ANLY 512: Statistical Learning
Lujia Deng, Jon Kang, Xueyan Liu, Yunfei Zhang

Predicting Unemployment Rate

I. Introduction

The unemployment rate, defined as the percentage of unemployed workers in the total labor force, is often seen as a reflection of both the economic performance of a country and the well-being of its people.

When unemployment becomes an issue, it doesn't simply have an effect on these human beings who are out of work, but the level and persistence of unemployment will have wide-ranging effects across the broader economy. When workers are unemployed, the country as a whole loses their contribution to the economy in terms of the goods or services that could have been produced. Both the unemployed workers and their families lose or have a lower purchasing power, which can lead to unemployment for other workers, creating a cascading effect that ripples through the economy.

The causation of unemployment can be different for each country and is often the complex interactions among many economic factors. Macroeconomic variables are identified to have strong relationships with unemployment, such as gross domestic product (GDP) and interest rate. A large GDP indicates a high level of national production, which usually demands more labor force. A high interest rate, on the other hand, may have a negative impact on the employment because the increased cost of capital can deter business activities. There is also a debate on the influence of globalization and international trade on unemployment. While specialization of countries in global production, resulting from their different competitive advantages, may cause unemployment. Other studies argue that the relationship needs to be evaluated in a broader context with other factors, such as advances in technology and government intervention should be taken into account (Landmann, 2000; Ohlin, 1935; Slaughter, 1997; Spence, Katz, & Lawrence, 2011). Other impactful factors studied include education, social benefits, and rural development (Nickell, 1979; Esser, et al., 2013).

We are interested in mining the underlying relationships between unemployment and relevant economic factors. Specifically, we want to understand the impact of a nation’s macroeconomics (e.g. GDP, foreign direct investment, inflation, consumption expenditure, etc.), population characteristics, economic development (e.g. education, healthcare, social benefits, infrastructure, etc.) on a nation’s unemployment rate. We aim to find which variables are more correlated with unemployment and how they perform in predicting unemployment rates.

II. Data and Methodology

1. Overview of Datasets

The datasets we use here are the World Development Indicator (WDI), published by the World Bank, and Economic Freedom Index, published by the heritage foundation and Wall Street Journal. Both are country-based and we focus on the same time period from 2014 to 2018.

The WDI is a premier compilation of cross-country comparable data and is provided by the World Bank Group, which is a platform offering sources of funding and knowledge for developing countries. It has the most current and accurate global development data available based on nine fields, including agriculture, education, finance, and infrastructure, etc. The unemployment rate (% of the total labor force) is our target variable. Based on preliminary research[1] on unemployment, we first selected 44 from 1000+ indicators that represent the following categories: GDP, population, finance, education, society benefit, health, environment, and transportation. Then we filtered out indicators with over 50% missing values, leaving 29 attributes from the World Development Indicators Database.

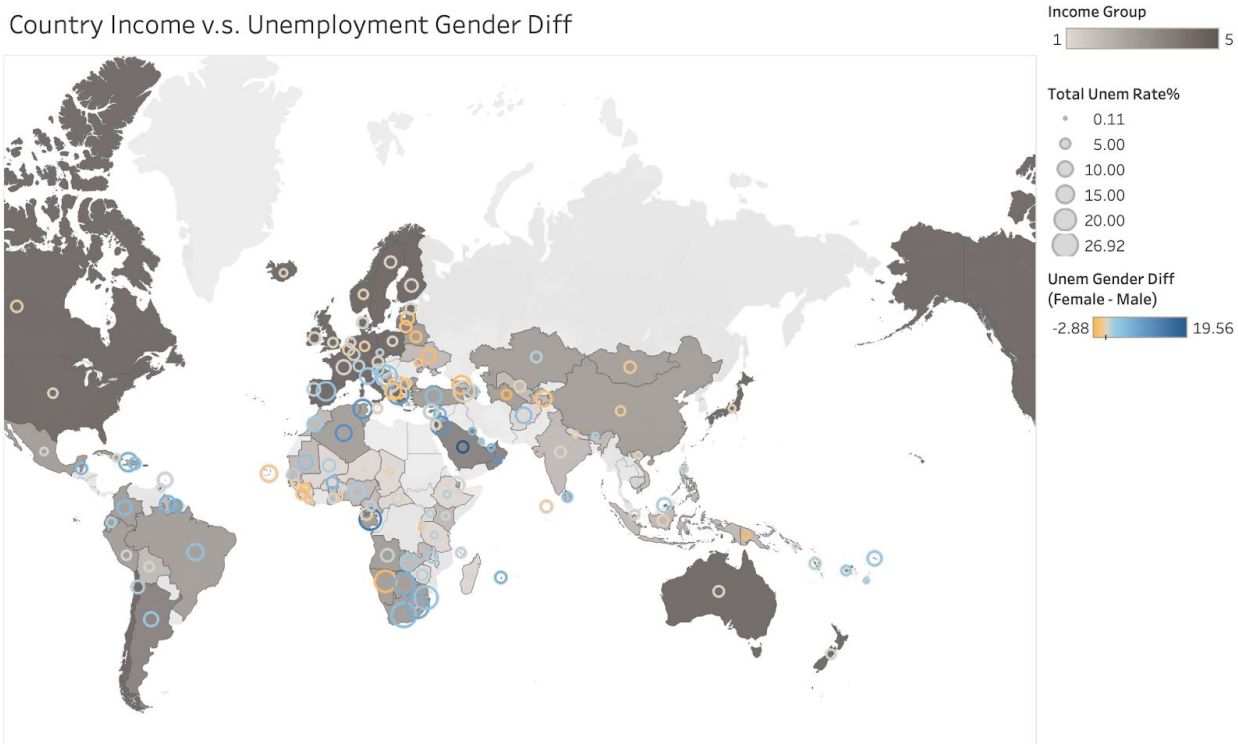
The classical economic theory suggests that economic freedom has a significant influence on a country's economic performance. Economic freedom measures whether individuals are free to work, produce, consume, and invest and whether governments allow labor, capital, and goods to move freely. Therefore, we also include the Economic Freedom Index dataset in our analysis. There are 19 economic freedom variables with a scale from 0 to 100, in the following four fields - the rule of law, government size, regulatory efficiency, and open markets. Ten features are selected, including property rights, government integrity, tax burden, government spending, business freedom, labor freedom, monetary freedom, trade freedom, investment freedom, and financial freedom.

After merging the two datasets by country and year, we have a dataset that contains the unemployment rate from 2014 to 2018 of 165 countries and 38 factors(10.9% missing values).

[1] https://www.thebalance.com/causes-of-unemployment-7-main-reasons-3305596 and https://en.wikipedia.org/wiki/Causes_of_unemployment_in_the_United_States

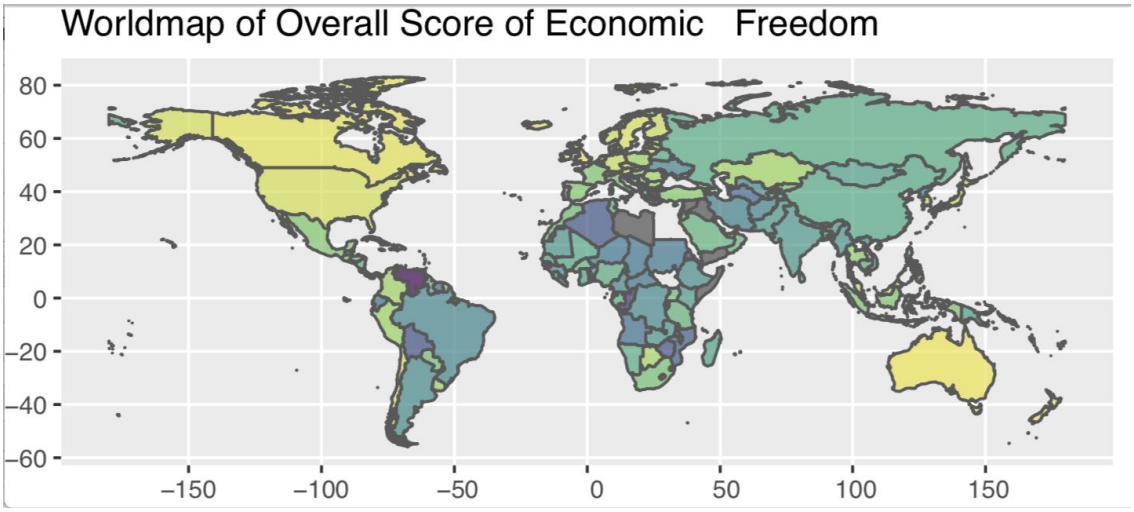
2. Exploratory Data Analysis

To get an overall understanding of the unemployment and economic freedom condition globally, we plotted the following two maps. (Figure 1 and Figure 2)



(Figure 1)

As can be seen in this chart, darker shades indicate higher income; larger bubbles represent higher total unemployment rate while blue indicates greater gender difference in unemployment rate. For most of the countries, a higher income is usually associated with a lower unemployment rate and a smaller gender gap. However, in countries such as Saudi Arabia, the gender difference in unemployment rate is quite high even though its income is high.



(Figure 2)

The second map shows the overall level of economic freedom for each country. If we compare those two maps together, in some regions, unemployment is negatively correlated with income and economic freedom. For example, the southern areas in Africa have lower levels of income and economic freedom and higher unemployment rates. On the other hand, for some countries in the Eastern and Southern Europe, the relationship among the three variables is reversed. Therefore, we derive the conclusion that unemployment is not solely based on a nation's income or economic freedom. Other factors need to be considered.

3. Data Cleaning and Preprocessing

Missing Value Handling

10.9% of our dataset are missing values. We handled them through two approaches, missing value removal and value imputation.

After removing columns with a high proportion of NA and then all incomplete rows, we got a final dataset with 32 columns and 802 rows. For value imputation, we used a regularized iterative PCA algorithm since it can provide better estimates when variables are correlated[2]. This algorithm estimates the missing values by firstly creating an iteration[3]:

a. Replace the missing value by mean;

b. Apply PCA;

c. Update the missing value by $\hat{\mathbf{X}}_S = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$;

d. Use a predefined number of dimensions S;

[2] http://pbil.univ-lyon1.fr/members/dray/files/articles/dray2015a.pdf
[3] http://pbil.univ-lyon1.fr/members/dray/files/articles/dray2015a.pdf

e. Repeat the iteration until convergence and then using a regularized algorithm avoids the overfitting.

We used the imputePCA function from the R package missMDA to implement this method. In the end, we obtained a dataset with 35 columns and 825 rows.

One-Hot Encoding and Data Scaling

We applied one hot encoding to convert the categorical variables into dummy codes. By doing so, models would be able to better employ categorical variables. For example, random forests can work without encoding, however, there might be bias in terms of feature importance for variables with different numbers of categories[4].

For LASSO and PCA, it is necessary to scale the dataset. Thus, we normalized the numeric predictors as well as the response.

4. Statistical Learning Models

Principal Component Regression (PCR)

Principal Component Regression is built on the Principal Component Analysis (PCA) technique. The purpose of PCA is to reduce the dimensionality of the dataset, while retaining its variation. In mathematics[5]PCA summarizes the p-dimensional data by projecting the original vectors onto q directions, where the principal components are projections spanning the subspace. In order to retain the variation, the projections can be found by maximizing the variance. The first principal component is the projection on the direction which has the largest variance in subspace. The second principal component is along with the direction which maximizes variance orthogonal to the first component.

Principal Component Regression (PCR), is an alternative to multiple linear regression[6] which uses PCA results to perform on predictors as new regressors. Compared to the multiple linear regression, PCR has two main advantages, dimensionality reduction and collinearity removal. Since our dataset has 35 variables (21 variables in the case when we removed all the missing values), we wanted to reduce the redundant information. Another method that we used to reduce variables was LASSO Regression, which we will discuss later. The other advantage of PCR is removing collinearity among variables since all principal components are orthogonal with each other. For selecting the number of principal components, we can refer to the cumulative proportion or apply cross-validation. We chose the principal components that can cumulatively explain over 95% variance of dataset.

---

[4] http://rnowling.github.io/machine/learning/2015/08/10/random-forest-bias.html
[5] https://www.stat.cmu.edu/~cshalizi/350/lectures/10/lecture-10.pdf
[6] https://rpubs.com/esobolewska/pcr-step-by-step

Random Forests

Random forests is a tree-based machine learning algorithm. In decision tree algorithms, branches would divide the feature space and predict the observations that fall into each region as the mean of response value for the training observations in the same region.

Random forests can be seen as a set of decision trees that apply the mean of all the trees as the final predictions to the responses. In random forests, the number of trees needs to be predetermined for the model, as well as the number of randomly sampled predictors considered at each split. Typically, the number of predictors considered at each split is chosen as the square root of the total number of predictors.

In our model, we applied a random forest algorithm to build the model, used RMSE (root-mean-square deviation) as the result metrics, and plotted the feature importance chart to understand the relationships between response and its predictors.

LASSO and Gradient Boosting

LASSO regression is a simple technique to reduce model complexity and prevent overfitting which may result from simple linear regression. LASSO regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This results in sparse models with few coefficients; some coefficients can become zero and eliminated from the model.

We used LASSO regression for feature selection because we wanted to reduce the dimensionality of feature space.

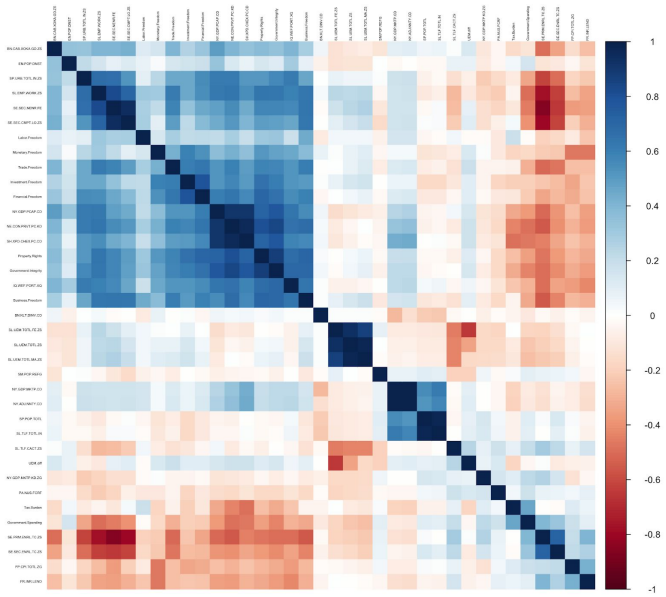
First, we created a variable matrix and a labeled vector. Then we split the data into test and train and conducted k-fold cross-validation for glmnet. Based on the result, we identified the best lambda and listed out the coefficients for each variable.

Gradient boosting produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The algorithm first starts with calculating the average of the target label and the residuals. Next, it constructs a decision tree then predicts the target label using all of the trees within the ensemble. From the prediction it calculates new residuals and repeats the steps until the number of iterations matches the number specified by the hyperparameter. Finally, once trained, it uses all of the trees in the ensemble to make a final prediction as to the value of the target variable.

After LASSO regression, we fit a Gradient Boosting Regression with variables picked by LASSO. The parameters we used are 1) distribution as "gaussian" to show squared error, 2) n.trees as 5000, 3) interaction.depth as 4, 4) shrinkage as 0.1, and 5) verbose as false.

III. Results

1. Correlation Analysis



(Figure 3)

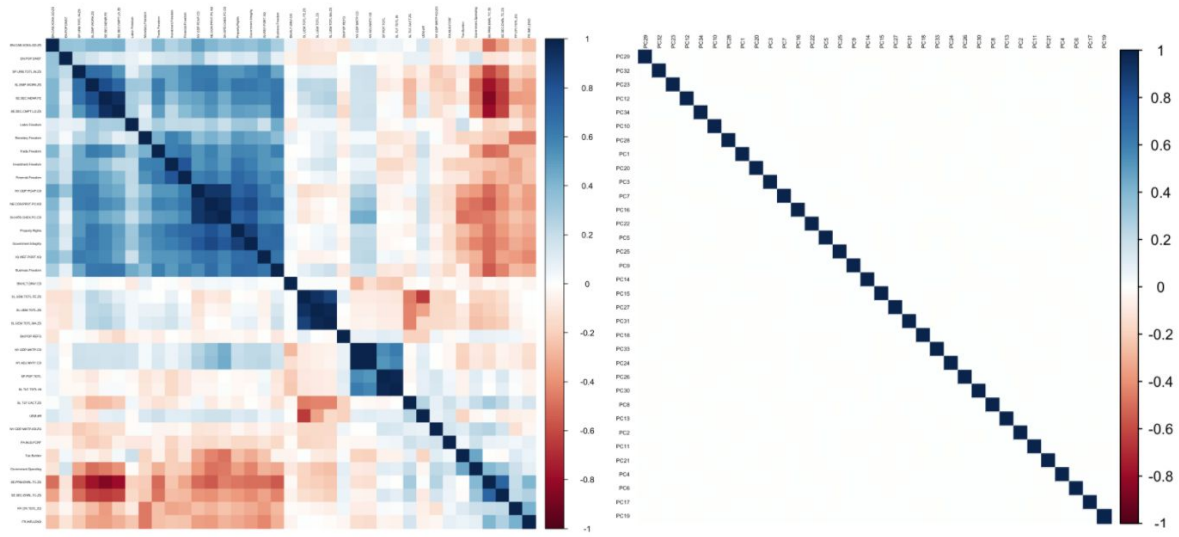
The heatmap illustrates the correlation between variables. Blue reflects the positive correlation while Red reflects negative correlations. Darker colors indicate the stronger correlations. According to the heatmap, our variables are not independent with each other. There are some correlation coefficients of variables near 1, such as unemployment rate and labor force, household final consumption expenditure per capita and GDP, and household final consumption expenditure per capita and government integrity. These highly correlated variables may lead to the multicollinearity problem. Thus, we used the PCR algorithm, which can remove the correlation, and decision tree algorithms, which are immune to multicollinearity by nature. Figure 4 shows the top 9 variables correlated to the response.

(Figure 4)

2. Predictive Models

Principal Component Regression

PCR models began with PCA. After implementing PCA, we first checked the heatmap of the correlation matrix. The left plot in Figure 5 shows the correction before we run the PCA and the right plot in Figure 5 shows the result after PCA; PCA solved the multicollinearity issue successfully.



(Figure 5)

Based on the results of fitted PCA (Table 6), PC1 explains 34.8% of the variability in the data and mainly contains information from the economic freedom index (property rights, government integrity, financial freedom, and business freedom have the largest magnitudes).

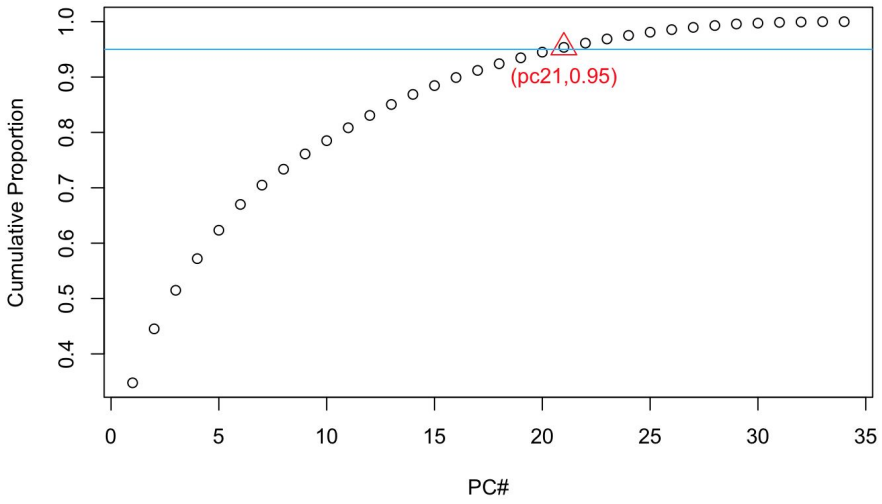
```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5
Standard deviation     3.3990 1.79943 1.52089 1.37817 1.30533
Proportion of Variance 0.3478 0.09747 0.06963 0.05717 0.05129
Cumulative Proportion  0.3478 0.44525 0.51488 0.57206 0.62335
```

```
```{r}
sort(pca.fit$rotation[,'PC1'])
```
```

```
     Property.Rights Government.Integrity    Financial.Freedom
          -0.35147697          -0.34399985          -0.31288383
     Business.Freedom       SL.EMP.WORK.ZS   Investment.Freedom
          -0.30635827          -0.30315897          -0.29633925
```

(Table 6)

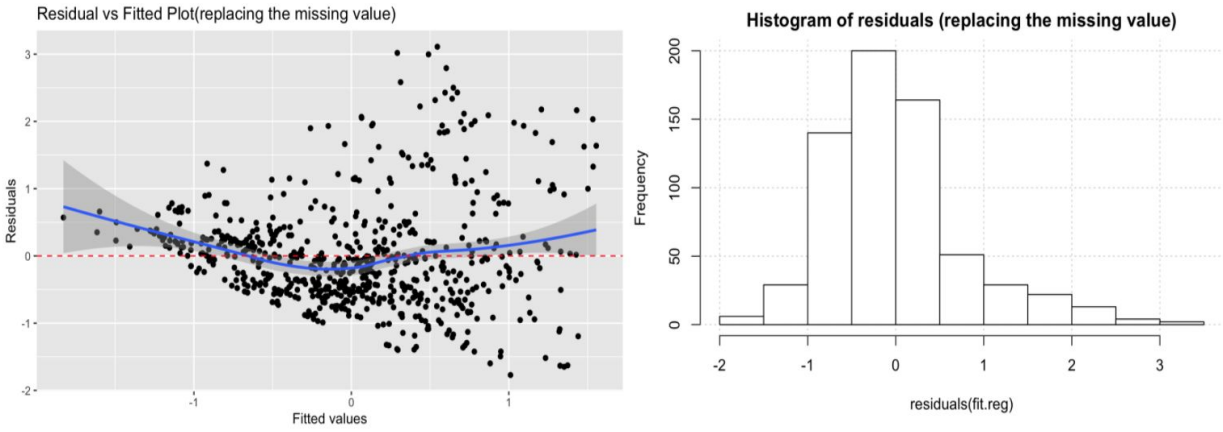
From the cumulative proportion plot (Figure 7) below, we noticed that the first 21 PCs have explained over 95% variability. Therefore, we used these 21 principal components as regressors to fit the regression models.



