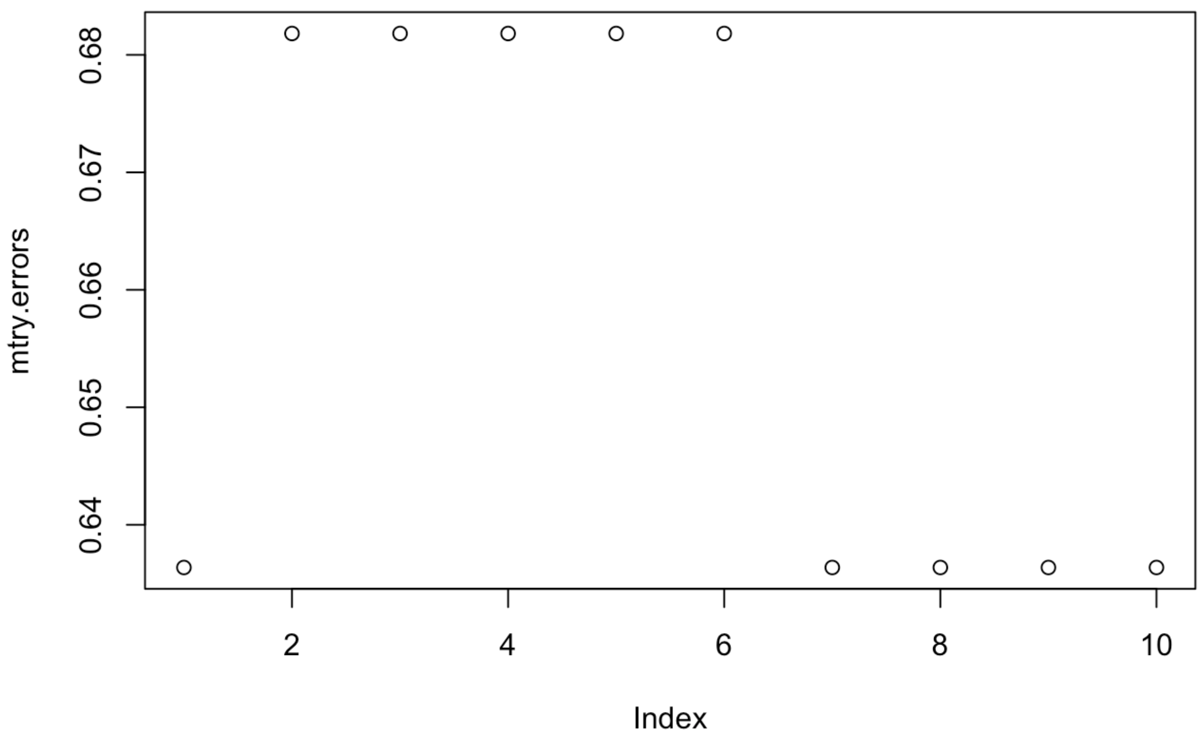
```
##              region2010.test
## bag.pred      EAP ECA EURO ZONE LAC MENA SOUTH ASIA SSA
##   EAP           0   0         0   0    1          1   0
##   ECA           1   1         0   0    0          1   1
##   EURO ZONE     0   1        11   0    0          0   0
##   LAC           0   0    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

```r
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)
}
```

```r
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?

```r
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)
```