EDA Report On World Value Survey Data

Kerui Cao

10/19/2019

Introduction to WVS

The World Values Survey (www.worldvaluessurvey.org) is a global network of social scientists studying changing values and their impact on social and political life.

The survey, which started in 1981, seeks to use the most rigorous, high-quality research designs in each country. The WVS consists of nationally representative surveys conducted in almost 100 countries which contain almost 90 percent of the world's population, using a common questionnaire. The WVS is the largest non-commercial, cross-national, time series investigation of human beliefs and values ever executed, currently including interviews with almost 400,000 respondents. Moreover the WVS is the only academic study covering the full range of global variations, from very poor to very rich countries, in all of the world's major cultural zones.

The WVS seeks to help scientists and policy makers understand changes in the beliefs, values and motivations of people throughout the world. Thousands of political scientists, sociologists, social psychologists, anthropologists and economists have used these data to analyze such topics as economic development, democratization, religion, gender equality, social capital, and subjective well-being. These data have also been widely used by government officials, journalists and students, and groups at the World Bank have analyzed the linkages between cultural factors and economic development.

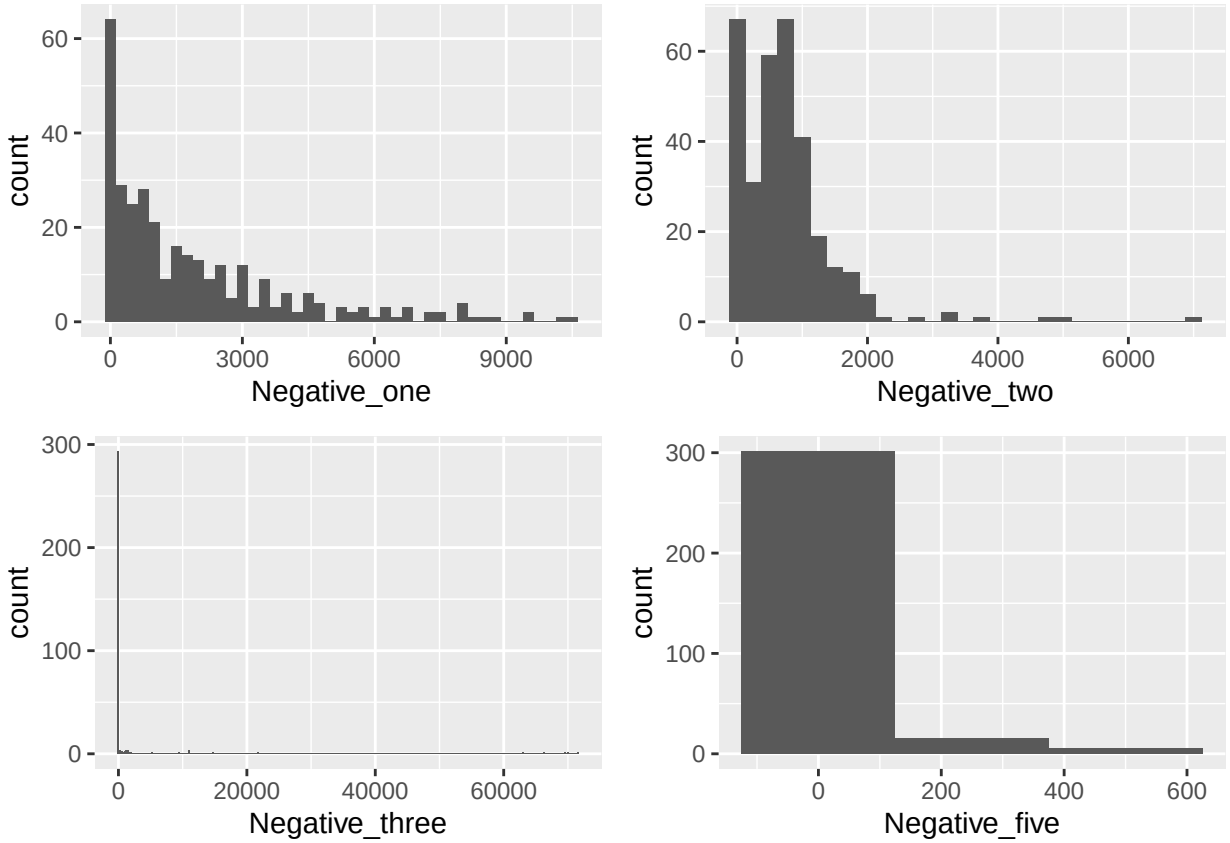
Introduction to Data

I choose the latest public WVS data (V20180912), which is the Wave 6 that carried out during 2010 to 2014. Raw data consists of 89565 0bservations and 440 variables, containing the survey result cross over 50 countries. According to the questionaire provided along the data, variables named after V250 provided information that is not related to world value.