(Figure 7)

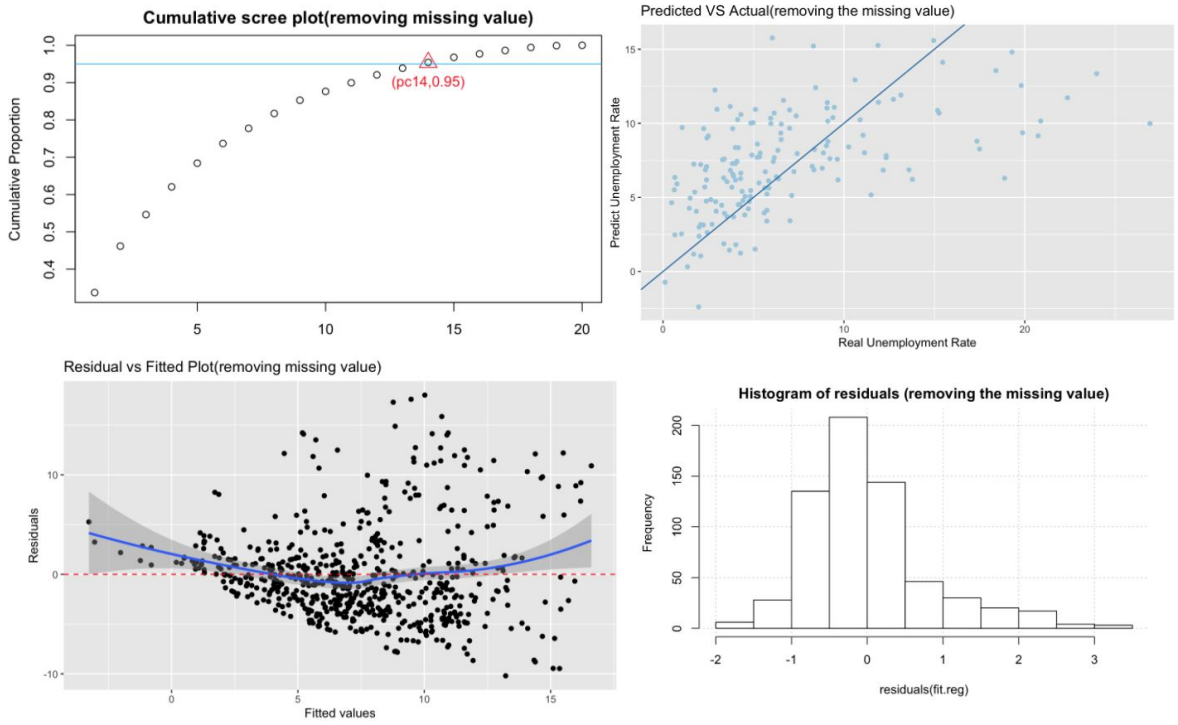
The resulting model has a R-squared score of 0.4, a training RMSE of 0.62 and a test RMSE of 0.56. Figure 8 shows the actual value and predicted value with diagonal lines. Although there is no evidence indicating overfitting, the prediction was not as accurate as we expected. The residuals-vs-fitted-value plot (left plot in Figure 9) shows a curve pattern and the residuals do not follow a normal distribution since they have a right-skewed distribution (right plot in Figure 9).

(Figure 8)



(Figure 9)

We also conducted the same procedure on the dataset generated by removing missing values. We picked the first 14 as regressors (top left plot in figure 10 shows scree plot of PCA) and then implemented the linear regression(top right plot in figure 10 shows the actual value and prediction value with diagonal lines). The PCR result is similar to the model applied to the filled missing value. The residuals-vs-fitted-value plot (bottom left plot in figure 10) also shows a curve pattern and the histogram is right-skewed (bottom right plot in figure 10).



(Figure 10)

Then we compared two results by R-squared and RMSE. The full comparison results are provided in Table 11:

Model with	R squared	RMSE (train)	RMSE (test)
Imputation	0.4019	0.6545855	0.6027348
Deletion	0.3672	0.6186431	0.5620509

(Table 11)

The PCR model with imputed missing values returned a relatively better result. The missing value deletion led to a decrease in the statistical power. Since the simple linear regression algorithm associated with PCR might not be flexible enough, we introduced the more complex model for unemployment rate prediction.

Random Forests

In random forests, we used the datasets preprocessed above without missing values. The predictors include macroeconomic and economics freedom variables mentioned in the previous sections, as well as categorical variables including the region and income group of a country. Some of the features were further manipulated, for example, instead of using the male and female unemployment rate individually, we included their difference as one predictor.

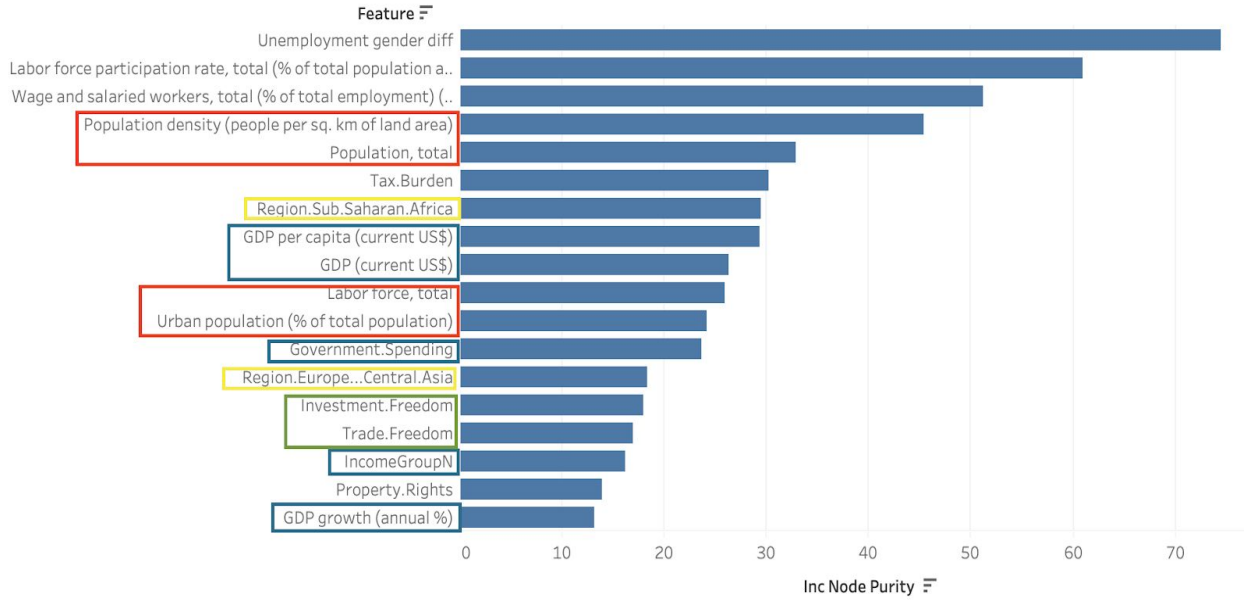
In random forests algorithm, number of trees (ntree) and number of predictors considered in each split (m) are the main parameters. To ensure the best performance of the model, we searched for the best parameters from m in [p/2, p/3, sqrt(p)], where p refers to the total number of predictors, and ntree in [50, 100, 150, 200]. Then, we fit the model with the best combination of parameters and applied 10-fold cross validation to evaluate the accuracy of the model. As a result, we got a 0.71 RMSE for random forests in the scaled dataset, which was acceptable but not worse than the result of PCR. So, we also tried other algorithms in the later part.

By looking at the feature importance result of random forest (Figure 12), we could categorize the important features which contributed the most in the splitting decisions into 5 parts:

1. Labor force and gender equality: the gender difference in unemployment rate, labor force rate, wage and salaried workers rate;

2. Population: total population, population density, urban population;
3. Region;
4. GDP and income: total GDP, GDP per capita, GDP growth rate, government spending, income group;
5. Economic freedom: investment freedom, trade freedom.

The feature important results can shine a light on what features have the most impact on the unemployment issue, which will be discussed further in the conclusion part.



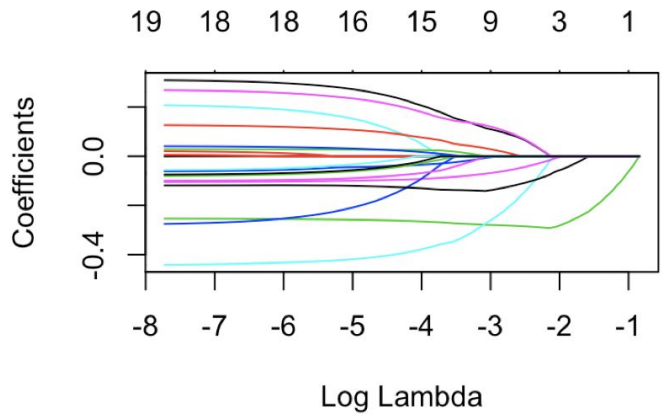
(Figure 12)

LASSO and GBM

The LASSO regression model revealed that the coefficients of SL.TLF.TOTL.IN (Labor force total) and SP.POP.TOTL(Population total) have been shirked to zero as shown in Figure 13. Seventeen variables were left in the model:

SL.TLF.CACT.ZS (Labor force participation rate, total)
NY.GDP.MKTP.CD (GDP)
NY.GDP.PCAP.CD (GDP per capita)
NY.GDP.MKTP.KD.ZG (GDP growth in annual by %)
SL.EMP.WORK.ZS (Wage and salaried workers % from total employment)
EN.POP.DNST (Population density)
SP.URB.TOTL.IN.ZS (Urban population % of total population)
Property Rights
Government Integrity

Tax.Burden
Government Spending
Business Freedom
Labor Freedom
Monetary Freedom
Trade Freedom
Investment Freedom
Financial Freedom

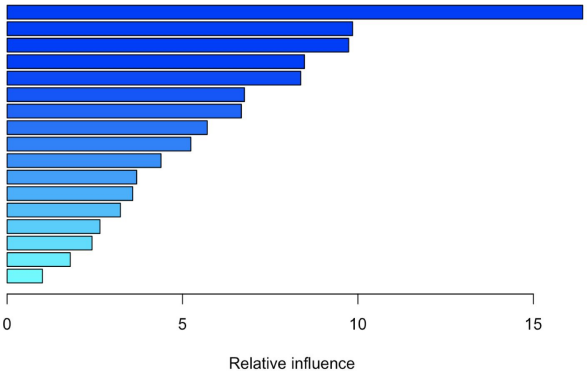



```
20 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)           0.059530522
SP.POP.TOTL           .
SL.TLF.TOTL.IN        .
SL.TLF.CACT.ZS       -0.295374023
NY.GDP.MKTP.CD       -0.019782280
NY.GDP.PCAP.CD       -0.393813235
NY.GDP.MKTP.KD.ZG    -0.168660772
SL.EMP.WORK.ZS        0.165369351
EN.POP.DNST           0.022127884
SP.URB.TOTL.IN.ZS    -0.040684109
Property.Rights      -0.197214408
Government.Integrity  0.185097040
Tax.Burden           -0.044399992
Government.Spending  -0.178550305
Business.Freedom      0.047255334
Labor.Freedom         0.029175794
Monetary.Freedom      0.008359196
Trade.Freedom         0.024739734
Investment.Freedom    0.286800982
Financial.Freedom    -0.156330364
```

(Figure 13)

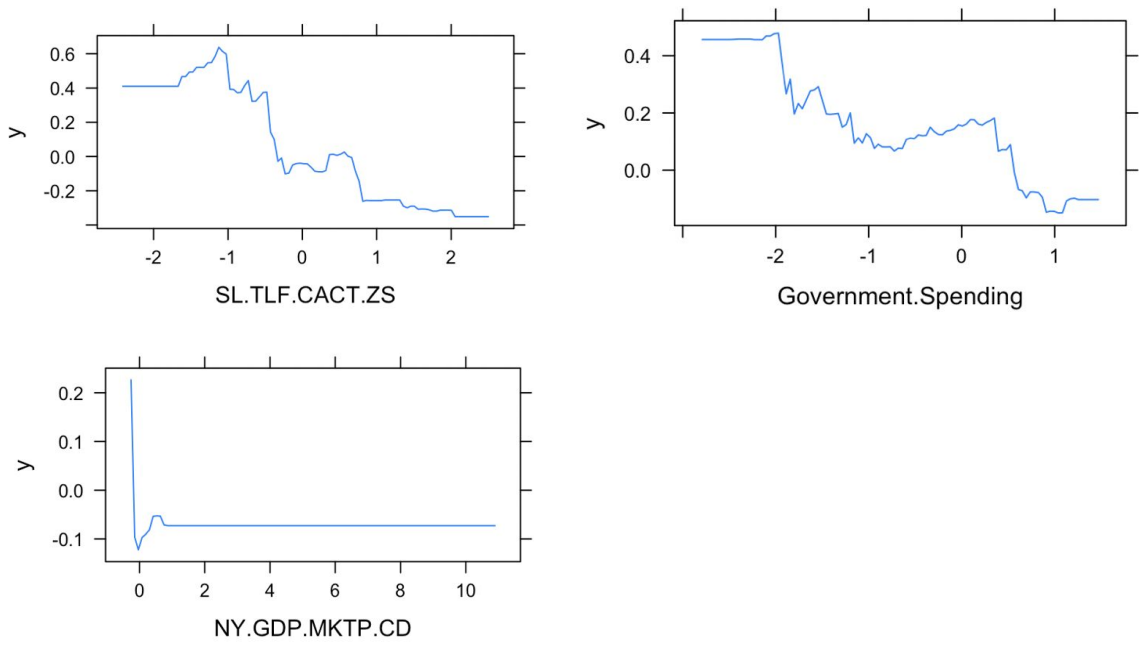
We used the selected variables to proceed with Gradient Boosted Regression Modeling. The Gradient Boosted Regression Model revealed SL.TLF.CACT.ZS (Labor force participation rate) had relative influence as 16.4, followed by GDP and government spending, as shown in Figure 14, which is also aligned with the result of the random forest model.

	var <fctr>	rel.inf <dbl>
SL.TLF.CACT.ZS	SL.TLF.CACT.ZS	16.407329
Government.Spending	Government.Spending	9.849126
NY.GDP.MKTP.CD	NY.GDP.MKTP.CD	9.735892
SL.EMP.WORK.ZS	SL.EMP.WORK.ZS	8.475886
EN.POP.DNST	EN.POP.DNST	8.371207
NY.GDP.MKTP.KD.ZG	NY.GDP.MKTP.KD.ZG	6.765676
SP.URB.TOTL.IN.ZS	SP.URB.TOTL.IN.ZS	6.677123
Trade.Freedom	Trade.Freedom	5.704722
NY.GDP.PCAP.CD	NY.GDP.PCAP.CD	5.239073
Labor.Freedom	Labor.Freedom	4.388042



(Figure 14)

Partial dependence plots of those 3 variables showed a clear relationship between unemployment rate. (Figure 15) First, as SL.TLF.CACT.ZS (Labor force participation rate, total) increased, unemployment rate decreased. Also, as government spending and NY.GDP.MKTP.CD (GDP) increased, unemployment rate decreased and as increased.



(Figure 15)

Finally, We got the RMSE of the model as 0.41.

IV. Conclusion and Discussion

In this study, we have found that 1) factors related to a country's **national production**, **labor and population characteristics,** and **economic freedom** are more correlated with the unemployment rate, 2) other factors, such as economic development attributes in education, social benefits, health, international trade surplus/deficits, interest rate and inflation rate, are not significant.

National production and economic freedom being the most important variables is aligned with our expectation. Economists have now reached the agreement that a country's economic condition is the largest contributor to unemployment. National production of a large and growing scale produces many job opportunities, as what we have recently seen in those growing economies. A higher level of economic freedom encourages people to

participate in more economic activities such as trade and investment, therefore lowering the country's unemployment rate. In addition, we are surprised to see that, keeping other variables constant, the higher the population density is, the lower the unemployment rate is. One explanation is that consumption demand increases as population density rises, therefore inducing more job opportunities.

Compared to GPD, population and economic freedom, variables related to international trades, economic development, as well as interest rate and inflation have less explanatory power in predicting unemployment. To some extent, the results are reasonable. GDP may already contain information of economic development, while the interest rate usually is used as a monetary policy to control and adjust the economic condition, therefore would come after those macroeconomic factors. Although inflation was perceived to have a strong inverse relationship with unemployment in the past, this model was subsequently contradicted (Phelps, 1967), and was proved to be at fault in many empirical cases.

Managing a country's unemployment rate should be at a government's top list, as jobs empower the people, help them provide for their families and themselves, and give them more control and choices in their life. In order to effectively achieve this goal, a government needs to continue to improve the economic conditions through promoting economic freedom and implementing proper policies to encourage economic activities.

We also recognize that relationships among economic factors can be intertwined and complex, and our models may not fully account for issues such as endogeneity, resulting from omitted variables and simultaneity. In the future, we want to 1) conduct more data collection and feature engineering, and 2) include the temporal and lagging effects underlying our data in our model to improve the results.

V. References

Esser, I., Ferrarini, T., Nelson, K., Palme, J., & Sjöberg, O. (2013). Unemployment benefits in EU member states.

Landmann, O. (2000). Wages, unemployment, and globalization: A tale of conventional wisdoms. In Globalization and unemployment (pp. 165-191). Springer, Berlin, Heidelberg.

Nickell, S. (1979). Education and lifetime patterns of unemployment. Journal of Political Economy, 87(5, Part 2), S117-S131.

Ohlin, B. (1935). Interregional and international trade. Harvard University Press, Cambridge.

Slaughter, M. J. (1997). *Does globalization lower wages and export jobs?* (Vol. 11). International Monetary Fund.

Spence, M., KATZ, R., & LAWRENCE, R. Z. (2011). Manufacturing Globalization: The Real Sources of US Inequality and Unemployment. *Foreign Affairs*, 166-171.

Phelps, E. S. (1967). Phillips curves, expectations of inflation and optimal unemployment over time. Economica, 254-281.

VI. Appendix

Unemployment_Rate

yunfeiz

4/30/2020

Part 1 - Data collection and cleaning

```
# Source:
# Worldbank 2014-2019
# Other years' data:  https://data.worldbank.org/indicator/
# Target variable: Unemployment rate  in country level
# Series code : (7 types Unemployment rate value)
#
# SL.UEM.TOTL.ZS         Unemployment, total (% of total labor force) (international estimate)(modeled
# SL.UEM.TOTL.FE.ZS:  female
# SL.UEM.TOTL.MA.ZS:     male
#
#
# SL.UEM.BASC.ZS         Unemployment with basic education  (% of total labor force with basic educatio
# SL.UEM.INTM.ZS         Unemployment with intermediate education (% of total labor force with intermed
# SL.UEM.ADVN.ZS           Unemployment with basic education (% of total labor force with basic educati
#
#
# Other variables:  [list Series code here]
# https://www.kaggle.com/worldbank/world-development-indicators
#
# ### population
# SP.POP.TOTL           population
# SM.POP.NETM             Net migration
#
# SL.TLF.TOTL.IN          Labor force, total number
# SL.TLF.CACT.ZS        Labor force rate
# SL.TLF.BASC.ZS          Labor force with basic education (% of total working-age population with basi
# SL.TLF.INTM.ZS        Labor force with intermediate education (% of total working-age population with
# SL.TLF.ADVN.ZS        Labor force with advanced education (% of total working-age population with adv
#
# SM.POP.REFG           Refugee population by country or territory of asylum
#
# ### GDP
# NY.GDP.MKTP.CD          GDP (current US$)
# NY.GDP.PCAP.CD          GDP per capita (current US$)
# NY.GDP.MKTP.KD.ZG     GDP growth (annual %)
#
#
#
# ### Economic
# BN.CAB.XOKA.GD.ZS :   Current account balance is the sum of net exports of goods and services, net pr
# NY.ADJ.NNTY.CD          Adjusted net national income (current US$)
# BN.KLT.DINV.CD          Foreign direct investment, net (BoP, current US$)
# NE.CON.PRVT.PC.KD     Households and NPISHs Final consumption expenditure per capita (constant 2010 U
# FP.CPI.TOTL.ZG          Inflation, consumer prices (annual %)
```

```
# PA.NUS.FCRF             Official  exchange rate (LCU per US$, period average)
# FR.INR.LEND             Lending interest rate (%)
#
#
# ### Edu
# SE.XPD.TOTL.GD.ZS     Government expenditure on education, total (% of GDP)
# SE.PRM.ENRL.TC.ZS     Pupil-teacher ratio, primary
# SE.SEC.ENRL.TC.ZS     Pupil-teacher ratio, secondary
# SE.ADT.LITR.ZS          Literacy rate, adult total (% of people ages 15 and above)
# SE.SEC.CMPT.LO.ZS     Lower secondary completion rate, total (% of relevant age group)
#
#
# ### Society benefit:
# Per_allsp.cov_pop_tot        Coverage of social protection and labor programs (% of population)
# Per_si_allsi.cov_pop_tot     Coverage of social insurance programs (% of population)
# per_lm_alllm.cov_pop_tot     Coverage of unemployment benefits and ALMP (% of population)
# SL.EMP.WORK.ZS                 Wage and salaried workers, total (% of total employed)
#
#
#
# ### Enviroment
# EN.POP.DNST                    Population density (people per sq. km of land area)
# SP.URB.TOTL.IN.ZS            Urban population (% of total)
# EG.USE.PCAP.KG.OE            Energy use (kg of oil equivalent per capita
#
# ### Health
# SH.XPD.CHEX.PC.CD            Current health expenditure per capita (current US$)
#
# ### Poverty
# GB.XPD.RSDV.GD.ZS            Research and development expenditure (% of GDP)
# SI.POV.GINI                    GINI index (World Bank estimate)
# SI.POV.NAGP                    Poverty gap at national poverty lines (%)
# SI.POV.NAHC                    Poverty headcount ratio at national poverty lines (% of population)
#
# ### Infrastructure
# IQ.WEF.PORT.XQ               Quality of port infrastructure  WEF (1=extremely underdeveloped to 7=wel
#
#
#
# Features variables:
# Economic freedom index
# https://www.kaggle.com/gsutters/economic-freedom
# https://www.kaggle.com/lewisduncan93/the-economic-freedom-index (2019)
# Happiness
# https://www.kaggle.com/PromptCloudHQ/world-happiness-report-2019
# Other variables:  [list Series code here]
# https://www.kaggle.com/worldbank/world-development-indicators
```

1.1 Dataset#1: World Bank data

load data from world bank

```
# Load the workbank data
df = read.csv('Original_Unem_Data.csv')
colnames(df)
```

```
## [1] "Country.Name"   "Country.Code"   "Series.Name"    "Series.Code"
## [5] "X2014..YR2014." "X2015..YR2015." "X2016..YR2016." "X2017..YR2017."
## [9] "X2018..YR2018."
```

```
dim(df)
```

```
## [1] 9336    9
```

look at the format

reformat

```
# create the df for reformat data

# create year col
year = sort(rep(c(2014,2015,2016,2017,2018),length(unique(df$Country.Code))))

# create empyt df
df2 = as.data.frame(matrix(,length(year),length(unique(df$Series.Code))))

# get country name and country code
country_name_list = as.character(unique(unlist(df$Country.Name)))
country_code_list = as.character(unique(unlist(df$Country.Code)))
country = rep(country_name_list,5)
country.code = rep(country_code_list,5)

# add country name and country code to empty df
df2 = cbind(country.name = country,country.code= country.code , year = year, df2)
# change data type
colnames(df2)[4:46]  =  as.character(unique(unlist(df$Series.Code)))
```

```
# fill the data into empty df
options(digits=10)
#df2[df2['country.code']=='AFG' & df2['year']=='2014', 'SL.UEM.TOTL.ZS']  = as.numeric(as.character(df[
#as.numeric(as.character(df[df['Country.Code']== ctry & df['Series.Code']==code,yr]))
#as.numeric(as.character(df[df['Country.Code']=="ZWE" & df['Series.Code']=="IQ.WEF.PORT.XQ",'2018']))

i = 0
# fill data based on year
for (yr in as.character(unique(df2$year))){
  # fill data based on country names
  for( ctry in as.character(unique(df2$country.code))){
    # fill data based on features code names
    for ( code in  as.character(unique(unlist(df$Series.Code)))  ){

      df1_loc = as.numeric(as.character(df[df['Country.Code']==ctry & df['Series.Code']==code,yr]))
```

```
      df2[df2['country.code']==ctry & df2['year']==yr, code] = df1_loc

    }
  }
}
#head(df2)
```

Fill missing values

1. filter out the the columns that contains too many missing values
2. remove the countries without unemployment rate

```
# check missing value %
table(is.na.data.frame(df2))[2]/sum(table(is.na.data.frame(df2)))
```

```
##         TRUE
## 0.3996193148
```

```
dim(df2)
```

```
## [1] 1085   46
```

```
#write.csv(df2,"clean_df.csv", row.names = FALSE) # backupp data

#  remove the country without unemployment rate
df3 = df2[!is.na(df2$SL.UEM.TOTL.ZS),]

#df_backup = df3

# remove the col with more than 50% empty value
for (i in colnames(df3)){
  if (mean(is.na(df3[i])) > 0.5 &  mean(is.na(df3[i])) < 1){
  df3[i] = NULL
    }
}

# check missing value %
table(is.na.data.frame(df3))[2]/sum(table(is.na.data.frame(df3)))
```

```
##         TRUE
## 0.1718360071
```

Unified the country name

```
#df4 = df_backup
df4 = df3
df4$country.name <- as.character(df4$country.name)
#str(df4)

df4$country.name[df4$country.name == 'Korea, Dem. People's Rep.'] = 'North Korea'
df4$country.name[df4$country.name == 'Korea, Rep.'] = 'South Korea'
df4$country.name[df4$country.name == 'Iran, Islamic Rep.'] = 'Iran'
df4$country.name[df4$country.name == 'Egypt, Arab Rep.'] = 'Egypt'
df4$country.name[df4$country.name == 'Gambia, The'] = 'Gambia'
```

```
df4$country.name[df4$country.name == 'Russian Federation'] = 'Russia'
#df4[grepl(paste('Micronesia', collapse = "|"), df4[["country.name"]]),]
df4$country.name[df4$country.name == 'Slovak Republic'] = 'Slovakia'
df4$country.name[df4$country.name == 'Syrian Arab Republic'] = 'Syria'
df4$country.name[df4$country.name == 'Bahamas, The'] = 'Bahamas'
df4$country.name[df4$country.name == 'Venezuela, RB'] = 'Venezuela'
df4$country.name[df4$country.name == 'Yemen, Rep.'] = 'Yemen'
#unique(df4$country.name )
#length(which(df=='..'))
```

1.2 Dataset#2: Econ Freedom data

Unified the country name

```
ec = read.csv('Original_Freedom_Data.csv')
ec$Name <- as.character(ec$Name)
# remove space after strings
ec$Name = stringr::str_trim(as.character(ec$Name),'right')

ec$Name[ec$Name == 'Democratic Republic of Congo'] = 'Congo, Dem. Rep.'
ec$Name[ec$Name == 'Republic of Congo'] = 'Congo, Rep.'

ec$Name[ec$Name == 'The Gambia'] = 'Gambia'
ec$Name[ec$Name == 'Hong Kong'] = 'Hong Kong SAR, China'
ec$Name[ec$Name == 'Macau'] = 'Macao SAR, China'
ec$Name[ec$Name == 'Burma'] = 'Myanmar'

ec$Name[ec$Name == 'Saint Lucia'] = 'St. Lucia'
ec$Name[ec$Name == 'Saint Vincent and the Grenadines'] = 'St. Vincent and the Grenadines'

ec$Name[ec$Name == 'São Tomé and Príncipe'] = 'Sao Tome and Principe'

ec$Name[ec$Name == 'Micronesia'] = 'Egypt, Arab Rep.'
ec$Name[ec$Name == 'Egypt'] = 'Egypt, Arab Rep.'
#unique(ec$Name)
```

1.3 Merge 2 datasets

```
ec$Name <- as.factor(ec$Name)
df4$country.name <- as.factor(df4$country.name)
colnames(ec)[1:2] <- c("country.name","year")
# str(ec)
# str(df4)

df_merge = merge(df4, ec, by=c("country.name","year"))
df_merge[df_merge=='N/A']=NA



table(is.na.data.frame(df_merge))
```

```
##
## FALSE  TRUE
## 32214  5411
```

```
# missing value rate
table(is.na.data.frame(df_merge))[2]/sum(table(is.na.data.frame(df_merge)))
```

```
##         TRUE
## 0.1438139535
```

```
#look at the merged dataset
dim(df_merge)
```

```
## [1] 875  43
```

```
#number of countries
dim(df_merge)[1]/5
```

```
## [1] 175
```

```
#write.csv(df.temp,"df.temp .csv", row.names = FALSE)
```

further manipulation to missing values

Remvoe the country that contain more than 70% missing values in both data

```
#cot = 0
#df_merge_backup = df_merge

# remove the col with more than 50% empty value
for (i in colnames(df_merge)){
  if (mean(is.na(df_merge[i])) > 0.5 &  mean(is.na(df_merge[i])) < 1){
    # print(colnames(df_merge[i]))
    # print(mean(is.na(df_merge[i])))
    # cot = cot+1
    # print(cot)
    df_merge[i] = NULL
    }
}

# get  the country list that  contain more than 70% missing values in both data
cot2=0
temp_country = c()
for (i in row.names(df_merge)){
  if (mean(is.na(df_merge[i,])) > 0.3){
    temp_country=c(temp_country,as.character(df_merge[i,]$country.name))
  }
}
unique(temp_country)
```

```
##  [1] "Eritrea"      "Iran"         "Iraq"         "Libya"
##  [5] "North Korea"  "Somalia"      "Syria"        "Turkmenistan"
##  [9] "Venezuela"    "Yemen"
```

```
df_merge2 = df_merge[!df_merge$country.name %in% unique(temp_country),]
```

```r
#look at the merged dataset
dim(df_merge2)
```

```
## [1] 825  41
```

```r
#number of countries
dim(df_merge2)[1]/5
```

```
## [1] 165
```

```r
# compute the missing value rate
table(is.na.data.frame(df_merge2))[2]/sum(table(is.na.data.frame(df_merge2)))
```

```
##         TRUE
## 0.1070805617
```

Remove Overall.Score from the dataset as it's calculated from the other freedom features

```r
df_merge3 = df_merge2
df_merge3$Overall.Score = NULL

#look at the merged dataset
dim(df_merge3)
```

```
## [1] 825  40
```

```r
#number of countries
dim(df_merge3)[1]/5
```

```
## [1] 165
```

```r
# compute the missing value rate
missing_value = table(is.na.data.frame(df_merge3))[2]/sum(table(is.na.data.frame(df_merge3)))

cat("In our final dataset, the missing value rate is",missing_value)
```

```
## In our final dataset, the missing value rate is 0.1095757576
```

```r
## output the data
write.csv(df_merge3,"clean_df_with_ec.csv", row.names = FALSE)
```

1.4 Data preprocessing

One-hot encode

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
df = read.csv("clean_df_with_ec.csv")
#df = df_merge3
# one hot encode 'year'
# convert the data type to character
df$year <- sapply(df$year, as.character)
```

```
# dummy code
dummy <- dummyVars("~ year", data=df)
onehot <- data.frame(predict(dummy, newdata = df))
df<-cbind(df,onehot)
write.csv(df,"onehot_cleaned.csv")
```

Create another dataset without NA for those algorithms can't handle missing value

```
library(tidyverse)
```

```
## -- Attaching packages ------------------- tidyverse 1.3.0 --
```

```
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3
```

```
## -- Conflicts ---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(dplyr)
### combine add country income and region group info to data
#dat <- read.csv("onehot_cleaned.csv")
dat<-df
country <- read.csv("country_extra_info.csv")
groups <- country[c("CountryCode", "Region", "IncomeGroup")]
names(groups)[1] = "country.code"
full <- left_join(dat, groups, by = "country.code")
```

```
## Warning: Column `country.code` joining factors with different levels,
## coercing to character vector
```

```
### Remove missing values

## get names of columns with too many NA
na_counts <- sapply(full, function(x) sum(is.na(x)))
na_50 <- na_counts>50
na_50
```

```
##         country.name                 year         country.code
##                FALSE                FALSE                FALSE
##       SL.UEM.TOTL.ZS    SL.UEM.TOTL.FE.ZS    SL.UEM.TOTL.MA.ZS
##                FALSE                FALSE                FALSE
##          SP.POP.TOTL       SL.TLF.TOTL.IN       SL.TLF.CACT.ZS
##                FALSE                FALSE                FALSE
##          SM.POP.REFG       NY.GDP.MKTP.CD       NY.GDP.PCAP.CD
##                 TRUE                FALSE                FALSE
##    NY.GDP.MKTP.KD.ZG    BN.CAB.XOKA.GD.ZS       NY.ADJ.NNTY.CD
##                FALSE                FALSE                 TRUE
##       BN.KLT.DINV.CD    NE.CON.PRVT.PC.KD       FP.CPI.TOTL.ZG
##                FALSE                 TRUE                 TRUE
##          PA.NUS.FCRF          FR.INR.LEND       SE.SEC.NENR.FE
```

```
##                 TRUE                 TRUE                 TRUE
##    SE.PRM.ENRL.TC.ZS    SE.SEC.ENRL.TC.ZS    SE.SEC.CMPT.LO.ZS
##                 TRUE                 TRUE                 TRUE
##       SL.EMP.WORK.ZS          EN.POP.DNST    SP.URB.TOTL.IN.ZS
##                FALSE                FALSE                FALSE
##    SH.XPD.CHEX.PC.CD          SI.POV.NAGP       IQ.WEF.PORT.XQ
##                 TRUE                 TRUE                 TRUE
##      Property.Rights Government.Integrity           Tax.Burden
##                FALSE                FALSE                FALSE
##  Government.Spending     Business.Freedom        Labor.Freedom
##                FALSE                FALSE                FALSE
##     Monetary.Freedom        Trade.Freedom   Investment.Freedom
##                FALSE                FALSE                FALSE
##    Financial.Freedom             year2014             year2015
##                FALSE                FALSE                FALSE
##             year2016             year2017             year2018
##                FALSE                FALSE                FALSE
##               Region          IncomeGroup
##                FALSE                FALSE
```

```
remove_cols <- c("SM.POP.NETM", "NY.ADJ.NNTY.CD", "NE.CON.PRVT.PC.KD","PA.NUS.FCRF", "FR.INR.LEND", "SE

## drop columns with over 50 missing values - remain 33 columns
nonNA <- full[, !names(full) %in% remove_cols]

## drop rows with missing values - remain 802 rows
nonNA <- drop_na(nonNA)

## write to a new datafile
write.csv(nonNA, "nonNA.csv")
```

One-hot encode the dataset without NA

```
#df_n = read.csv("nonNA.csv")
df_n = nonNA
# dummy code for Region
dummy <- dummyVars("~ Region", data=df_n)
onehot <- data.frame(predict(dummy, newdata = df_n))
df_n<-cbind(df_n,onehot)

## convert income group to numeric
df_n[df_n[,"IncomeGroup"]=='Low income','IncomeGroupN']<-1
df_n[df_n[,"IncomeGroup"]=='Lower middle income','IncomeGroupN']<-2
df_n[df_n[,"IncomeGroup"]=='Upper middle income','IncomeGroupN']<-3
df_n[df_n[,"IncomeGroup"]=='High income: nonOECD','IncomeGroupN']<-4
df_n[df_n[,"IncomeGroup"]=='High income: OECD','IncomeGroupN']<-5
```

1.5 Create series notes for feature description

```
# create series notes
ser = read.csv('Series_Data.csv')
```

```
series_note = ser[ser$Code %in% colnames(df_merge3)[4:32],][c('Code','Indicator.Name','Long.definition'

series_note[1:4] = sapply(series_note[1:4],as.character)
str(series_note)
```

```
## 'data.frame':    27 obs. of  4 variables:
##  $ Code           : chr  "SL.UEM.TOTL.ZS" "SL.UEM.TOTL.FE.ZS" "SL.UEM.TOTL.MA.ZS" "SP.POP.TOTL" ...
##  $ Indicator.Name : chr  "Unemployment, total (% of total labor force) (modeled ILO estimate)" "Unemp
##  $ Long.definition: chr  "Unemployment refers to the share of the labor force that is without work bu
##  $ Topic          : chr  "Social Protection & Labor: Unemployment" "Social Protection & Labor: Unempl
```

```
#https://www.heritage.org/index/book/chapter-2



series_note = rbind(series_note,c("property rights","property rights","Property rights are a primary fa

series_note = rbind(series_note,c("government integrity","government integrity","In a world characteriz

series_note = rbind(series_note,c("tax burden","tax burden","All governments impose fiscal burdens on e

series_note = rbind(series_note,c("government spending","government spending","The cost, size, and intr

series_note = rbind(series_note,c("business freedom","business freedom","An individual's ability to est

series_note = rbind(series_note,c("labor freedom","labor freedom","The ability of individuals to find e

series_note = rbind(series_note,c("monetary freedom","monetary freedom","Monetary freedom requires a st

series_note = rbind(series_note,c("trade freedom","trade freedom","Many governments place restrictions

series_note =rbind(series_note,c("investment freedom","investment freedom"," A free and open investment

series_note =rbind(series_note,c("financial freedom","financial freedom","An accessible and efficiently

write.csv(series_note,"series_note.csv", row.names = FALSE)
```

Part 2 - Exploratory Analysis

2.1 World Map

world map of Econ Freedom dataset

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 3.6.2
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

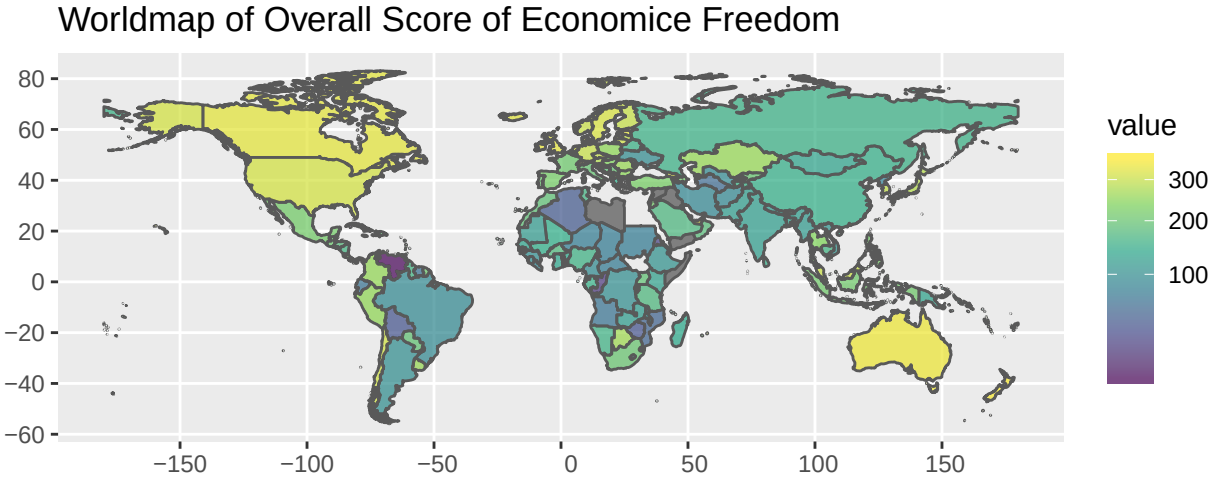
```
library(rnaturalearth)
library(ggspatial)
```

```
## Warning: package 'ggspatial' was built under R version 3.6.2
```

```
library(cowplot)
```

```
##
## ************************************************************
## Note: As of version 1.0.0, cowplot does not change the
##   default ggplot2 theme anymore. To recover the previous
##   behavior, execute:
##   theme_set(theme_cowplot())
## ************************************************************
```

```
library(googleway)
library(ggrepel)
library(ggplot2)
world <- ne_countries(scale = "medium", returnclass = "sf")
#ec2 = read.csv("df_merge.csv")
ec2 = df_merge

names(ec2)[3]= 'code'

map_ec_temp=ec2[ec2$year==2018,c("code","Overall.Score")]

names(map_ec_temp) = c("gu_a3","value")
map_ec_temp = map_ec_temp[!duplicated(map_ec_temp$gu_a3), ]
# merge the data with poly spaital data
df_map2= merge(map_ec_temp,world[c("gu_a3","geometry")])
df_map2$value = as.numeric(df_map2$value)

ggplot(data = df_map2,aes(geometry = geometry)) +
      geom_sf(data = df_map2, aes(fill = value))+
      scale_fill_viridis_c(trans = "sqrt", alpha = .7)+
  labs(title="Worldmap of Overall Score of Economice Freedom")
```

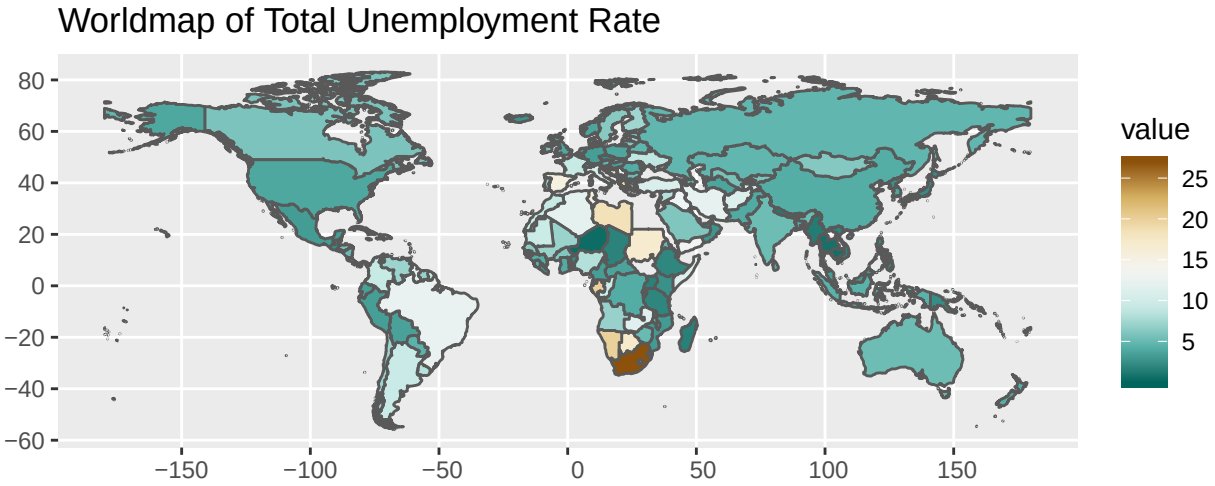


world map of Unemployment Rate dataset