Data Cleaning Process

I take the first step of data cleaning as Missing Value Analysis, missing value in questionaire data can be a complacate situation, because the cause of missing value of can be various, whether the respondent do not want to answer, do not know how to answer, and so on, luckily according to the description of the survey data, it tells us that different cause of missing value were coded into different numbers, such as -1 represents do not know, -2 represents no answer, -3 represents not applicable. For simplicity, we treat all kinds of negtive value as missing. First we need to obtain a big picture of the distribution of missing values.

```
# First we choose the variables from V1 to V250
da %<>% select(V1:V250)
# Account the number of different missing values in each variables.
Negative_one = da %>% apply(MARGIN = 2,function(x){as.numeric(sum(x == -1))})
Negative_two = da %>% apply(MARGIN = 2,function(x){as.numeric(sum(x == -2))})
Negative_three = da %>% apply(MARGIN = 2,function(x){as.numeric(sum(x == -3))})
Negative_five = da %>% apply(MARGIN = 2,function(x){as.numeric(sum(x == -5))})
```



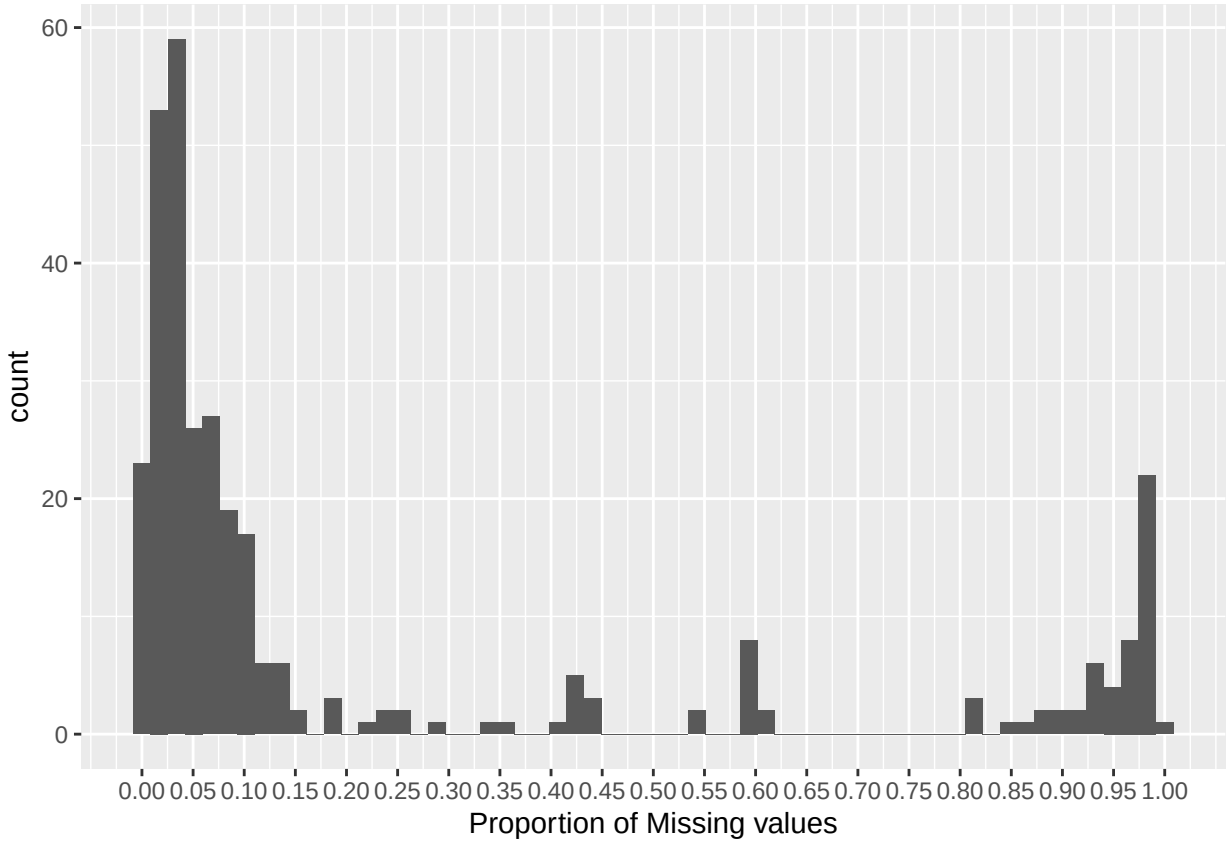
Above histgram show the distribution of different types of missing value over all the variables, we can see that, for missing value type -1, most of the variables have less than 3000 missing values, for all four types of missing value, -1 is the most "popular" one.

Now we want to solve the problem by deleting missing values, we can choose to delete vatiables or delete observations, for here we choose to delete variables containing a large portion of missing values, than delete observations containing a large portion of missing values.

```
# transform all negative values to NA
da = data.frame(apply(da,MARGIN = 2,function(x){x = ifelse(x<0, NA, x)}))

# Calculate proportion of missing value of each variables
NA_NUM = data.frame(da %>% apply(MARGIN = 2,function(x){round(sum(is.na(x))/length(x),2)}))
NA_NUM["name"] = rownames(NA_NUM)
colnames(NA_NUM) = c("Prop","name")

# Plot the distribution of NA of each variables
ggplot() + geom_histogram(aes(x = NA_NUM$Prop), bins = 60) +
  scale_x_continuous(breaks=seq(0,1,0.05)) + xlab("Proportion of Missing values")
```



We can see that most of variables have less than 10% missing values, so we choose to delete all that variables containing more than 10% missing values, then we delete observations containing missing values.

```
# delete variables with nore than 10% NA
NA_NUM %<>% filter(Prop<=0.1)
da %<>% select(NA_NUM[,2])

# V1,V2A,V3 are code for questionaire
da %<>% select(-V1,-V2A,-V3)

#delete NA observations
da = na.omit(da)
```

After solving the missing value problem, we still have 18445 observations and 215 variables.

EDA Process

Below is the actual meaning of the first nine variables:

- V2: Country Code
- V4: How important is family? 1 to 4 represents very important, rather important, not very important and not at all important.
- V5: How important is friends? 1 to 4 represents very important, rather important, not very important and not at all important.

- V6: How important is leisure time? 1 to 4 represents very important, rather important, not very important and not at all important.
- V7: How important is politics? 1 to 4 represents very important, rather important, not very important and not at all important.

- V8: How important is work? 1 to 4 represents very important, rather important, not very important and not at all important.

- V9: How important is religion? 1 to 4 represents very important, rather important, not very important and not at all important.

- V10: How happy the respondent is? 1 to 4 represents very happy, rather happy, not very happy and not at all happy.