```
library(RColorBrewer)

map_ec_temp=ec2[ec2$year==2018,c("code","SL.UEM.TOTL.ZS")]
```

```
names(map_ec_temp) = c("gu_a3","value")
map_ec_temp = map_ec_temp[!duplicated(map_ec_temp$gu_a3), ]
# merge the data with poly spaital data
df_map2= merge(map_ec_temp,world[c("gu_a3","geometry")])
df_map2$value = as.numeric(df_map2$value)

ggplot(data = df_map2,aes(geometry = geometry)) +
      geom_sf(data = df_map2, aes(fill = value))+
      scale_fill_viridis_c(trans = "sqrt", alpha = .7)+
  labs(title="Worldmap of Total Unemployment Rate")+ scale_fill_distiller(palette = "BrBG")
```

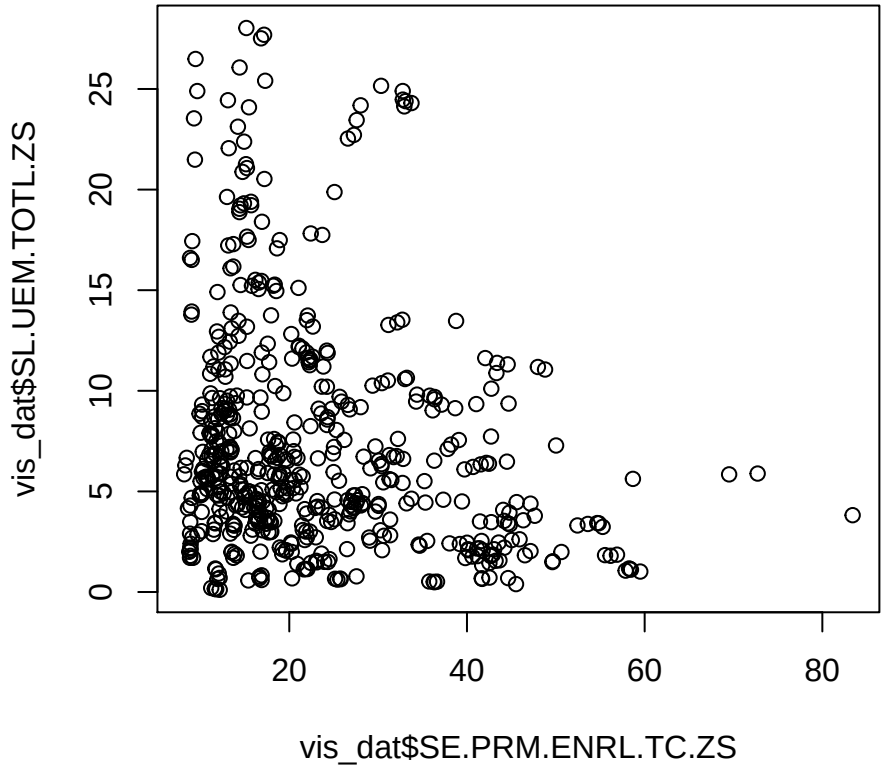
```
## Scale for 'fill' is already present. Adding another scale for 'fill',
## which will replace the existing scale.
```



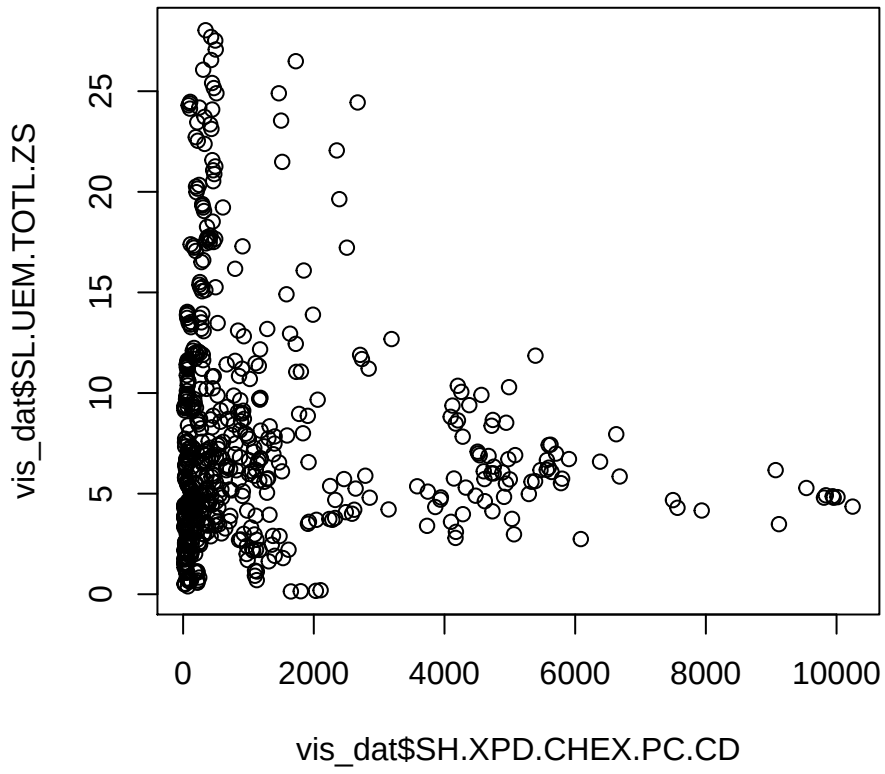
2.2 Scatter plot to show the relationships between variables and response

```
library(ggplot2)
vis_dat <- full[4:41]

## Loop over every quantitative variable and find those which show stronger correlations with the unemp
# SP.POP.TOTL          SM.POP.NETM       SL.TLF.TOTL.IN      SL.TLF.CACT.ZS    SM.POP.REFG   NY.GDP.MKTP.C
#for (i in 1:10){
  #plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat[,i])
#}

# NY.GDP.MKTP.KD.ZG    BN.CAB.XOKA.GD.ZS    NY.ADJ.NNTY.CD      BN.KLT.DINV.CD       NE.CON.PRVT.PC.KD FP
#for (i in 11:20){
  #plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat[,i])
#}

# SE.PRM.ENRL.TC.ZS    SE.SEC.ENRL.TC.ZS    SE.SEC.CMPT.LO.ZS    per_lm_alllm.cov_pop_tot   SL.EMP.WORK.ZS

#for (i in 21:25){
  #plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat[,i])
#}
```

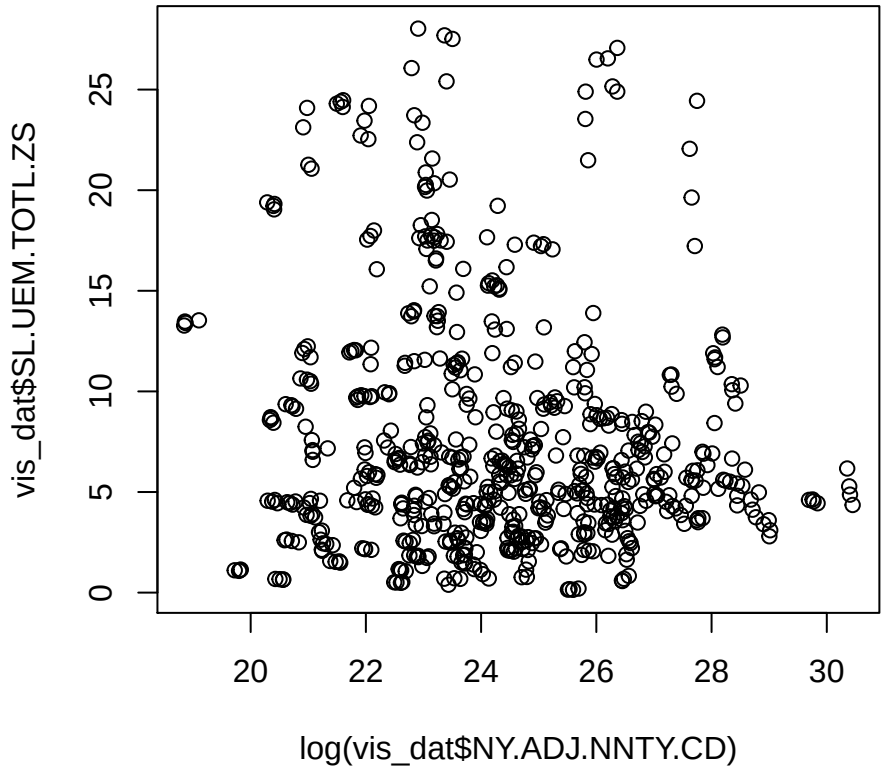
```r
#for (i in 27:37){
  #plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat[,i])
#}

## select proper plots
plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat$SE.PRM.ENRL.TC.ZS)
```



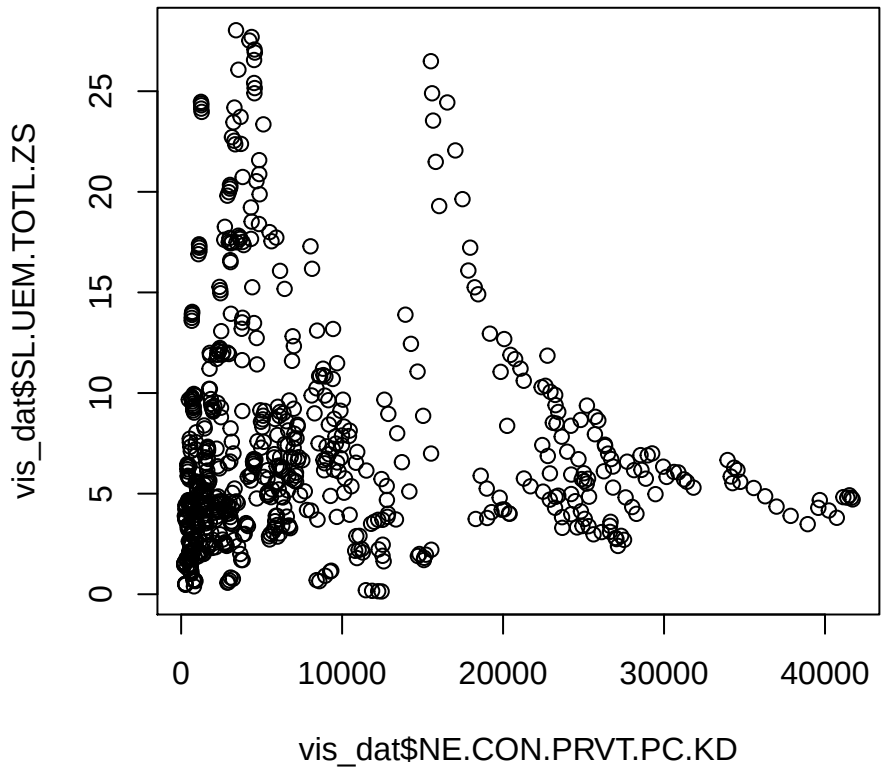

```r
plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat$SH.XPD.CHEX.PC.CD)
```




```
plot(vis_dat$SL.UEM.TOTL.ZS~log(vis_dat$NY.ADJ.NNTY.CD))
```




```
plot(vis_dat$SL.UEM.TOTL.ZS~vis_dat$NE.CON.PRVT.PC.KD)
```



2.3 Clustering Analysis

```
library(tidyverse)
library(dplyr)
library(deldir)
```

```
## deldir 0.1-25
```

```
library(cluster)
```

Data preparation

```
cluster_data <- nonNA

## normalize quantitative cols
x <- data.matrix(cluster_data[, c(4:25)])
x_norm <- scale(x)
```

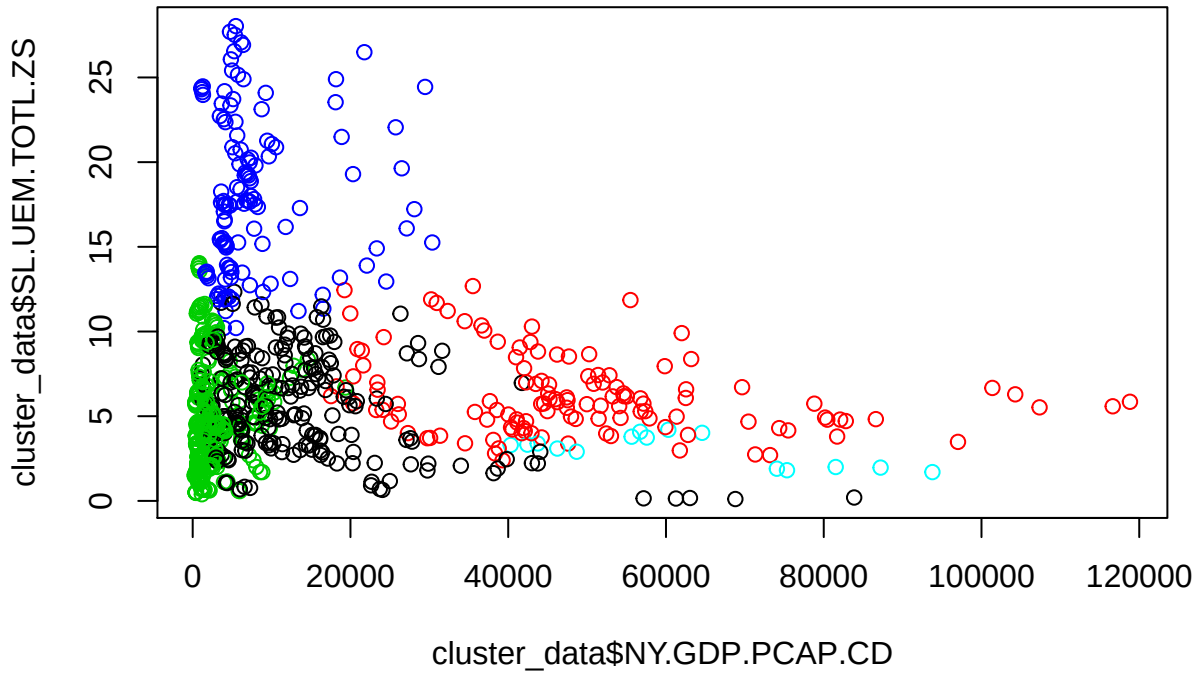
K-MEANS

```
# K=5
km.out_5=kmeans(x_norm,5)
km_group <- km.out_5$cluster

cluster_data$cluster_group <- km_group
```

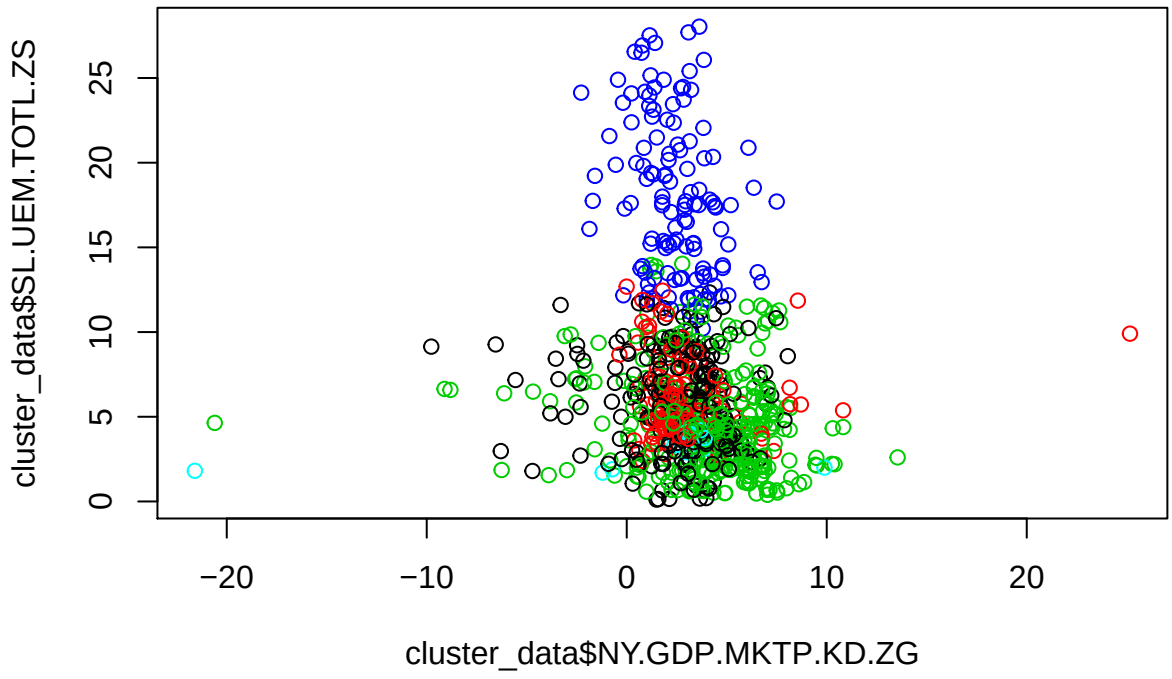
```r
table(cluster_data$cluster_group,cluster_data$IncomeGroup)
```

```
##
##         High income: nonOECD High income: OECD Low income
##   1   0                   66                27          0
##   2   0                    5               121          0
##   3   0                   10                 0        127
##   4   0                    9                12          0
##   5   0                   15                 0          0
##
##     Lower middle income Upper middle income
##   1                  39                  95
##   2                   0                   0
##   3                 127                  41
##   4                  34                  74
##   5                   0                   0
```

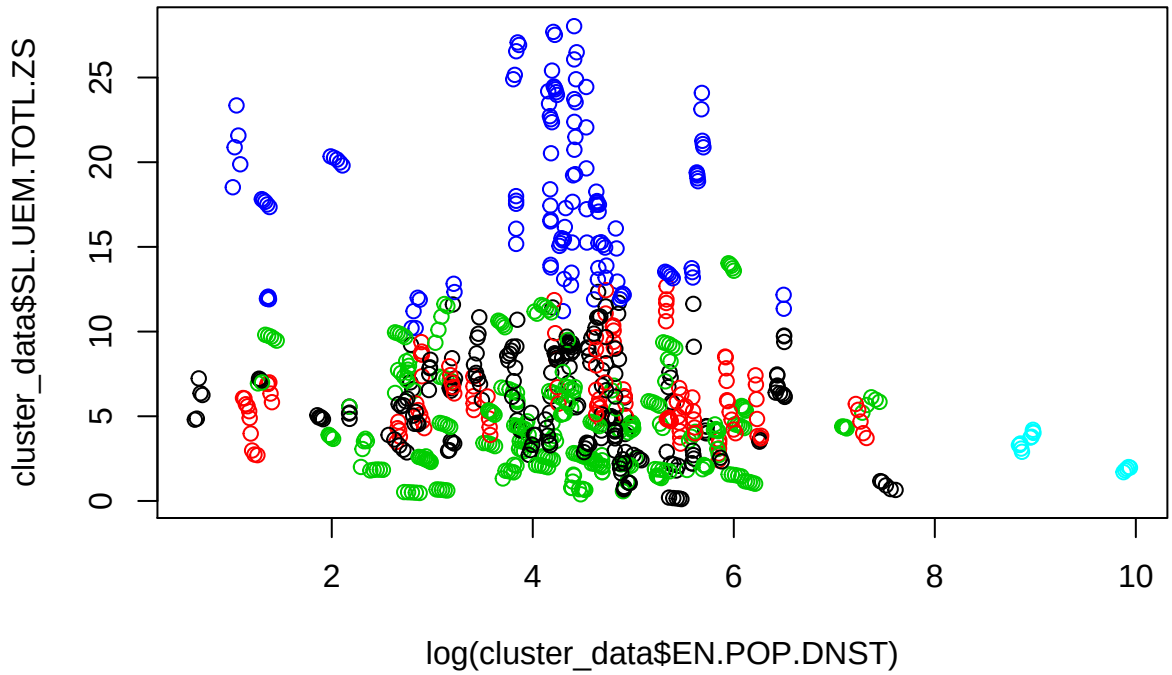
```r
plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.PCAP.CD, col = km_group) # GDP Per Capita
```




```r
plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.MKTP.KD.ZG, col = km_group)
```




```
plot(cluster_data$SL.UEM.TOTL.ZS~log(cluster_data$EN.POP.DNST), col = km_group)
```