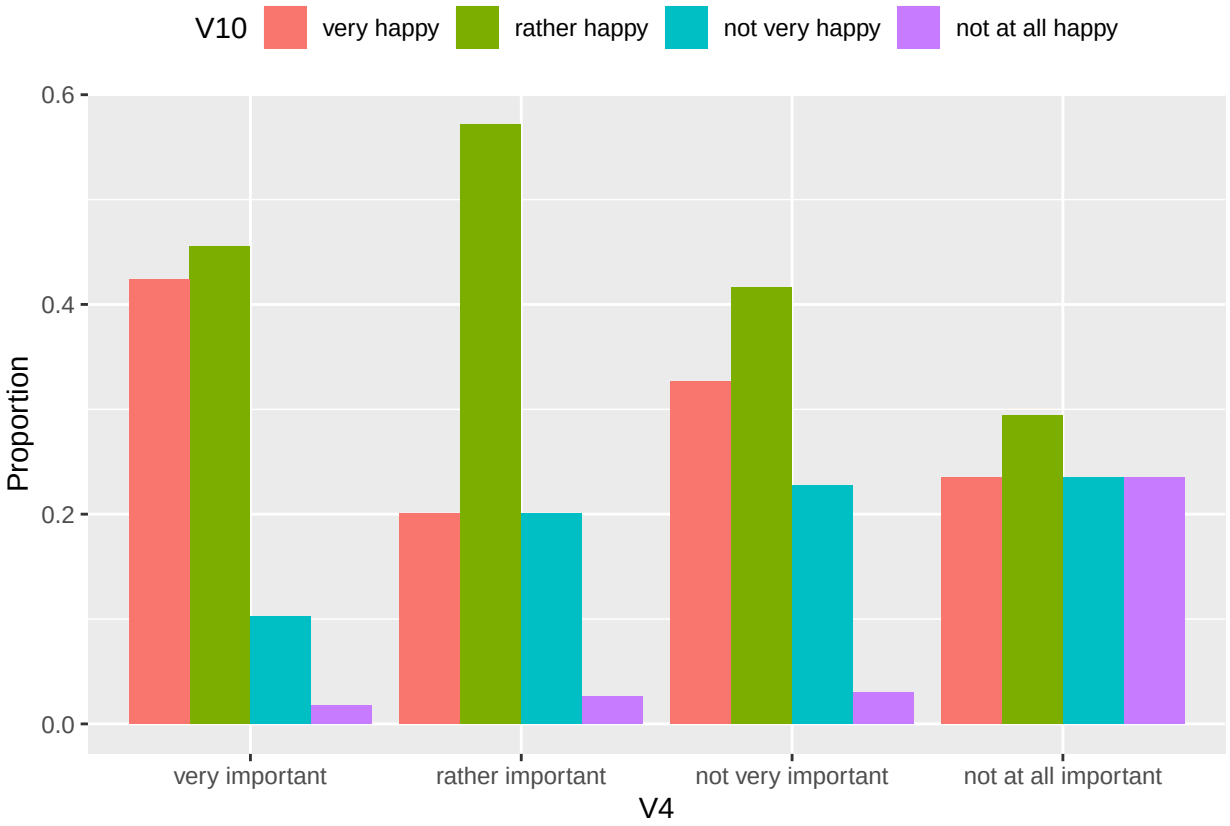
Cross Table Analysis

One important way to analyze questionaire data is to do cross analysis, for example as the variables V4 and V10, we can analyze how many people who thinks family is important is happy about their life:

```
# Generate Cross Table
cross = xtabs(data = da,~V4+V10)
# Transform table into dataFrame
crossdf = data.frame(cross)
# reorganize dataframe
cross1 =crossdf %>% pivot_wider(names_from = V4, values_from = Freq)
# rename data frame
colnames(cross1) = c("V10",paste("V4=",c(1,2,3,4),sep = ""))
```

Table 1: Cross Analysis (Freqency)

V10	V4=1	V4=2	V4=3	V4=4
1	7421	166	33	4
2	7974	473	42	5
3	1802	166	23	4
4	303	22	3	4



We can find that in the group of people who think family is not important have higher portion of people who are happy with their current life, which may suggest that there may be some relationship.

Factor Analysis

Even after the data clening process, the number of variables decreased from 440 to 215, there are still too many variables, so further more, we choose to use Factor Analysis to reduce the number of variables.

Covariance Matrix

Before we do factor analysis we have to calculate the covariance matrix:

```
cor = cor(da[,2:215])
```

```
## Warning in cor(da[, 2:215]): the standard deviation is zero
```

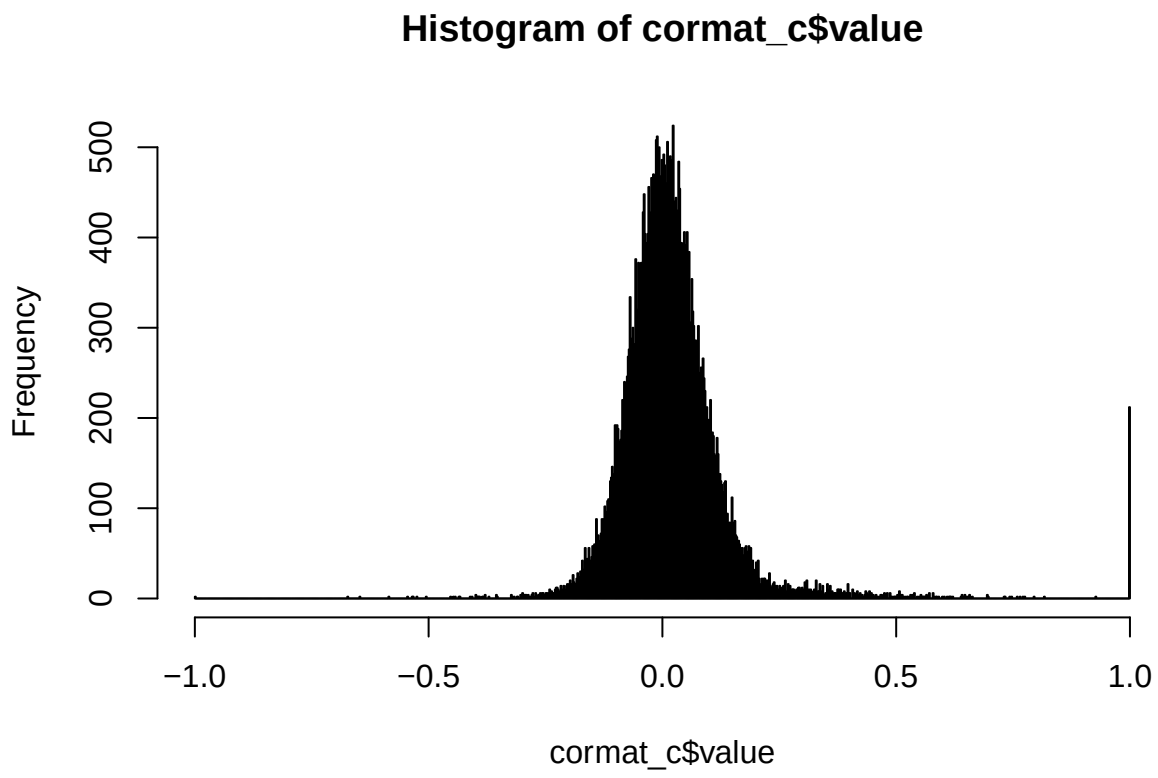
When Calculating covariance matrix, I got the warning message that some of the standard diviation is 0, which means some variables have only one unique value, so we have to delete those variables.

```
# calculate the unique value of each variables.
ui = data.frame(da %>% apply(MARGIN = 2,function(x){unique(x)%>%length}))
# Delete Variables with only one unique variables
da %<>% select(-V125_16,-V125_17)
```