```
# # K=7
# km.out_7=kmeans(x_norm,7)
# km_group <- km.out_7$cluster
#
# cluster_data$cluster_group <- km_group
#
# table(cluster_data$cluster_group,cluster_data$IncomeGroup)
# plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.PCAP.CD, col = km_group) # GDP Per Capita
# plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.MKTP.KD.ZG, col = km_group)
```

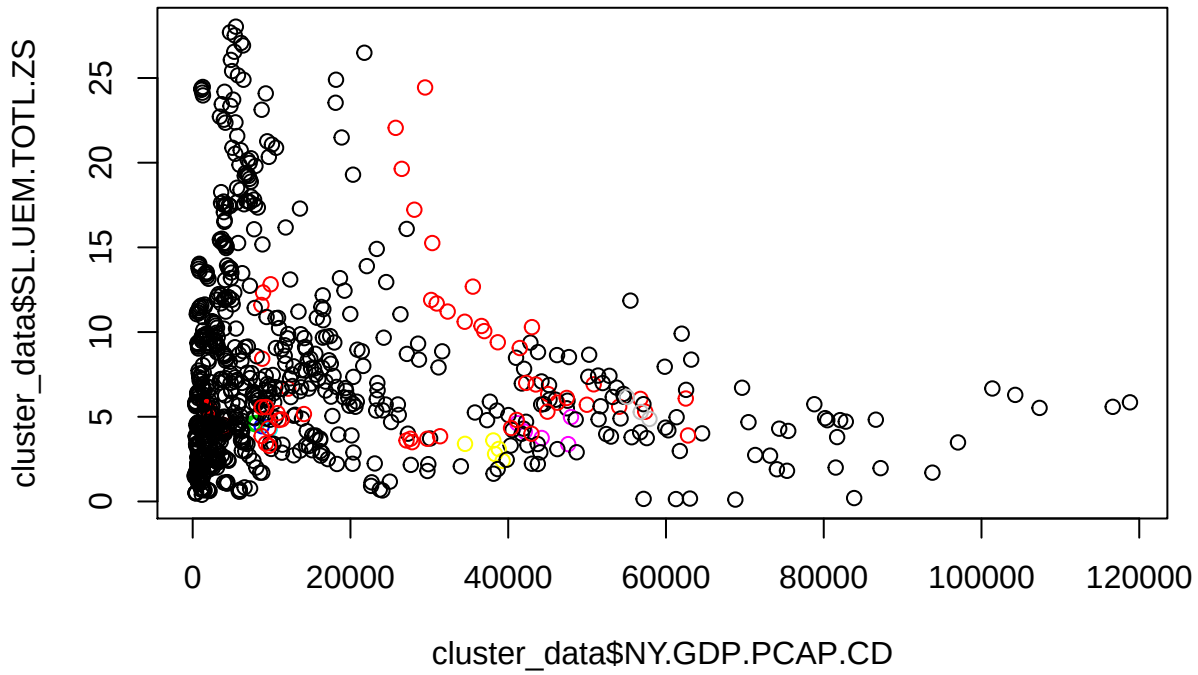
```
# plot(cluster_data$SL.UEM.TOTL.ZS~log(cluster_data$EN.POP.DNST), col = km_group)
#
#
# # K=10
# km.out_10=kmeans(x_norm,10)
# km_group <- km.out_7$cluster
#
# cluster_data$cluster_group <- km_group
#
# table(cluster_data$cluster_group,cluster_data$IncomeGroup)
# plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.PCAP.CD, col = km_group) # GDP Per Capita
# plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.MKTP.KD.ZG, col = km_group)
# plot(cluster_data$SL.UEM.TOTL.ZS~log(cluster_data$EN.POP.DNST), col = km_group)
```

Hierarchical Clustering

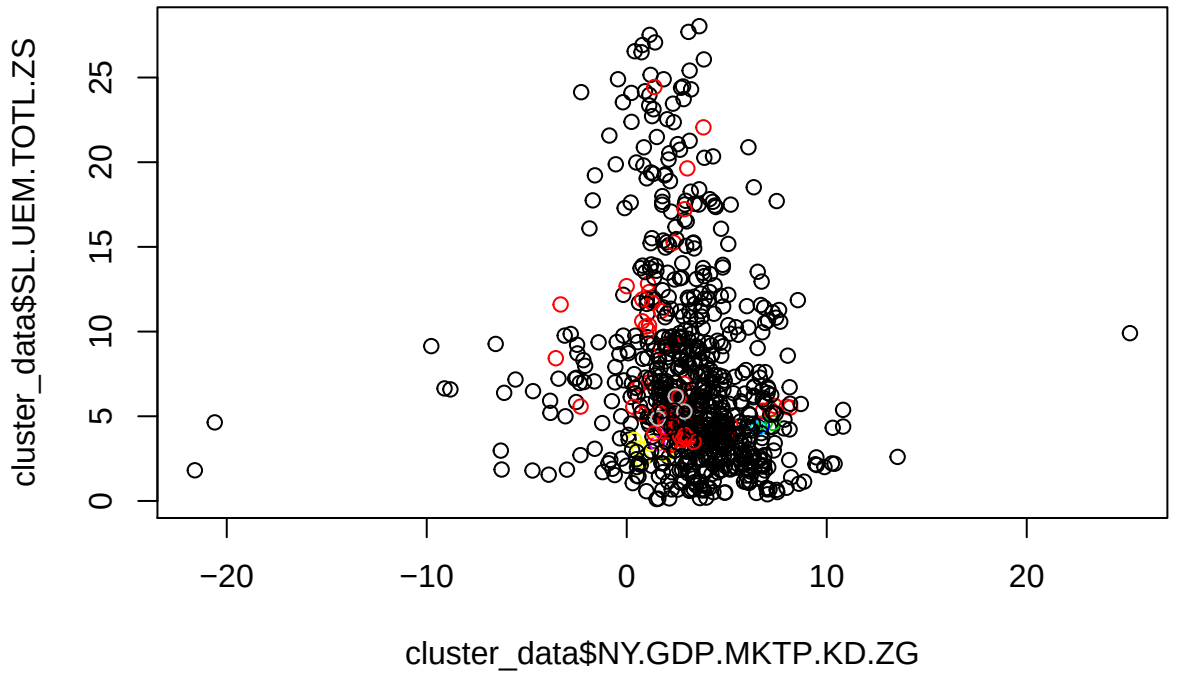
```
# hclust
hclust_avg <- hclust(dist(x), method ="average")
h_group <- cutree(hclust_avg, 10)
table(h_group ,cluster_data$IncomeGroup)
```

```
##
## h_group     High income: nonOECD High income: OECD Low income
##       1    0                  100               110        127
##       2    0                    5                35          0
##       3    0                    0                 0          0
##       4    0                    0                 0          0
##       5    0                    0                 0          0
##       6    0                    0                 5          0
##       7    0                    0                 5          0
##       8    0                    0                 3          0
##       9    0                    0                 1          0
##       10   0                    0                 1          0
##
## h_group Lower middle income Upper middle income
##       1                 193                 195
##       2                   7                  10
##       3                   0                   3
##       4                   0                   1
##       5                   0                   1
##       6                   0                   0
##       7                   0                   0
##       8                   0                   0
##       9                   0                   0
##       10                  0                   0
```

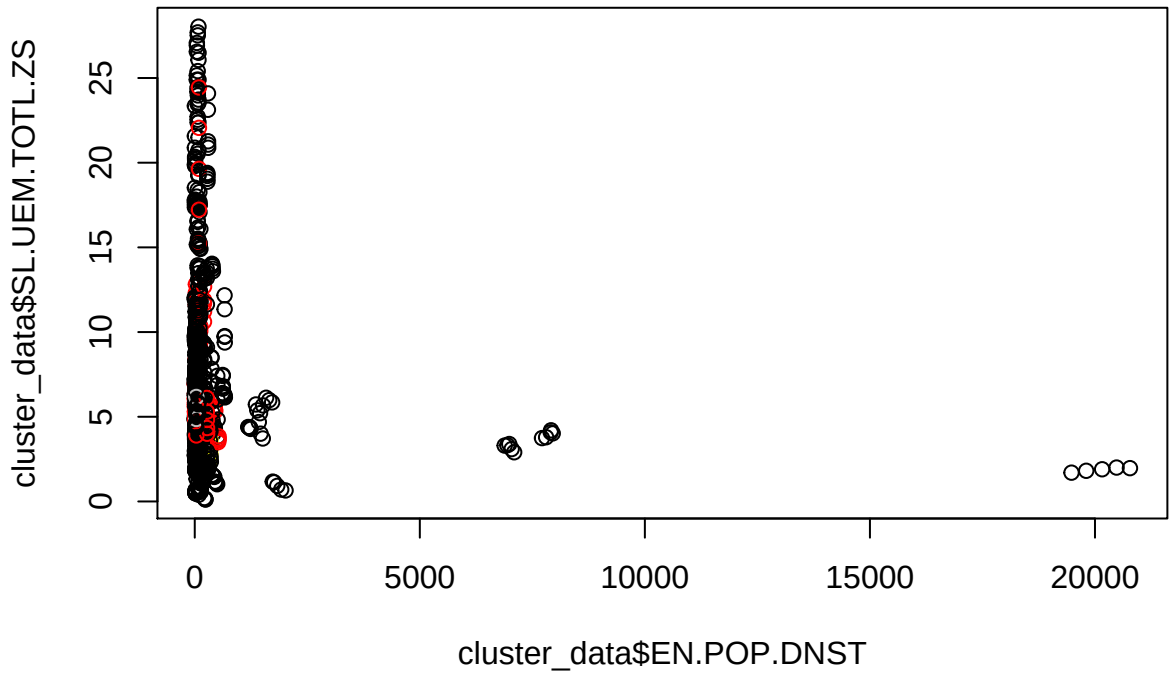
```
##
plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.PCAP.CD, col = h_group)
```




```
plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$NY.GDP.MKTP.KD.ZG, col = h_group)
```




```
plot(cluster_data$SL.UEM.TOTL.ZS~cluster_data$EN.POP.DNST, col = h_group)
```



Part 3 Statistical Learning Models

3.1 Principal Component Regression

```
# library(clusterSim)
library(ggplot2)
library(RColorBrewer)
library(ggpubr)
```

```
## Loading required package: magrittr
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
##
## Attaching package: 'ggpubr'
```

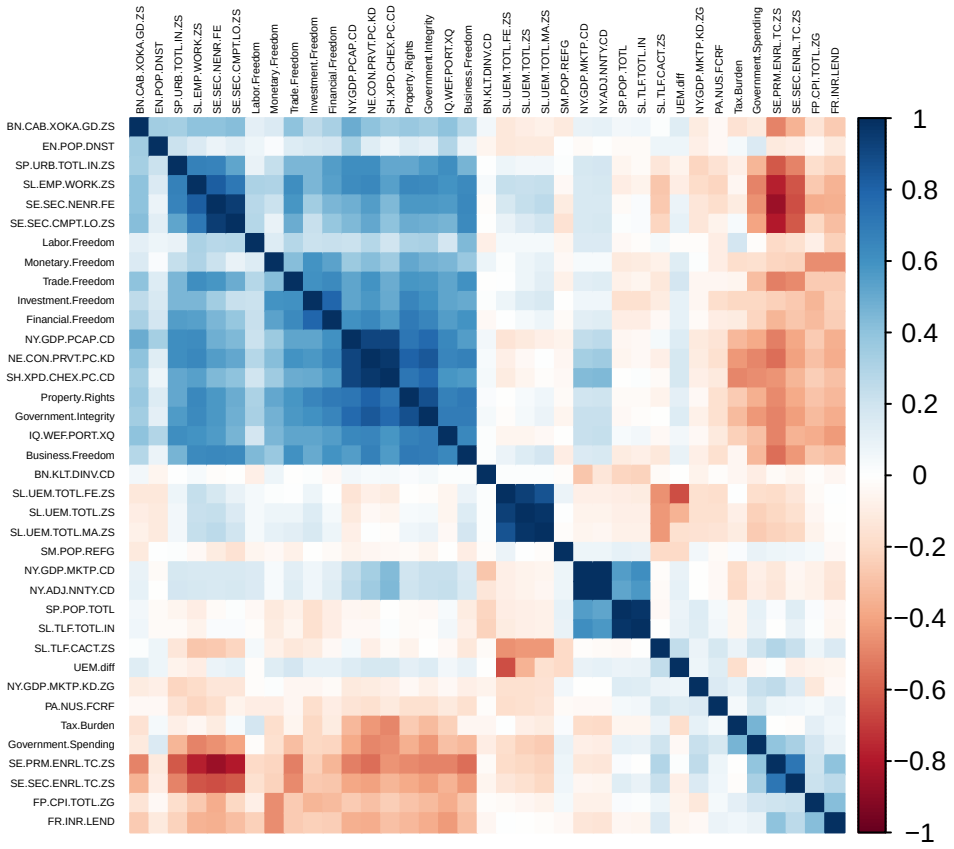
```
## The following object is masked from 'package:cowplot':
##
##     get_legend
```

Load data

```
df=read.csv("clean_df_with_ec.csv")
df['UEM.diff']=df$SL.UEM.TOTL.MA.ZS-df$SL.UEM.TOTL.FE.ZS
```

Visualization of the correlation before PCA

```
df_cor = df
res <- as.data.frame(cor(df_cor[,4:41], method="pearson", use="p"))
# remove the na from correlation matrix
res [,"SI.POV.NAGP"]=NULL
res=res[row.names(res)!="SI.POV.NAGP",]
corrplot::corrplot(as.matrix(res), method= "color", order = "hclust", tl.cex = 0.3,tl.col = "black")
```




```
# format with better view
cor_r = tibble::rownames_to_column(as.data.frame(res), var = "row")
cor_r <- tidyr::gather(cor_r, column, cor, -1)
# check the top 9 correlated var with unemployment rate
cor_r2=cor_r[abs(cor_r$cor)>0.8 & abs(cor_r$cor)!=1  ,]
print(cor_r2)
```

```
##                        row              column          cor
## 2        SL.UEM.TOTL.FE.ZS       SL.UEM.TOTL.ZS  0.9388370255
## 3        SL.UEM.TOTL.MA.ZS       SL.UEM.TOTL.ZS  0.9790007304
## 38          SL.UEM.TOTL.ZS    SL.UEM.TOTL.FE.ZS  0.9388370255
## 40       SL.UEM.TOTL.MA.ZS    SL.UEM.TOTL.FE.ZS  0.8607609909
```

```
## 75         SL.UEM.TOTL.ZS    SL.UEM.TOTL.MA.ZS  0.9790007304
## 76      SL.UEM.TOTL.FE.ZS    SL.UEM.TOTL.MA.ZS  0.8607609909
## 116        SL.TLF.TOTL.IN          SP.POP.TOTL  0.9762279681
## 152           SP.POP.TOTL       SL.TLF.TOTL.IN  0.9762279681
## 271        NY.ADJ.NNTY.CD       NY.GDP.MKTP.CD  0.9984450264
## 310     NE.CON.PRVT.PC.KD       NY.GDP.PCAP.CD  0.9167134938
## 321     SH.XPD.CHEX.PC.CD       NY.GDP.PCAP.CD  0.9108992269
## 415        NY.GDP.MKTP.CD       NY.ADJ.NNTY.CD  0.9984450264
## 490        NY.GDP.PCAP.CD    NE.CON.PRVT.PC.KD  0.9167134938
## 506     SH.XPD.CHEX.PC.CD    NE.CON.PRVT.PC.KD  0.9625608470
## 508       Property.Rights    NE.CON.PRVT.PC.KD  0.8049822205
## 509  Government.Integrity    NE.CON.PRVT.PC.KD  0.8472066184
## 648     SE.PRM.ENRL.TC.ZS       SE.SEC.NENR.FE -0.8559223834
## 650     SE.SEC.CMPT.LO.ZS       SE.SEC.NENR.FE  0.9445200327
## 651        SL.EMP.WORK.ZS       SE.SEC.NENR.FE  0.8212944074
## 684        SE.SEC.NENR.FE    SE.PRM.ENRL.TC.ZS -0.8559223834
## 758        SE.SEC.NENR.FE    SE.SEC.CMPT.LO.ZS  0.9445200327
## 795        SE.SEC.NENR.FE       SL.EMP.WORK.ZS  0.8212944074
## 897        NY.GDP.PCAP.CD    SH.XPD.CHEX.PC.CD  0.9108992269
## 902     NE.CON.PRVT.PC.KD    SH.XPD.CHEX.PC.CD  0.9625608470
## 976     NE.CON.PRVT.PC.KD      Property.Rights  0.8049822205
## 990  Government.Integrity      Property.Rights  0.8761273741
## 1013    NE.CON.PRVT.PC.KD Government.Integrity  0.8472066184
## 1026      Property.Rights Government.Integrity  0.8761273741
```

```
cor_r[ with(cor_r[cor_r$row=="SL.UEM.TOTL.ZS",], order(abs(cor))),]
```

```
##                     row         column              cor
## 17          FR.INR.LEND SL.UEM.TOTL.ZS -8.543562062e-05
## 7           SM.POP.REFG SL.UEM.TOTL.ZS  6.719778636e-03
## 13       BN.KLT.DINV.CD SL.UEM.TOTL.ZS  1.548825570e-02
## 27      Property.Rights SL.UEM.TOTL.ZS  3.223913823e-02
## 14    NE.CON.PRVT.PC.KD SL.UEM.TOTL.ZS -3.348030216e-02
## 28 Government.Integrity SL.UEM.TOTL.ZS  4.122760802e-02
## 32        Labor.Freedom SL.UEM.TOTL.ZS  4.190995428e-02
## 36    Financial.Freedom SL.UEM.TOTL.ZS  4.800450956e-02
## 15       FP.CPI.TOTL.ZG SL.UEM.TOTL.ZS -4.885093567e-02
## 24    SP.URB.TOTL.IN.ZS SL.UEM.TOTL.ZS  5.032094580e-02
## 25    SH.XPD.CHEX.PC.CD SL.UEM.TOTL.ZS -5.121821031e-02
## 26       IQ.WEF.PORT.XQ SL.UEM.TOTL.ZS -5.240688357e-02
## 12       NY.ADJ.NNTY.CD SL.UEM.TOTL.ZS -6.856405918e-02
## 8        NY.GDP.MKTP.CD SL.UEM.TOTL.ZS -6.939514729e-02
## 33     Monetary.Freedom SL.UEM.TOTL.ZS  7.495764453e-02
## 34        Trade.Freedom SL.UEM.TOTL.ZS  7.972724309e-02
## 29           Tax.Burden SL.UEM.TOTL.ZS -8.046055210e-02
## 4           SP.POP.TOTL SL.UEM.TOTL.ZS -8.578612914e-02
## 5        SL.TLF.TOTL.IN SL.UEM.TOTL.ZS -8.945003463e-02
## 11    BN.CAB.XOKA.GD.ZS SL.UEM.TOTL.ZS -9.340603616e-02
## 9        NY.GDP.PCAP.CD SL.UEM.TOTL.ZS -1.179582701e-01
## 23          EN.POP.DNST SL.UEM.TOTL.ZS -1.187972011e-01
## 35   Investment.Freedom SL.UEM.TOTL.ZS  1.403178366e-01
## 31     Business.Freedom SL.UEM.TOTL.ZS  1.415665852e-01
## 21    SE.SEC.CMPT.LO.ZS SL.UEM.TOTL.ZS  1.538970719e-01
## 16          PA.NUS.FCRF SL.UEM.TOTL.ZS -1.605779896e-01
## 10    NY.GDP.MKTP.KD.ZG SL.UEM.TOTL.ZS -1.619462211e-01
```

```
## 20    SE.SEC.ENRL.TC.ZS SL.UEM.TOTL.ZS -1.695676897e-01
## 19    SE.PRM.ENRL.TC.ZS SL.UEM.TOTL.ZS -2.049796527e-01
## 22       SL.EMP.WORK.ZS SL.UEM.TOTL.ZS  2.262304921e-01
## 18       SE.SEC.NENR.FE SL.UEM.TOTL.ZS  2.306146194e-01
## 30  Government.Spending SL.UEM.TOTL.ZS -2.307825606e-01
## 37             UEM.diff SL.UEM.TOTL.ZS -3.498662827e-01
## 6        SL.TLF.CACT.ZS SL.UEM.TOTL.ZS -4.395386424e-01
## 2     SL.UEM.TOTL.FE.ZS SL.UEM.TOTL.ZS  9.388370255e-01
## 3     SL.UEM.TOTL.MA.ZS SL.UEM.TOTL.ZS  9.790007304e-01
## 1        SL.UEM.TOTL.ZS SL.UEM.TOTL.ZS  1.000000000e+00
```

```r
# plot the scatter plot of unemployment rate with top9 correlated
col_prod = rev(brewer.pal(9,"GnBu"))
vis1 = ggplot(df,aes(y=SL.UEM.TOTL.ZS) ) +
  geom_point(aes(x=SL.UEM.TOTL.MA.ZS), color=col_prod[1],alpha=0.7) +
  labs( x='', y = "Unemployment")+rremove("ylab")+rremove('xlab')

vis2 = ggplot(df,aes(y=SL.UEM.TOTL.ZS) ) +
  geom_point(aes(x=SL.UEM.TOTL.FE.ZS), color=col_prod[2],alpha=0.7) +
  labs( x="Unemployment, female (% of female labor force)", y = "Unemployment")+rremove("ylab")+rremove

vis3 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=SL.TLF.CACT.ZS), color=col_prod[3],alpha=0.7) +  labs(
       x="Labor force rate, total (% of total population) ", y = "Unemployment")+rremove("ylab")+rremov

vis4 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=UEM.diff), color=col_prod[4],alpha=0.7) +  labs(
       x="Difference between Male and Female Unemploymente rate", y = "Unemployment")+rremove("ylab")+r

vis5 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=SL.EMP.WORK.ZS), color=col_prod[5],alpha=0.7) +  labs(
       x="Wage and salaried workers, total (% of total employment)", y = "Unemployment")+rremove("ylab"

vis6 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=SE.PRM.ENRL.TC.ZS), color=col_prod[7],alpha=0.7) +  labs(
       x="Pupil-teacher ratio, primary", y = "Unemployment")+rremove("ylab")+rremove('xlab')

vis7 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=Government.Spending), color=col_prod[6],alpha=0.7) +rremove('xlab')+rremove("ylab")


vis8 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=NY.GDP.MKTP.KD.ZG), color=col_prod[8]) +rremove("ylab")+rremove('xlab')


vis9 =  ggplot(df,aes(y=SL.UEM.TOTL.ZS)) +
  geom_point(aes( x=Business.Freedom), color=col_prod[9]) +rremove("ylab")+rremove('xlab')



ggarrange(vis1, vis2,vis3,vis4,vis5,vis7,vis6,vis8,vis9, ncol = 3, nrow = 3, legend = "none" ,
labels = c("Unemployment, male \n(% of male labor force)", "Unemployment, female \n(% of female labor fo
```

Method #1: Fill the missing value and scale the data

Reference: https://rpubs.com/esobolewska/pcr-step-by-step http://pbil.univ-lyon1.fr/members/dray/files/articles/dray2015a.pdf