Besides the unique values problem, after reading the article of Jon Starkweather (Factor Analysis with Binary items: A quick review with examples), due to the fact that my data has some binary variables, we have to use special function to calculate the covariance matrix to guarantee R will calculate the right covariance matrix.

Another problem I encounted is that the data set is too big to be processed by my laptop when calculating the covariance matrix, so I have to draw samples from the whole dataset.

```
# Draw Samples from the data
sa = sample(18445,replace = F,size = 1000)
das = da[sa,]
dasa = da[sa,]
# Transform binary variables to factors
ui = data.frame(da %>% apply(MARGIN = 2,function(x){unique(x)%>%length}))
ui["name"] = rownames(ui)
colnames(ui) = c("Uni_value","name")
cat = ui$name[which(ui$Uni_value==2)]
for(i in cat){
  das[i] = as.factor(as.matrix(das[i]))
}
# Calculate Correlation matrix
library(polycor)
cormat = hetcor(das[,2:213])$cor
# Plot the distribution of the covariance between variables
cormat_c = reshape2::melt(cormat)
hist(cormat_c$value,breaks = 1000)
```

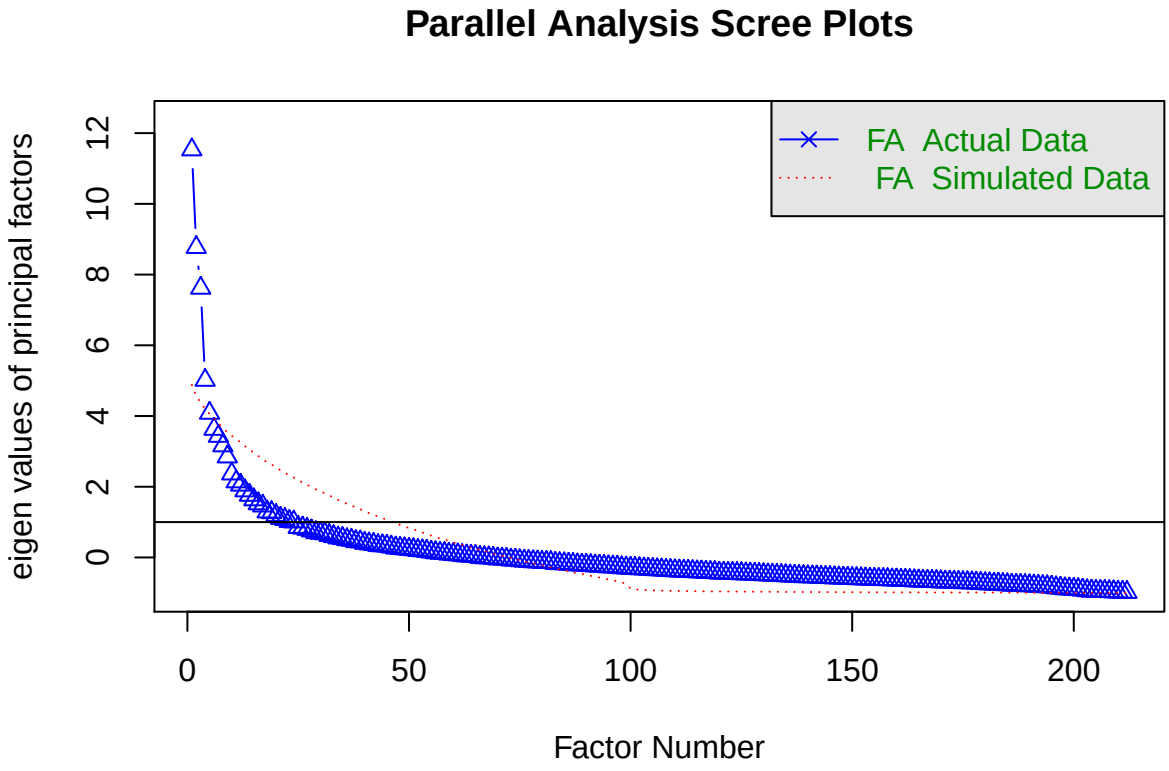


Above is the Histgram of the correlation coefficients, we canconclude that most of the variables have no correlation with other variables.

Factor Analysis

As the first step of factor analysis, we have to decide the number of factors, we use parallel analysis to determine to factors.

```
# Parallele Analysis
set.seed(2019)
parallel = fa.parallel(cormat,fm = "pa",fa = "fa")
```




```
## Parallel analysis suggests that the number of factors =  5  and the number of components =  NA
```

Based on the Parallel Analysis, we can roughly determine the number of factor to be 6.

```
# Do Factor Analysis
pa.fa = fa(r = cormat,nfactors = 6,rotate = "varimax", fm = "pa")
# Extract loading matrix
lo = matrix(nrow = 212,ncol = 6)
va = as.matrix(da[,2:213])
vaa = as.matrix(dasa[,2:213])
for(i in 1 : 212){
  for(j in 1:6){
    lo[i,j] = pa.fa$loadings[i,j]
  }
}
# Calculate Factors
dasa = data.frame(cbind(das[,1],vaa %*% lo))
dafa = data.frame(cbind(da[,1],va %*% lo))
colnames(dafa) = c("Country Code",paste("Factor",seq(1,6,1)))
```

```
colnames(dasa) = c("Country Code",paste("Factor",seq(1,6,1)))
```

Part of the result of factor analysis is shown below:

Table 2: Part of result of Factor Analysis

	Country Code	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6
2231	51	-266.00	215.59	278.10	-48.94	-358.30	-1388.59
2232	51	-263.85	221.83	285.95	-49.52	-364.76	-1383.33
2233	51	-274.51	222.68	292.34	-52.07	-371.18	-1441.32
2234	51	-262.36	235.34	283.47	-44.14	-367.87	-1438.87
2240	51	-257.96	209.67	273.33	-49.38	-351.18	-1339.30
2243	51	-261.51	228.52	284.67	-49.02	-363.51	-1405.58

Cluster Analysis

After using the factor analysis, the number of variables has been reduced to 6, so now we can do more anaysis based on the result of Factor Analysis, here I choose to do cluster analysis using K-means Algorithm.

Same as factor analysis, the first step of cluster analysis is to determine the number of clusters, we can determine through observing the change of SSE, if k is smaller than the real number of clusters, as k increase, the SSE will decrease significantly, but once k exceed the real number of clusters, the decrease of SSE will be smaller, so the plot of SSE will contains a obvious turning point.

$$SSE = \sum_{i=1}^{k} \sum_{p_j \in C_i} \|p_j - m_i\|_2^2$$

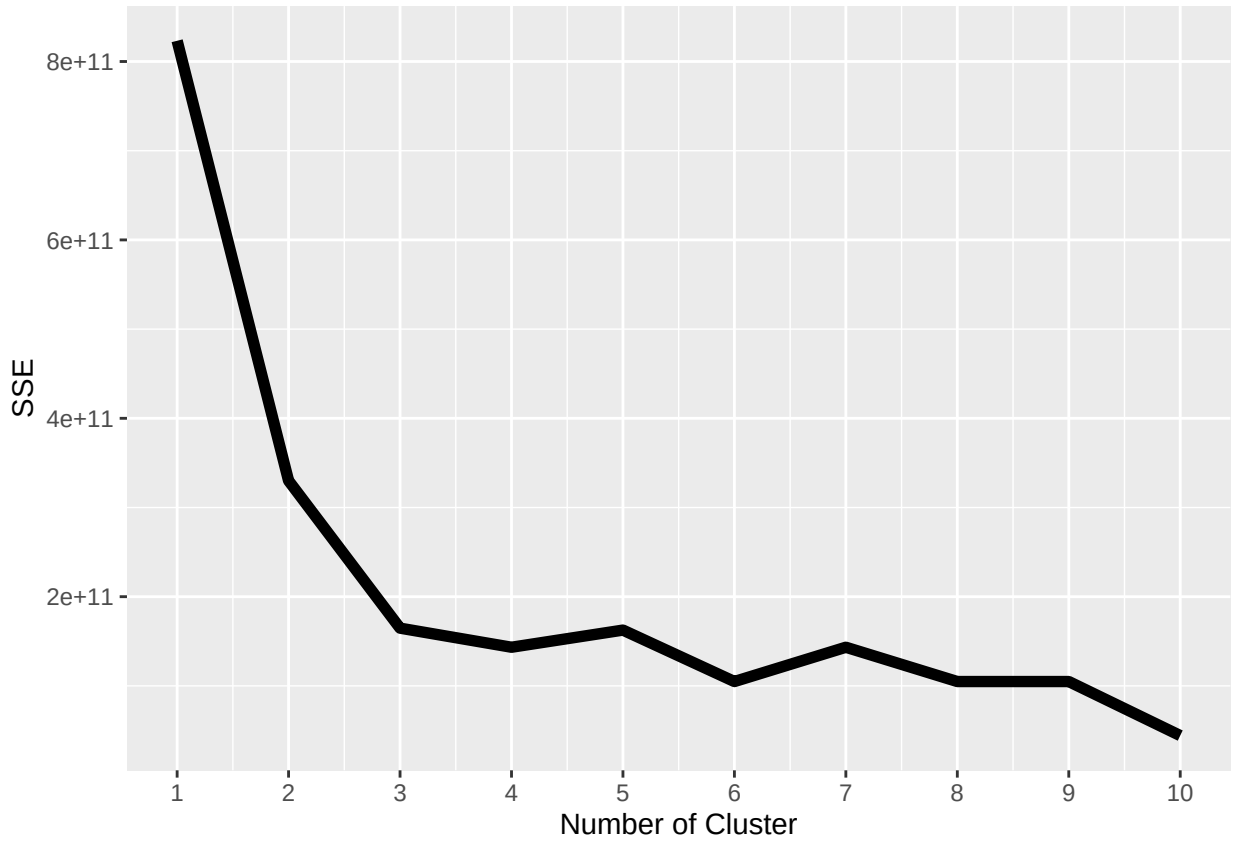
- C_i is the i_{th} cluster
- m_i is the center of the i_{th} cluster
- $\|p_j - m_i\|_2^2$ is the Euclidian Distance between C_i and m_i

Another important fact about k-means is that this algorithm will randomly decide k initial point as the center of each cluster, so the final result of k-means may varies, so for each k, we do several times of clustering, and take the average of SSE of experiments with same k value as the real SSE for coresponding k value.

Here we also draw sample from original data set.

```
library(cluster)
m=10 # max number of clusters
n=10  # number of repeats experiments for each k value
dasc = dasa[-1]
tot = matrix(nrow = m,ncol  = n)
f = matrix(nrow = m,ncol  = n)
# Calculate the total withingroup errors
for(i in 1:m){
  for(j in 1:n){
    cl = kmeans(x = dasc,centers = i)
    f[i,j] = cl$betweenss/cl$tot.withinss
    tot[i,j] = cl$tot.withinss
  }
```

```
}
# calculate average total within-group error
tot %<>% apply(MARGIN = 1,mean)
f %<>% apply(MARGIN = 1,mean)
# plot errors
tot = data.frame(cbind(tot,seq(1,10,1)))
ggplot(tot) + geom_line(aes(y=tot,x = V2),size = 2) +
  scale_x_continuous("Number of Cluster",breaks = seq(1,10,1)) +
  ylab("SSE")
```



According to above plot, the real k value should be 4. So now we set the k value as 4 and do several times of clustering and record the one with smallest SSE.

```
best = NULL
toot = 2.146540e+20
set.seed(2019)

# pick the best clustering
for(i in 1:1000){
  cl = kmeans(dafa,4)
  if (cl$tot.withinss<toot) {
    toot = cl$tot.withinss
    best = cl$cluster
  }
}
da["cluster"]=best
```

Cluster Result Analysis

We would like to know the cluster in each countries.