```
# read and scale data
df=read.csv("onehot_cleaned.csv")
df$X = NULL

#df = dfback
df$SI.POV.NAGP=NULL
df['UEM.diff']=df$SL.UEM.TOTL.MA.ZS-df$SL.UEM.TOTL.FE.ZS
df_scale = data.frame(scale(df[,c(4,7:39,45)]))
names(df_scale)[1] = 'y'
df_scale['y_unscaled'] = df$SL.UEM.TOTL.ZS
df_scale['year'] = df$year
```

Dealing with missing value for PCA

```
set.seed(101)
library(missMDA)

# estimate number of components
nb <- estim_ncpPCA(df_scale[1:35],ncp.max=6,method = c("Regularized"))
# actual impute
x.impute <- imputePCA(df_scale[1:35],nb$ncp)
x_imp = as.data.frame(x.impute)
```

```r
x_imp["year"]=df_scale['year']
x_imp["y_unscaled"]=df_scale['y_unscaled']
names(x_imp)[1]='y'

row.18 = which(x_imp$year == 2018)
x.18=x_imp[row.18,]
x.yr=x_imp[-row.18,]
#final dim of imptation missing value of data
dim(x_imp)
```

```
## [1] 825  72
```

```r
#[1] 825  72
# variable: 2:35
# final dim for fill data is (825,35)
```

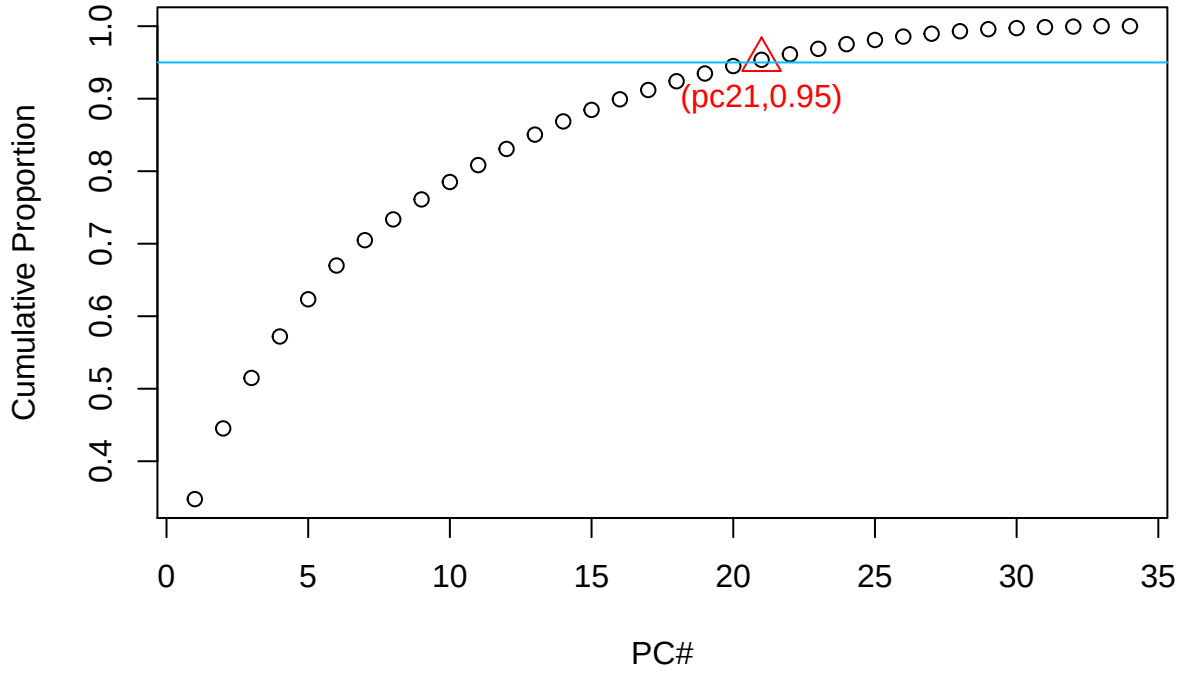
PCR

```r
# fit pca with completeObs
pca.fit = prcomp(x.yr[,c(2:35)])
summary(pca.fit)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     3.39901 1.799431 1.520892 1.378167 1.305333
## Proportion of Variance 0.34778 0.097470 0.069630 0.057170 0.051290
## Cumulative Proportion  0.34778 0.445250 0.514880 0.572060 0.623350
##                             PC6      PC7       PC8       PC9      PC10
## Standard deviation     1.243908 1.076912 0.9768205 0.9567377 0.8929801
## Proportion of Variance 0.046580 0.034910 0.0287200 0.0275500 0.0240000
## Cumulative Proportion  0.669930 0.704840 0.7335600 0.7611100 0.7851200
##                            PC11      PC12     PC13      PC14     PC15
## Standard deviation     0.8793978 0.8610336 0.810367 0.7775199 0.726698
## Proportion of Variance 0.0232800 0.0223200 0.019770 0.0182000 0.015900
## Cumulative Proportion  0.8084000 0.8307100 0.850480 0.8686800 0.884580
##                            PC16      PC17      PC18      PC19      PC20
## Standard deviation     0.6957981 0.6536296 0.6329276 0.5958684 0.5834901
## Proportion of Variance 0.0145700 0.0128600 0.0120600 0.0106900 0.0102500
## Cumulative Proportion  0.8991500 0.9120100 0.9240700 0.9347600 0.9450100
##                            PC21      PC22      PC23      PC24      PC25
## Standard deviation     0.5348669 0.5045697 0.4986885 0.4626089 0.4377567
## Proportion of Variance 0.0086100 0.0076600 0.0074900 0.0064400 0.0057700
## Cumulative Proportion  0.9536200 0.9612800 0.9687700 0.9752100 0.9809800
##                            PC26      PC27      PC28      PC29      PC30
## Standard deviation     0.3939014 0.3663717 0.3338583 0.3061012 0.2228207
## Proportion of Variance 0.0046700 0.0040400 0.0033600 0.0028200 0.0014900
## Cumulative Proportion  0.9856500 0.9896900 0.9930500 0.9958700 0.9973600
##                            PC31      PC32      PC33       PC34
## Standard deviation     0.2093755 0.1605943 0.1309322 0.02977715
## Proportion of Variance 0.0013200 0.0007800 0.0005200 0.00003000
## Cumulative Proportion  0.9986800 0.9994600 0.9999700 1.00000000
```

```r
# plot the Cumulative Proportion to choose the number of pc
plot(summary(pca.fit)$importance[3,],ylab='Cumulative Proportion',xlab='PC#')
points(21,summary(pca.fit)$importance[3,][21],pch=2,cex=2,col='red')
abline(h=0.95,col='deepskyblue1',lty=1)
```

```
text(21,0.9,'(pc21,0.95)',col='red',cex=1)
```




```
ols.data <- cbind(x.yr$y, as.data.frame(pca.fit$x[,1:21]))
names(ols.data)[1]='y'
fit.reg <- lm(y~ ., data = ols.data)
summary(fit.reg)
```

```
##
## Call:
## lm(formula = y ~ ., data = ols.data)
##
## Residuals:
##         Min          1Q      Median          3Q         Max
## -1.77105068 -0.51340596 -0.08896767  0.28625681  3.10989872
##
## Coefficients:
##                 Estimate   Std. Error  t value   Pr(>|t|)
## (Intercept)  0.024257904  0.031139364  0.77901 0.43626212
## PC1         -0.033814187  0.009168253 -3.68818 0.00024511 ***
## PC2         -0.099192506  0.017318237 -5.72763 1.5696e-08 ***
## PC3         -0.204856313  0.020489933 -9.99790 < 2.22e-16 ***
## PC4         -0.091902861  0.022611907 -4.06436 5.4178e-05 ***
## PC5         -0.121729189  0.023873580 -5.09891 4.5090e-07 ***
## PC6          0.298221903  0.025052480 11.90389 < 2.22e-16 ***
## PC7          0.084985892  0.028937347  2.93689 0.00343469 **
## PC8          0.057066002  0.031902464  1.78876 0.07412704 .
## PC9          0.075993191  0.032572125  2.33307 0.01995412 *
## PC10        -0.096182009  0.034897732 -2.75611 0.00601644 **
## PC11         0.019741524  0.035436728  0.55709 0.57765983
## PC12         0.054107848  0.036192526  1.49500 0.13540873
## PC13        -0.176323444  0.038455392 -4.58514 5.4610e-06 ***
## PC14         0.051197394  0.040079977  1.27738 0.20193247
## PC15         0.057846864  0.042882987  1.34895 0.17783266
```

```
## PC16         0.230091852  0.044787391  5.13742 3.7061e-07 ***
## PC17         0.010588229  0.047676823  0.22208 0.82432006
## PC18         0.139090114  0.049236249  2.82495 0.00487656 **
## PC19        -0.123834856  0.052298425 -2.36785 0.01818900 *
## PC20        -0.113769934  0.053407900 -2.13021 0.03353640 *
## PC21         0.012313133  0.058263053  0.21134 0.83269188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7999847 on 638 degrees of freedom
## Multiple R-squared:  0.400986,   Adjusted R-squared:  0.3812692
## F-statistic: 20.33731 on 21 and 638 DF,  p-value: < 2.2204e-16
```

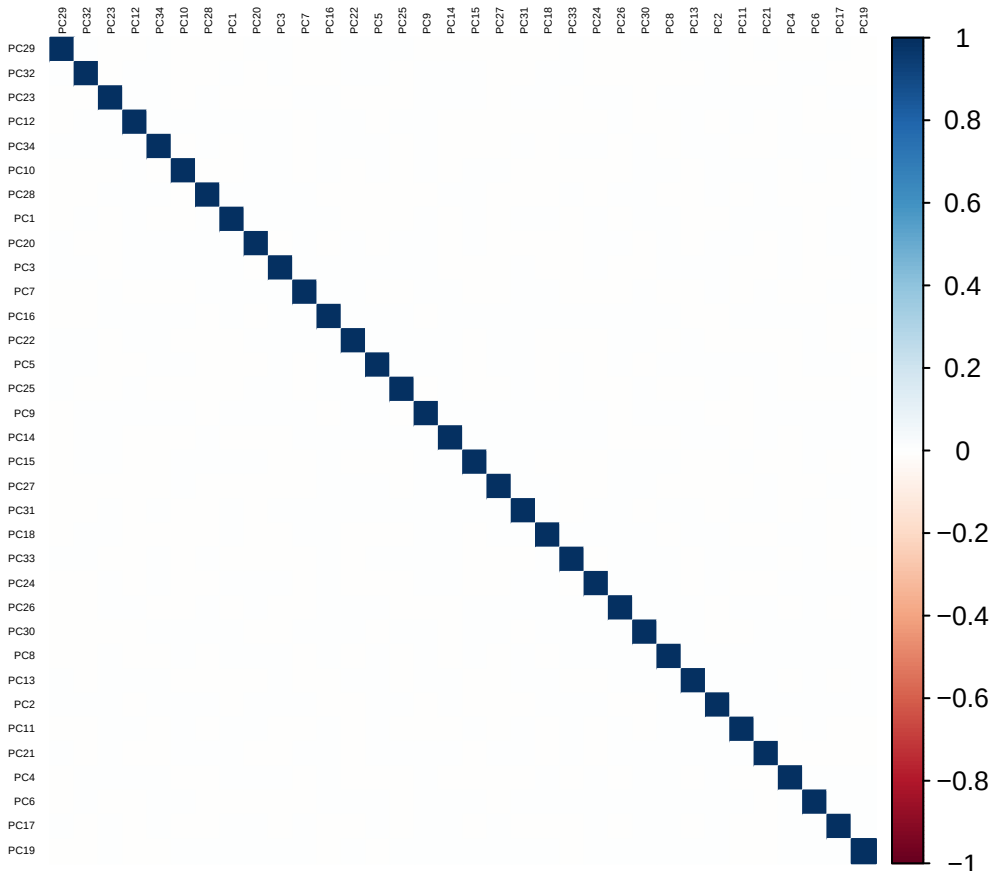
```
#Multiple R-squared:  0.401,

# get pca test
pca.test = predict(pca.fit,newdata=x.18[,c(2:35)])
ols.data2 = cbind(x.18$y, as.data.frame(pca.test[,1:21]))
# get regression test
test.pred = predict(fit.reg, newdata = ols.data2)
train.pred = predict(fit.reg, newdata = ols.data)

rmse.test = mean((x.18$y-test.pred)^2)
rmse.train = mean((x.yr$y-train.pred)^2)
cat("rmse.test = ",rmse.test)
```

```
## rmse.test =  0.5620508577
```

```
cat("rmse.train = ",rmse.train)
```

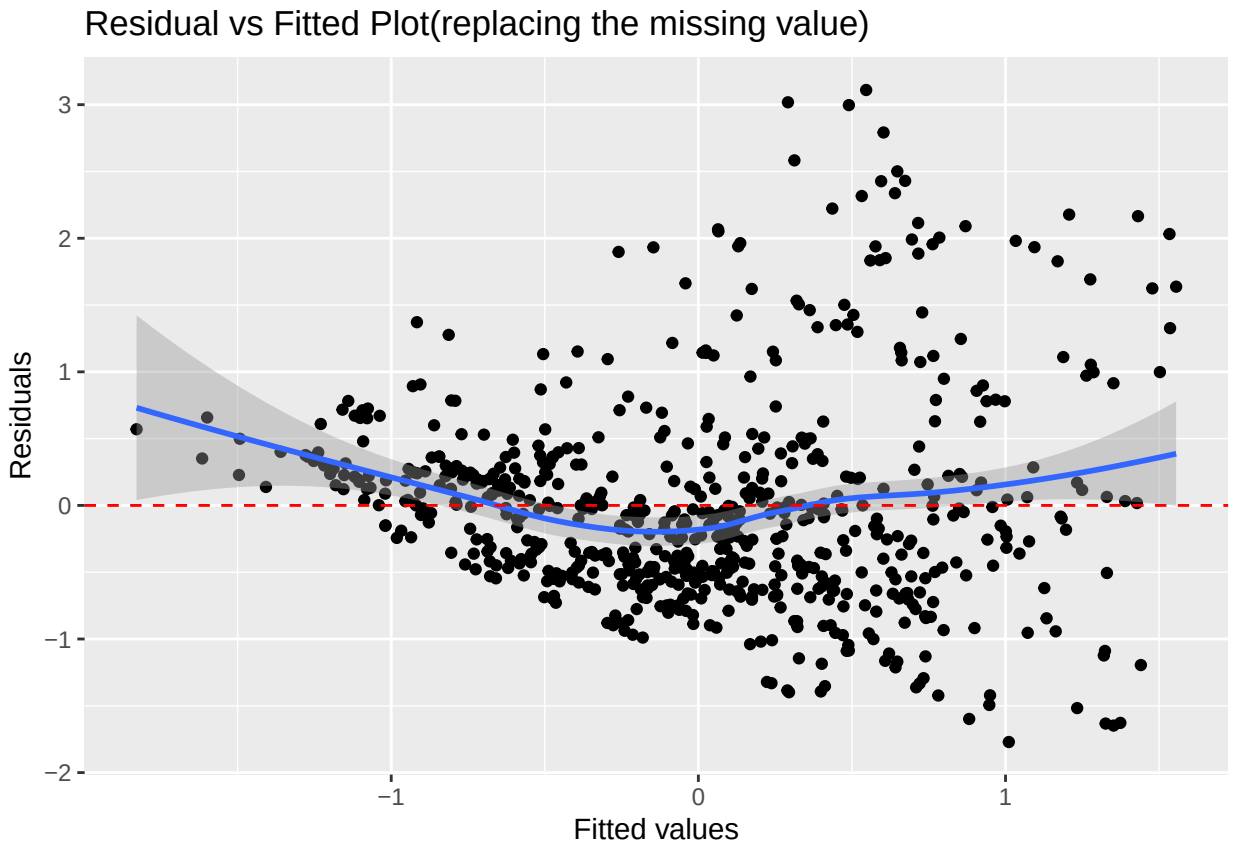
```
## rmse.train =  0.6186430586
```

```
#rmse.test =  0.5620509
#rmse.train =  0.6186431

# plot the correlation matrix again
res1 <- cor(pca.fit$x, method="pearson")
corrplot::corrplot(res1, method= "color", order = "hclust", tl.cex = 0.3,tl.col = "black")
```

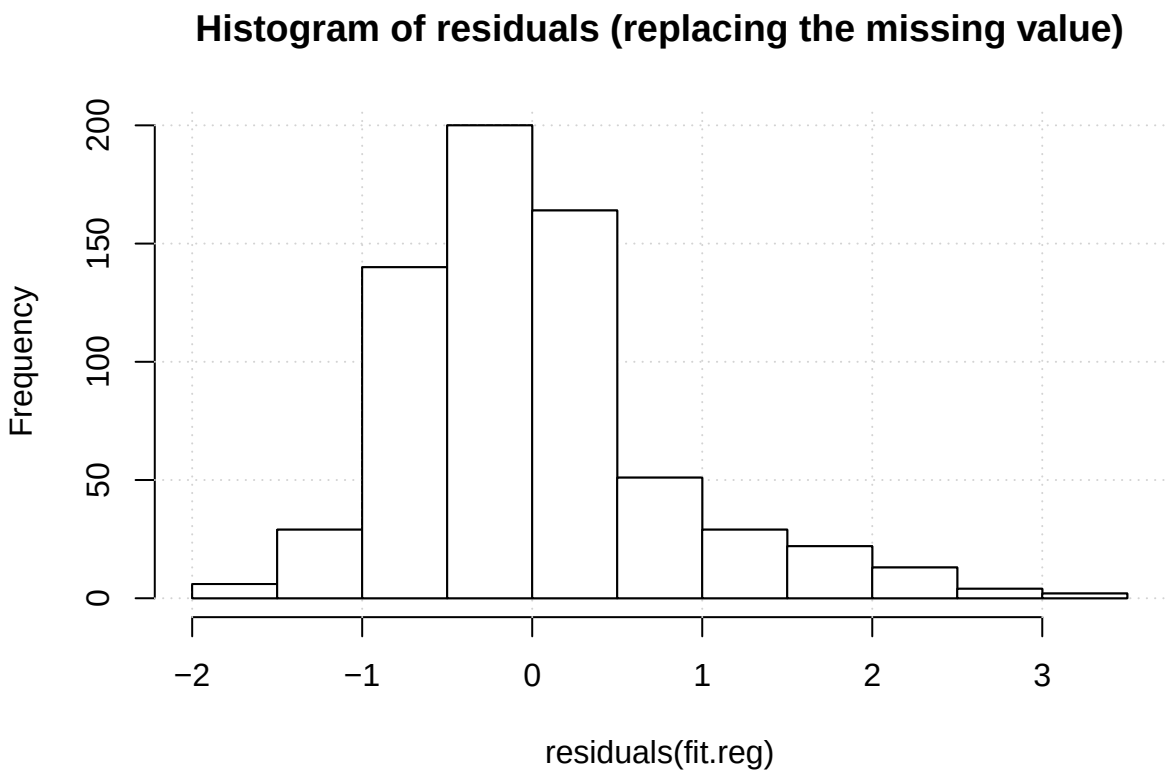


Plot the results

```
col_prod = brewer.pal(12,"Paired")
# plot the resid analysis
p1<-ggplot(fit.reg, aes(.fitted, .resid))+geom_point()
p1<-p1+stat_smooth(method="loess")+geom_hline(yintercept=0, col="red", linetype="dashed")
p1<-p1+xlab("Fitted values")+ylab("Residuals")
p1<-p1+ggtitle("Residual vs Fitted Plot(replacing the missing value)")+scale_fill_brewer( palette = 3)
p1
```




```
hist(residuals(fit.reg),main='Histogram of residuals (replacing the missing value)')
grid()
hist(residuals(fit.reg), add = TRUE, col = 'white',main='Histogram of residuals(replacing the missing va
```




```
#### plot prediction  with orginal value of y
ols.data <- cbind(x.yr$y_unscaled, as.data.frame(pca.fit$x[,1:22]))
names(ols.data)[1]='y_unscaled'
fit.reg <- lm(y_unscaled~ ., data = ols.data)
summary(fit.reg)
```

```
##
## Call:
## lm(formula = y_unscaled ~ ., data = ols.data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -9.9719666 -2.9153493 -0.5459102  1.6616075 17.4125697
##
## Coefficients:
##                Estimate  Std. Error  t value   Pr(>|t|)
## (Intercept)  7.46496970  0.17632540 42.33633 < 2.22e-16 ***
## PC1         -0.19146554  0.05191487 -3.68807 0.00024525 ***
## PC2         -0.56165617  0.09806383 -5.72746 1.5722e-08 ***
## PC3         -1.15995468  0.11602343 -9.99759 < 2.22e-16 ***
## PC4         -0.52038012  0.12803902 -4.06423 5.4216e-05 ***
## PC5         -0.68926527  0.13518319 -5.09875 4.5146e-07 ***
## PC6          1.68861719  0.14185867 11.90352 < 2.22e-16 ***
## PC7          0.48121428  0.16385657  2.93680 0.00343587 **
## PC8          0.32312392  0.18064643  1.78871 0.07413675 .
## PC9          0.43029505  0.18443836  2.33300 0.01995843 *
## PC10        -0.54460988  0.19760702 -2.75602 0.00601826 **
## PC11         0.11178212  0.20065906  0.55707 0.57767195
## PC12         0.30637401  0.20493874  1.49495 0.13542162
## PC13        -0.99839346  0.21775212 -4.58500 5.4661e-06 ***
```