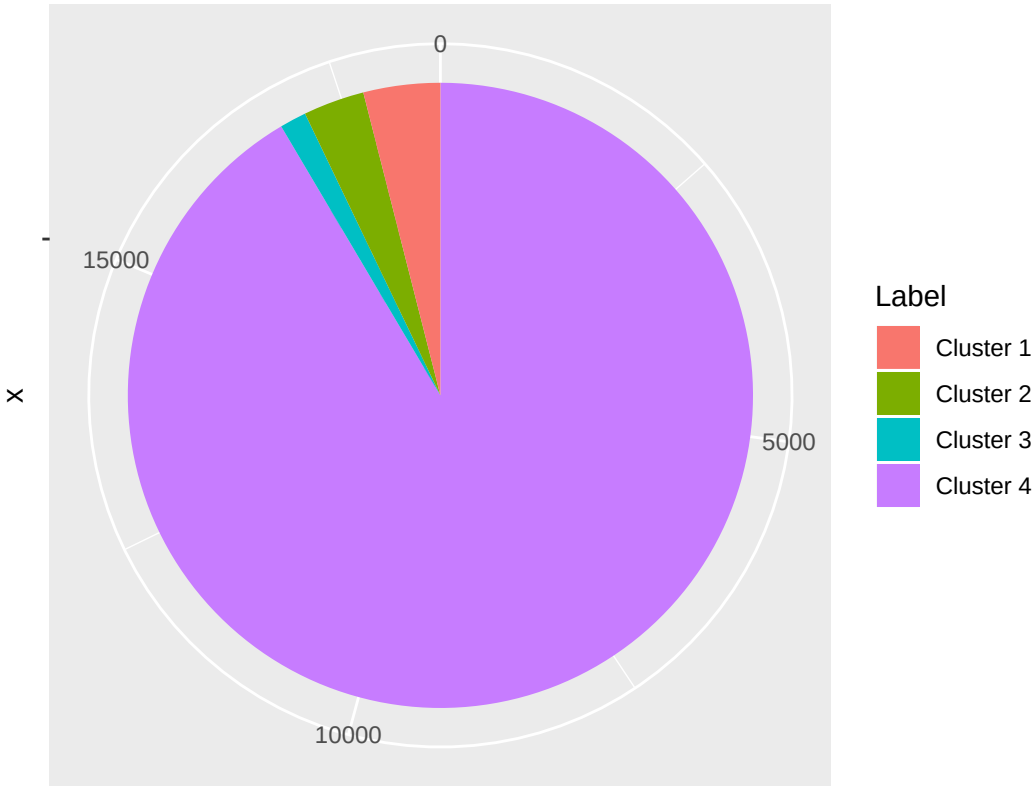
```
# Generate table showing the clustering for each countries
cl_for_each_country = da %>% group_by(V2) %>%
  summarise(num_of_people = length(cluster),num_of_c1 = sum(cluster==1),
            num_of_c2 = sum(cluster==2),num_of_c3 = sum(cluster==3),num_of_c4 = sum(cluster==4))%>%
  arrange(desc(num_of_c4),desc(num_of_c2),desc(num_of_c1),desc(num_of_c3))
colnames(cl_for_each_country) = c("Country Code","Number of People",paste("Cluster",seq(1,4,1)))
```

Table 3: Result of Clustering Analysis

Country Code	Number of People	Cluster 1	Cluster 2	Cluster 3	Cluster 4
484	1402	0	0	0	1402
566	1269	0	147	0	1122
840	1110	0	0	0	1110
458	1062	0	0	0	1062
710	1144	0	0	227	917
716	800	0	0	0	800
288	1249	483	0	0	766
398	753	0	0	0	753
608	919	0	292	0	627

We can find that most of the observations are allocated to cluster 4, which is

```
# Calculate the proportion of each groups
pro_for_each_country = cl_for_each_country %>% apply(MARGIN = 2,sum) %>% data.frame
pro_for_each_country["Label"] = rownames(pro_for_each_country)
colnames(pro_for_each_country) = c("Value","Label")
```


0
5000
10000
15000
x
Label
Cluster 1
Cluster 2
Cluster 3
Cluster 4