```
## PC14         0.28989420  0.22695127  1.27734 0.20194719
## PC15         0.32754539  0.24282321  1.34890 0.17784686
## PC16         1.30284547  0.25360681  5.13727 3.7107e-07 ***
## PC17         0.05995356  0.26996811  0.22208 0.82432553
## PC18         0.78756776  0.27879830  2.82487 0.00487811 **
## PC19        -0.70118815  0.29613775 -2.36778 0.01819305 *
## PC20        -0.64419770  0.30242010 -2.13014 0.03354248 *
## PC21         0.06972046  0.32991221  0.21133 0.83269711
## PC22         0.34272296  0.34972203  0.97999 0.32746469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.529882 on 637 degrees of freedom
## Multiple R-squared:  0.4018877,  Adjusted R-squared:  0.3812308
## F-statistic: 19.45534 on 22 and 637 DF,  p-value: < 2.2204e-16
```

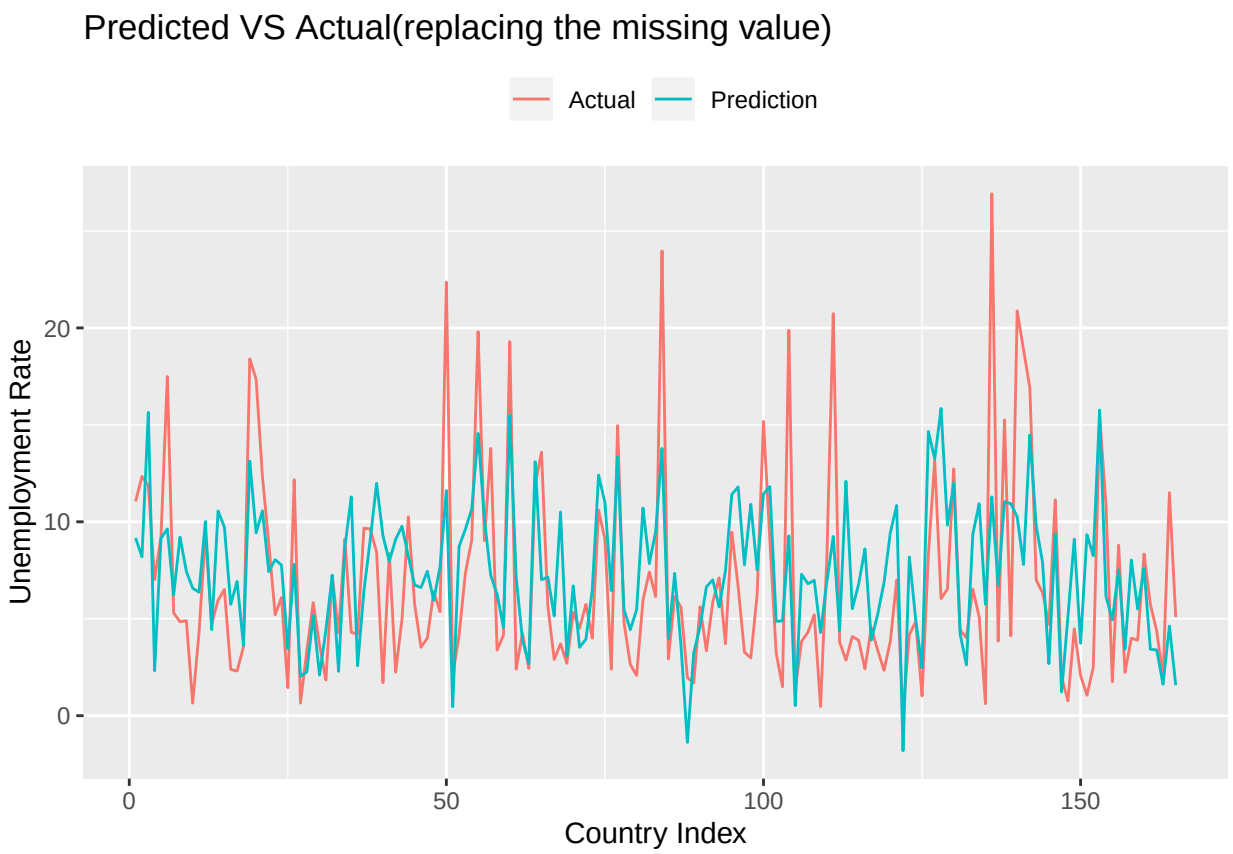
```
pca.test = predict(pca.fit,newdata=x.18[,c(2:40)])
ols.data2 = cbind(x.18$y_unscaled, as.data.frame(pca.test[,1:22]))
test.pred = predict(fit.reg, newdata = ols.data2)
train.pred = predict(fit.reg, newdata = ols.data)

rmse.test = mean((x.18$y_unscaled-test.pred)^2)
rmse.train = mean((x.yr$y_unscaled-train.pred)^2)
cat("rmse.test = ",rmse.test)
```

```
## rmse.test =  17.89699133
```

```
cat("rmse.train = ",rmse.train)
```

```
## rmse.train =  19.80474287
```

```
#rmse.test =  17.89699
#rmse.train =  19.80474


col_prod = brewer.pal(12,"Paired")
melt_pred1 = reshape2::melt(data.frame(Actual=x.18$y_unscaled,Prediction=test.pred,x=1:165),"x")
ggplot(melt_pred1, aes(x=x)) +
  geom_line(aes(y=value,col=variable)) +
  labs(title="Predicted VS Actual(replacing the missing value)",
       y="Unemployment Rate",
       x = "Country Index",
       color=NULL) +  # title
scale_fill_brewer( palette = 3) +
     theme(legend.position="top")
```



Method #2: Use the dataset without any missing value

```
df_na = read.csv("nonNA.csv")
df_na$X = NULL
df_na['UEM.diff']=df_na$SL.UEM.TOTL.MA.ZS-df_na$SL.UEM.TOTL.FE.ZS
df_na_scale = data.frame(scale(df_na[,c(4,7:25,33)]))
names(df_na_scale)[1] = 'y'
df_na_scale['y_unscaled'] = df_na$SL.UEM.TOTL.ZS
df_na_scale['year'] = df_na$year

row.18 = which(df_na_scale$year == 2018)

x.18=df_na_scale[row.18,]
x.yr=df_na_scale[-row.18,]
dim(df_na_scale)
```

```
## [1] 802  23
```

```
#dim of remove:
# variable 2:21
```

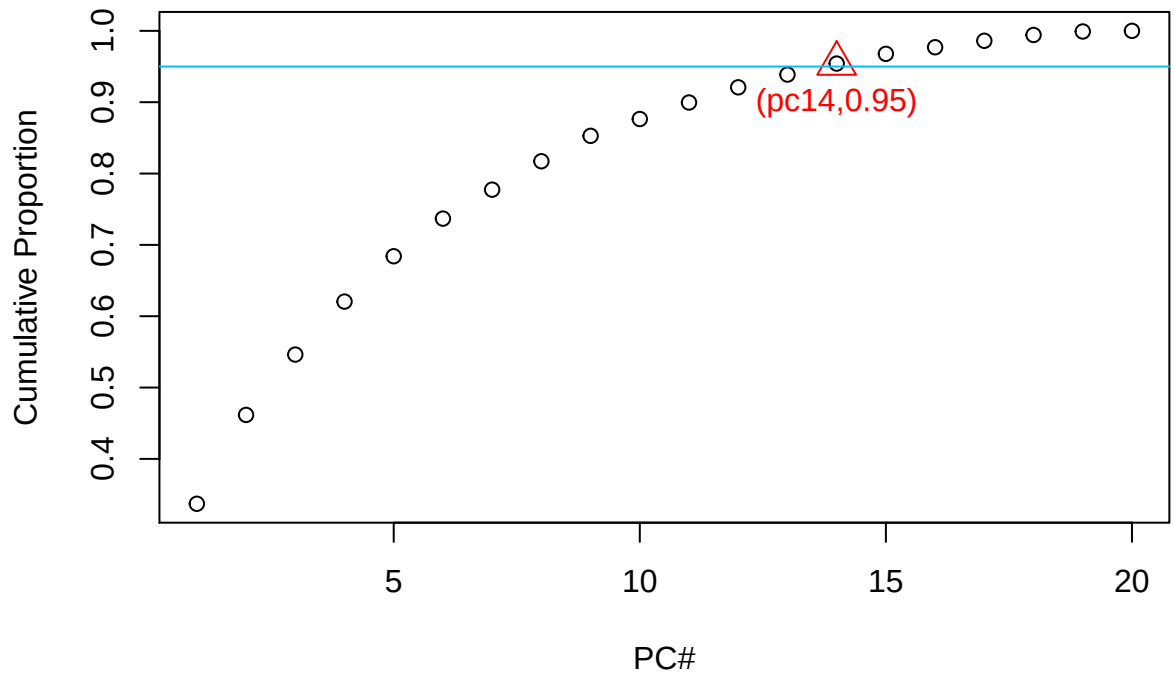
Residuals analysis

```
set.seed(101)

# fit pca with completeObs
pca.fit = prcomp(x.yr[,c(2:21)])
```

```r
summary(pca.fit)
```

```
## Importance of components:
##                             PC1      PC2     PC3      PC4      PC5
## Standard deviation     2.605516 1.583721 1.30517 1.223238 1.131091
## Proportion of Variance 0.337100 0.124550 0.08459 0.074300 0.063530
## Cumulative Proportion  0.337100 0.461650 0.54623 0.620530 0.684060
##                             PC6       PC7       PC8       PC9      PC10
## Standard deviation     1.030682 0.9041037 0.8956388 0.8467374 0.6893376
## Proportion of Variance 0.052750 0.0405900 0.0398300 0.0356000 0.0236000
## Cumulative Proportion  0.736810 0.7774000 0.8172300 0.8528400 0.8764300
##                            PC11      PC12      PC13      PC14      PC15
## Standard deviation     0.6827329 0.6545296 0.5999168 0.5572683 0.5243753
## Proportion of Variance 0.0231500 0.0212700 0.0178700 0.0154200 0.0136500
## Cumulative Proportion  0.8995800 0.9208500 0.9387200 0.9541400 0.9678000
##                            PC16      PC17      PC18     PC19     PC20
## Standard deviation     0.4314655 0.4276748 0.4051254 0.311669 0.134957
## Proportion of Variance 0.0092400 0.0090800 0.0081500 0.004820 0.000900
## Cumulative Proportion  0.9770400 0.9861200 0.9942700 0.999100 1.000000
```

```r
# select the pca that explained 95% variable
plot(summary(pca.fit)$importance[3,],ylab='Cumulative Proportion',xlab='PC#')
points(14,summary(pca.fit)$importance[3,][14],pch=2,cex=2,col='red')
abline(h=0.95,col='deepskyblue1',lty=1)
text(14,0.9,'(pc14,0.95)',col='red',cex=1)
```

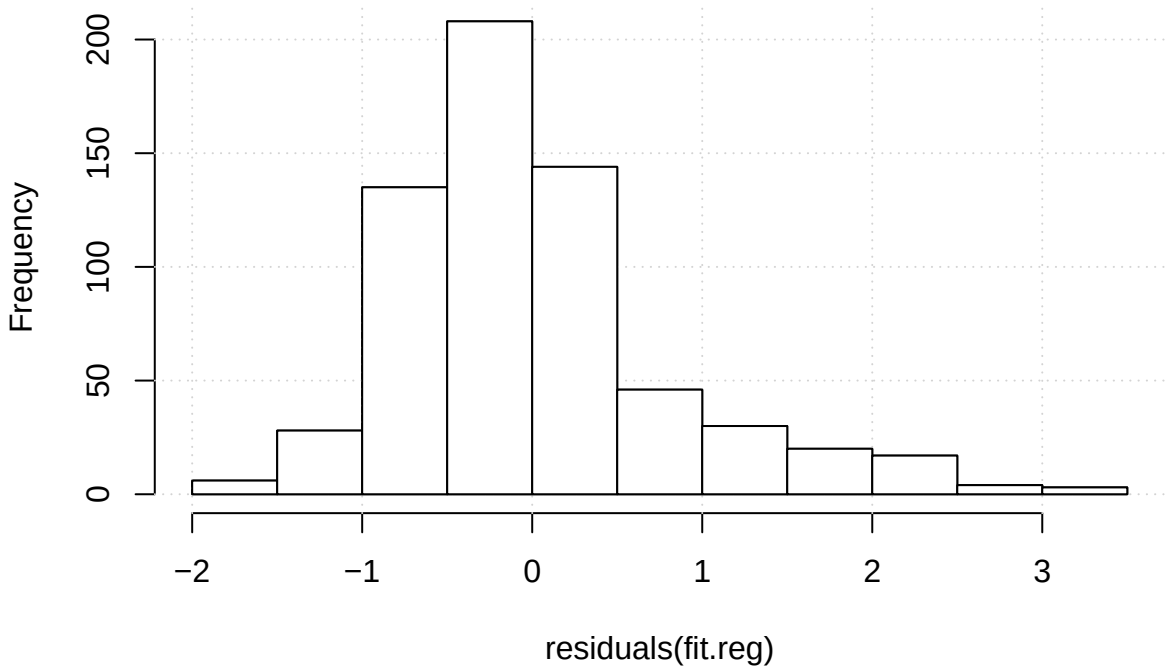



```r
ols.data <- cbind(x.yr$y, as.data.frame(pca.fit$x[,1:14]))
names(ols.data)[1]='y'
fit.reg <- lm(y~ ., data = ols.data)
summary(fit.reg)
```

```
##
## Call:
```

```
## lm(formula = y ~ ., data = ols.data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.7974179 -0.5268081 -0.1278413  0.2856818  3.1745075
##
## Coefficients:
##                Estimate  Std. Error   t value   Pr(>|t|)
## (Intercept)  0.02371725  0.03233672   0.73345 0.46356088
## PC1         -0.05810005  0.01242057  -4.67773 3.5562e-06 ***
## PC2          0.09150050  0.02043414   4.47783 8.9621e-06 ***
## PC3         -0.25867201  0.02479522 -10.43233 < 2.22e-16 ***
## PC4         -0.19735392  0.02645600  -7.45970 2.9085e-13 ***
## PC5          0.25433297  0.02861128   8.88925 < 2.22e-16 ***
## PC6         -0.09649928  0.03139860  -3.07336 0.00220863 **
## PC7         -0.15904774  0.03579454  -4.44335 1.0474e-05 ***
## PC8         -0.04486778  0.03613285  -1.24174 0.21479574
## PC9          0.16475718  0.03821962   4.31080 1.8891e-05 ***
## PC10        -0.03941306  0.04694649  -0.83953 0.40149157
## PC11         0.15869959  0.04740064   3.34805 0.00086263 ***
## PC12        -0.10413064  0.04944311  -2.10607 0.03559584 *
## PC13        -0.09810721  0.05394411  -1.81868 0.06943759 .
## PC14         0.17285643  0.05807252   2.97656 0.00302755 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8187005 on 626 degrees of freedom
## Multiple R-squared:  0.3672056,  Adjusted R-squared:  0.3530536
## F-statistic: 25.94735 on 14 and 626 DF,  p-value: < 2.2204e-16
```

```
hist(residuals(fit.reg),main="Histogram of residuals (removing the missing value)",)
grid()
hist(residuals(fit.reg), add = TRUE, col = 'white',main='Histogram of residuals')
```




```
pca.test = predict(pca.fit,newdata=x.18[,c(2:21)])
ols.data2 = cbind(x.18$y, as.data.frame(pca.test[,1:14]))
test.pred = predict(fit.reg, newdata = ols.data2)
train.pred = predict(fit.reg, newdata = ols.data)

rmse.test = mean((x.18$y-test.pred)^2)
#0.6027348
rmse.train = mean((x.yr$y-train.pred)^2)
#0.6545855
cat("rmse.test = ",rmse.test)
```

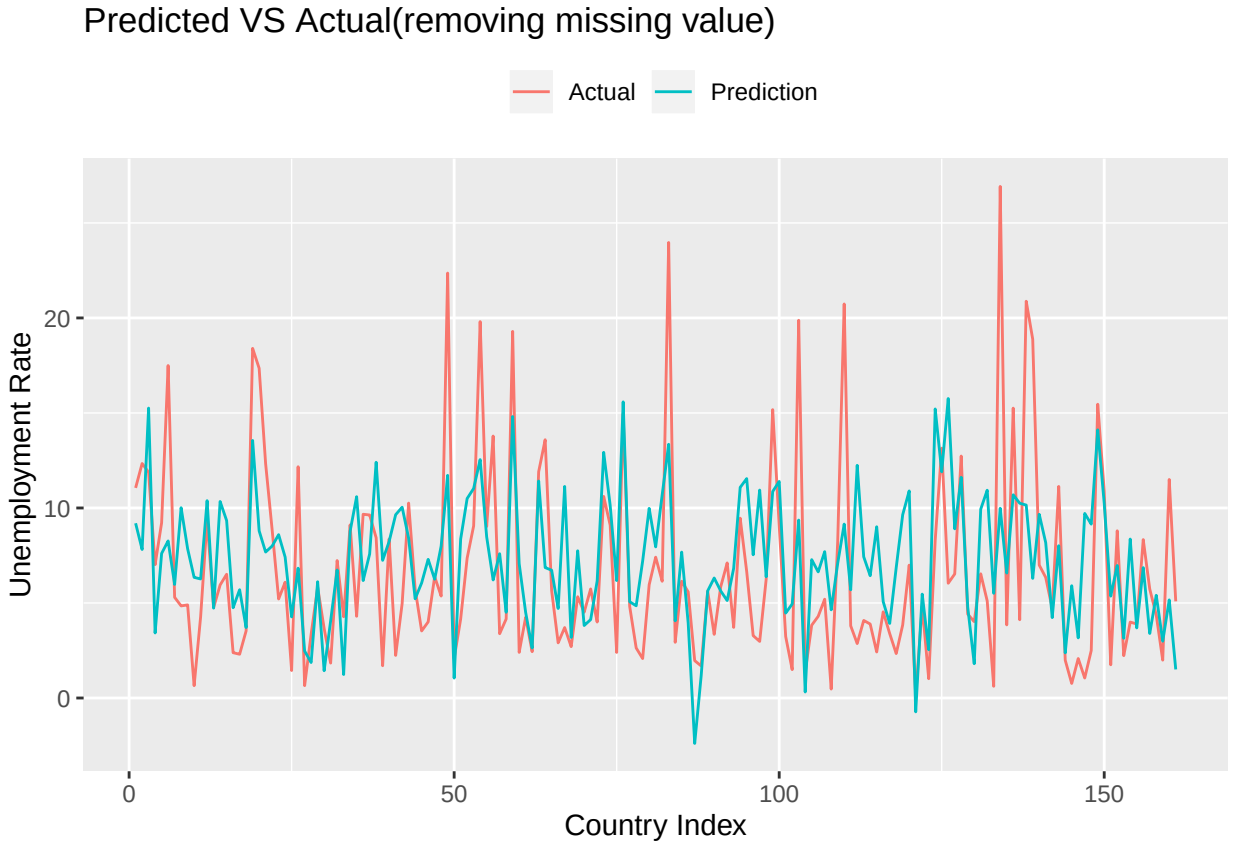
```
## rmse.test =  0.602734833
```

```
cat("rmse.train = ",rmse.train)
```

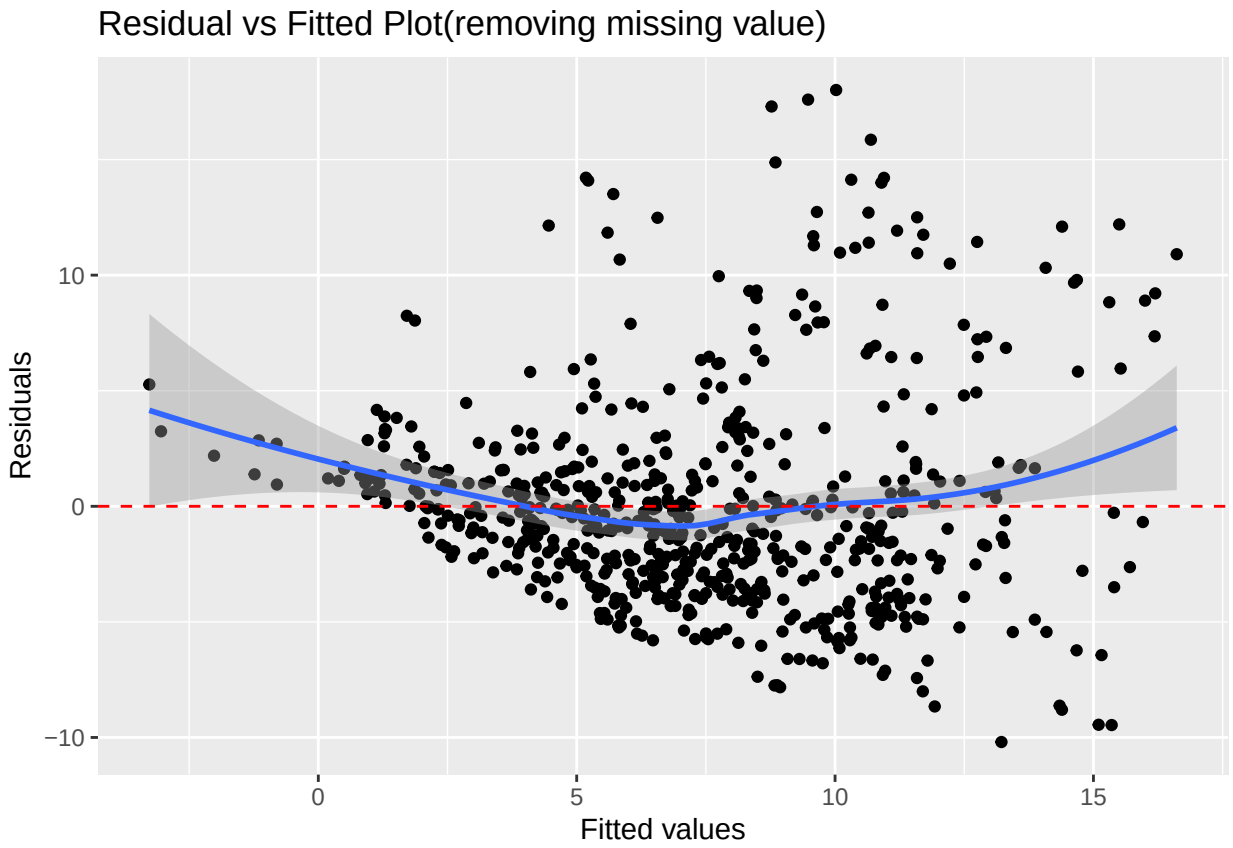
```
## rmse.train =  0.6545855055
```

```
### ploting with unscaled data
ols.data <- cbind(x.yr$y_unscaled, as.data.frame(pca.fit$x[,1:14]))
names(ols.data)[1]='y_unscaled'
fit.reg <- lm(y_unscaled~ ., data = ols.data)
#summary(fit.reg)

pca.test = predict(pca.fit,newdata=x.18[,c(2:21)])
ols.data2 = cbind(x.18$y_unscaled, as.data.frame(pca.test[,1:14]))
test.pred = predict(fit.reg, newdata = ols.data2)
train.pred = predict(fit.reg, newdata = ols.data)


col_prod = brewer.pal(12,"Paired")
melt_pred1 = reshape2::melt(data.frame(Actual=x.18$y_unscaled,Prediction=test.pred,x=1:161),"x")
```

```
ggplot(melt_pred1, aes(x=x)) +
  geom_line(aes(y=value,col=variable)) +
  labs(title="Predicted VS Actual(removing missing value)",
       y="Unemployment Rate",
       x = "Country Index",
       color=NULL) +  # title
scale_fill_brewer( palette = 3) +
     theme(legend.position="top")
```




```
p1<-ggplot(fit.reg, aes(.fitted, .resid))+geom_point()
p1<-p1+stat_smooth(method="loess")+geom_hline(yintercept=0, col="red", linetype="dashed")
p1<-p1+xlab("Fitted values")+ylab("Residuals")
p1<-p1+ggtitle("Residual vs Fitted Plot(removing missing value)")+scale_fill_brewer( palette = 3)
p1
```




```
print("Method 1 (filling NAs) got a better result than method 2 (removing all the NAs).")
```

```
## [1] "Method 1 (filling NAs) got a better result than method 2 (removing all the NAs)."
```

3.2 Random Forests

Data preparation

```
df<-df_n
# rename the response
colnames(df)[4] = 'unem_rate'

# feature engineering

# new feature: difference in unemployment rate between male and female
df$diff = df$SL.UEM.TOTL.MA.ZS-df$SL.UEM.TOTL.FE.ZS

# drop country and male & female unemployment rate
df <- subset(df, select = -c(1,2,3,5,6,IncomeGroup,Region))

# reorder columns
df<- subset(df, select=c(1,ncol(df),(ncol(df)-1),(4:ncol(df)-2)))
# scale
df[,1:(ncol(df)-12)]<- scale(df[,1:(ncol(df)-12)])
```

Build a single tree and illustrate it

```
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

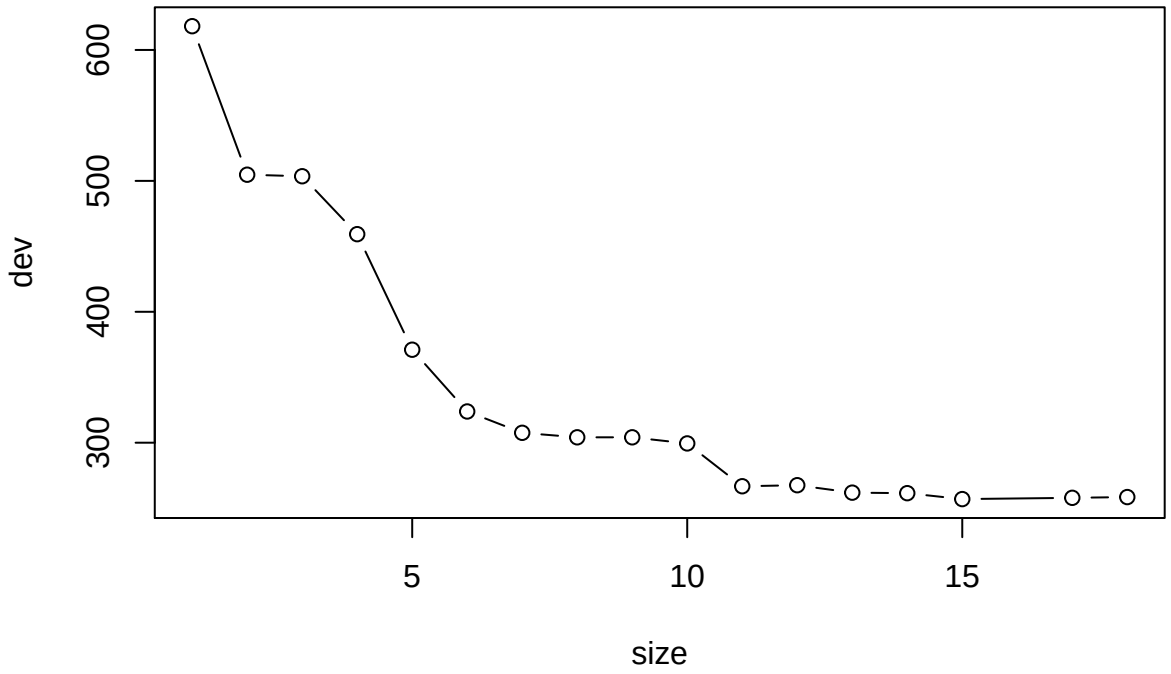
```
set.seed(101)
training = sample(1:nrow(df), 0.75*nrow(df))
train = df[training,]
test = df[-training,]
mytree = tree(unem_rate~., data=train)
summary(mytree)
```

```
##
## Regression tree:
## tree(formula = unem_rate ~ ., data = train)
## Variables actually used in tree construction:
##  [1] "diff"                          "Tax.Burden"
##  [3] "Government.Spending"           "Monetary.Freedom"
##  [5] "NY.GDP.MKTP.KD.ZG"             "NY.GDP.MKTP.CD"
##  [7] "SL.TLF.CACT.ZS"                "Region.Europe...Central.Asia"
##  [9] "SL.TLF.TOTL.IN"                "EN.POP.DNST"
## [11] "Region.Sub.Saharan.Africa"     "Trade.Freedom"
## Number of terminal nodes:  18
## Residual mean deviance:  0.2174607 = 126.7796 / 583
## Distribution of residuals:
##        Min.     1st Qu.      Median        Mean     3rd Qu.        Max.
## -1.61566600 -0.28920100 -0.03538795  0.00000000  0.22159670  1.82668000
```

```
# cross validation
mytree2 = cv.tree(mytree,FUN = prune.tree)
mytree2
```

```
## $size
##  [1] 18 17 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 258.4741975 257.8893538 256.9273980 261.4490510 261.8591143
##  [6] 267.5244870 266.7343487 299.4451098 304.1179272 304.1179272
## [11] 307.5607616 323.8532054 371.0250530 459.2875872 503.5383139
## [16] 504.7134265 618.1513439
##
## $k
##  [1]          -Inf   6.865645425   7.101890680   8.624445121   8.988239032
##  [6]   9.932642502  10.340762241  15.387856693  15.500809804  15.771138241
## [11]  20.173790236  29.220500063  37.139985934  47.693142637  56.159344586
## [16]  61.989426878 127.914831960
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(mytree2$size,mytree2$dev, xlab="size", ylab="dev",type = 'b')
```




```
cat("The best size is 9, as it has a drop in dev between 8 and 9.\n")
```

```
## The best size is 9, as it has a drop in dev between 8 and 9.
```

```
mytree3 = prune.tree(mytree,best= 9)
names(mytree3)
```

```
## [1] "frame"   "where"   "terms"   "call"    "y"       "weights"
```

```
plot(mytree3)
text(mytree3,pretty = 2)
```

Random Forests

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## 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
```

```
#df <- rfImpute(unem_rate ~ ., df,iter=5)
#df=df[sample(nrow(df)),]
set.seed(101)
#df<-df[,-2]

#Create 10 equally size folds
folds <- cut(seq(1,nrow(df)),breaks=10,labels=FALSE)
#Perform 10 fold cross validation
rmse_df = data.frame()
m = c(18,12,9,6)
n = c(50,100,150,200)
for(j in 1:3){
  cat("j =",j,'\n')
  for(k in 1:4){
    cat("k =",k,'\n')
    for(i in 1:10){
      testIndexes <- which(folds==i,arr.ind=TRUE)
      X.test <- df[testIndexes, -1]
      X.train <- df[-testIndexes, -1]

      y.test <- df[testIndexes, 1]
      y.train <- df[-testIndexes, 1]

      rf=randomForest(x = X.train,y = y.train,xtest = X.test, ytest = y.test,mtry=m[j],ntree = n[k])
      rmse_df[i,'rmse']=sqrt(mean(rf$test$mse))
      #print(cor(rf.concrete$test$predicted,y.test))
    }
    cat("The estimated rmse based on 10-fold cv is:",mean(rmse_df$rmse),'\n')
  }
}
```

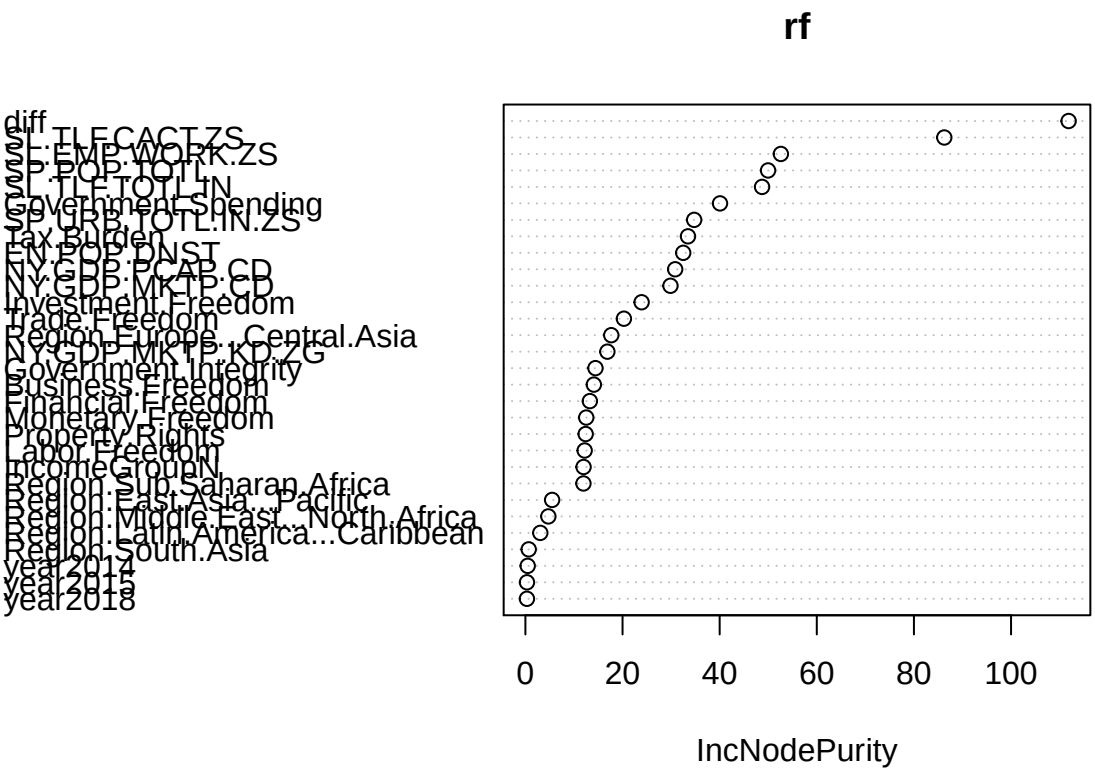
```
## j = 1
## k = 1
## The estimated rmse based on 10-fold cv is: 0.7409260438
## k = 2
## The estimated rmse based on 10-fold cv is: 0.7220030625
## k = 3
```

```
## The estimated rmse based on 10-fold cv is: 0.7429618904
## k = 4
## The estimated rmse based on 10-fold cv is: 0.7176261796
## j = 2
## k = 1
## The estimated rmse based on 10-fold cv is: 0.7529769558
## k = 2
## The estimated rmse based on 10-fold cv is: 0.7297307749
## k = 3
## The estimated rmse based on 10-fold cv is: 0.7192096454
## k = 4
## The estimated rmse based on 10-fold cv is: 0.7276849711
## j = 3
## k = 1
## The estimated rmse based on 10-fold cv is: 0.7239492076
## k = 2
## The estimated rmse based on 10-fold cv is: 0.7220323687
## k = 3
## The estimated rmse based on 10-fold cv is: 0.7199788717
## k = 4
## The estimated rmse based on 10-fold cv is: 0.7094852417
```

```
print("The model with ntree = 200 and m = 6 has the best rmse, 10-fold rmse = 0.71.")
```

```
## [1] "The model with ntree = 200 and m = 6 has the best rmse, 10-fold rmse = 0.71."
```

```
# feature importance
varImpPlot(rf)
```



3.3 Lasso selection + GBM

Data preparation

```
# Loaging the library
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

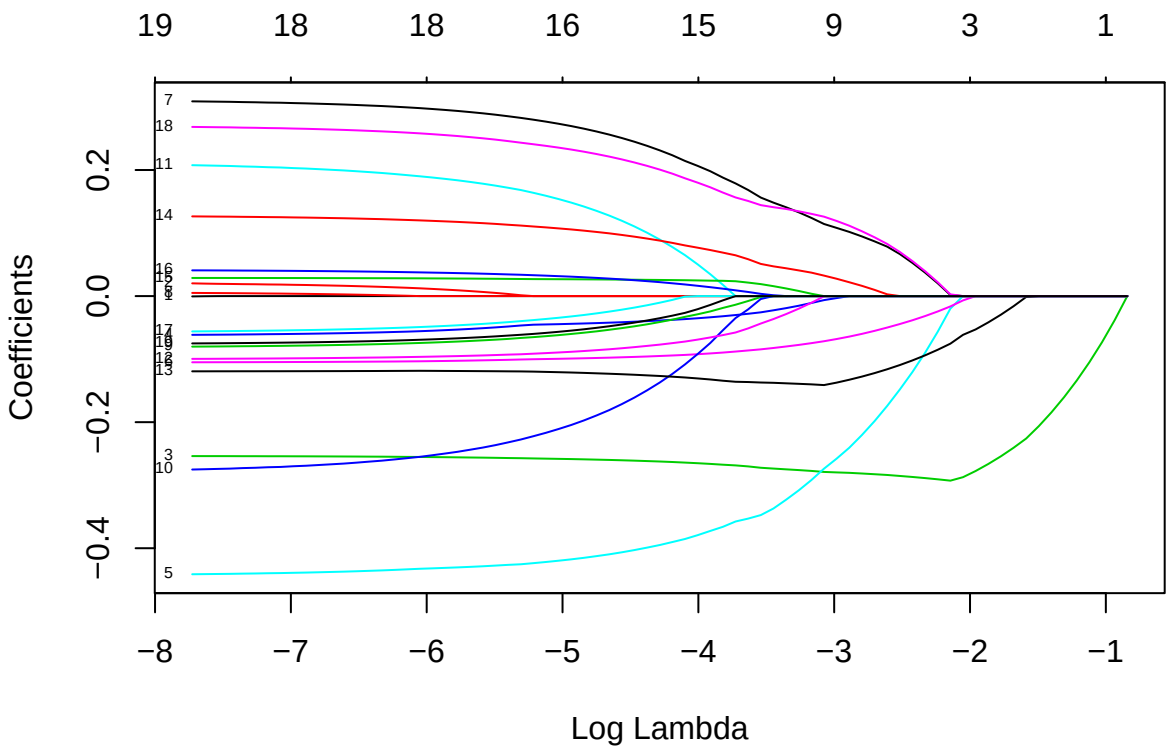
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 3.0-2
```

```
unemployment_rate <- nonNA


#ncol(unemployment_rate)

# Clean data and delete categorical variables
unemployment_rate = subset(unemployment_rate, select = -c(country.name, year, country.code,
                                                          SL.UEM.TOTL.FE.ZS, SL.UEM.TOTL.MA.ZS,
                                                          year2014, year2015, year2016,
                                                          year2017, year2018, Region, IncomeGroup
                                                          ))
# Scale dataset
unemployment_rate = scale(unemployment_rate)

# Save dataset as dataframe
unemployment_rate = as.data.frame(unemployment_rate)

# Get summary of updated data
#summary(unemployment_rate)
```

Use LASSO to select variables

```
# Prepare variables to run LASSO feature selection
x_vars <- model.matrix(SL.UEM.TOTL.ZS~., data = unemployment_rate)[,-1]

y_var <- unemployment_rate$SL.UEM.TOTL.ZS

fit.lasso=glmnet(x_vars,y_var)
plot(fit.lasso, xvar="lambda", label=TRUE)
```




```
lambda_seq <- 10^seq(2, -2, by = -.1)

# Splitting the data into test and train
set.seed(8470)
train = sample(1:nrow(x_vars), nrow(x_vars)/2)
test = (-train)
y_test = y_var[test]

cv_output <- cv.glmnet(x_vars[train,], y_var[train],
            alpha = 1, lambda = lambda_seq)

# identifying best lamda
best_lam <- cv_output$lambda.min

lasso_best <- glmnet(x_vars[train,], y_var[train], alpha = 1, lambda = best_lam)

# getting the list of important variables
coef(lasso_best) # Lists out coeffieicnts for each lambda
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                                  s0
## (Intercept)           0.041881423060
## SP.POP.TOTL          -0.006762877787
## SL.TLF.TOTL.IN        .
## SL.TLF.CACT.ZS       -0.304044924903
## NY.GDP.MKTP.CD       -0.036295447060
## NY.GDP.PCAP.CD       -0.459084181854
## NY.GDP.MKTP.KD.ZG    -0.078389439075
## SL.EMP.WORK.ZS        0.302578482741
## EN.POP.DNST           0.029670526658
```

```
## SP.URB.TOTL.IN.ZS    -0.102871589281
## Property.Rights      -0.137539846550
## Government.Integrity  0.212576771575
## Tax.Burden           -0.090671917591
## Government.Spending  -0.175721704451
## Business.Freedom      0.032259105320
## Labor.Freedom         0.002258497088
## Monetary.Freedom      0.041115942914
## Trade.Freedom        -0.037010758129
## Investment.Freedom    0.192791863948
## Financial.Freedom    -0.088672146758
```

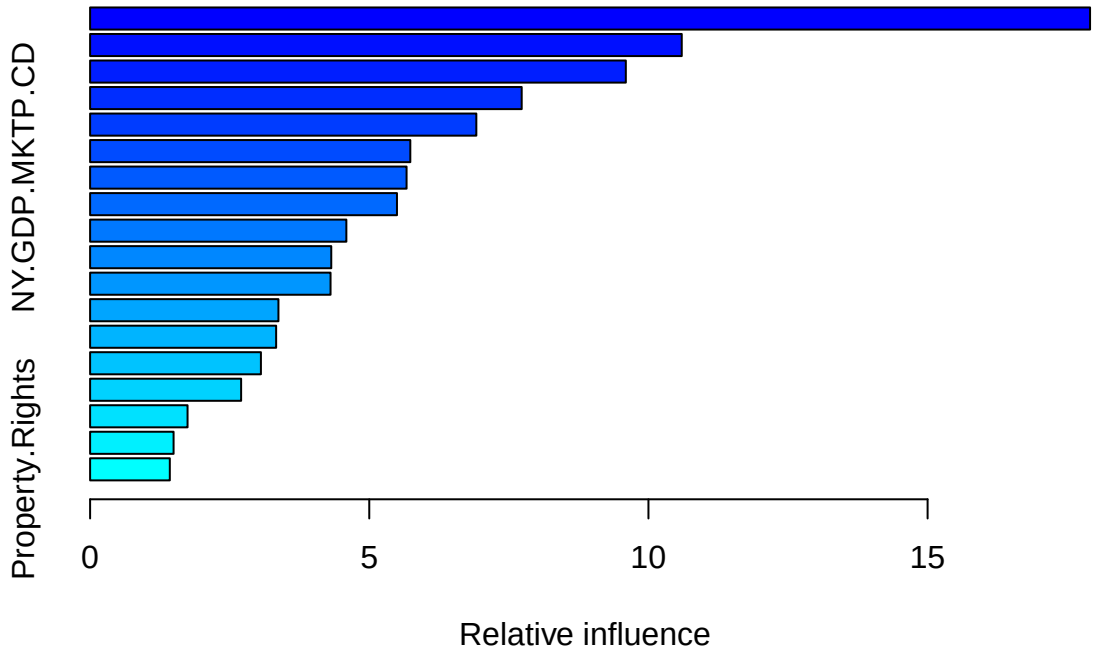
```
# The model indicates that the coefficients of SL.TLF.TOTL.IN (Labor force total) have been shriked to
```

Build the statistical learning model using Gradient Boosting Machine

```
# Load gbm library
library(gbm)
```

```
## Loaded gbm 2.1.5
```

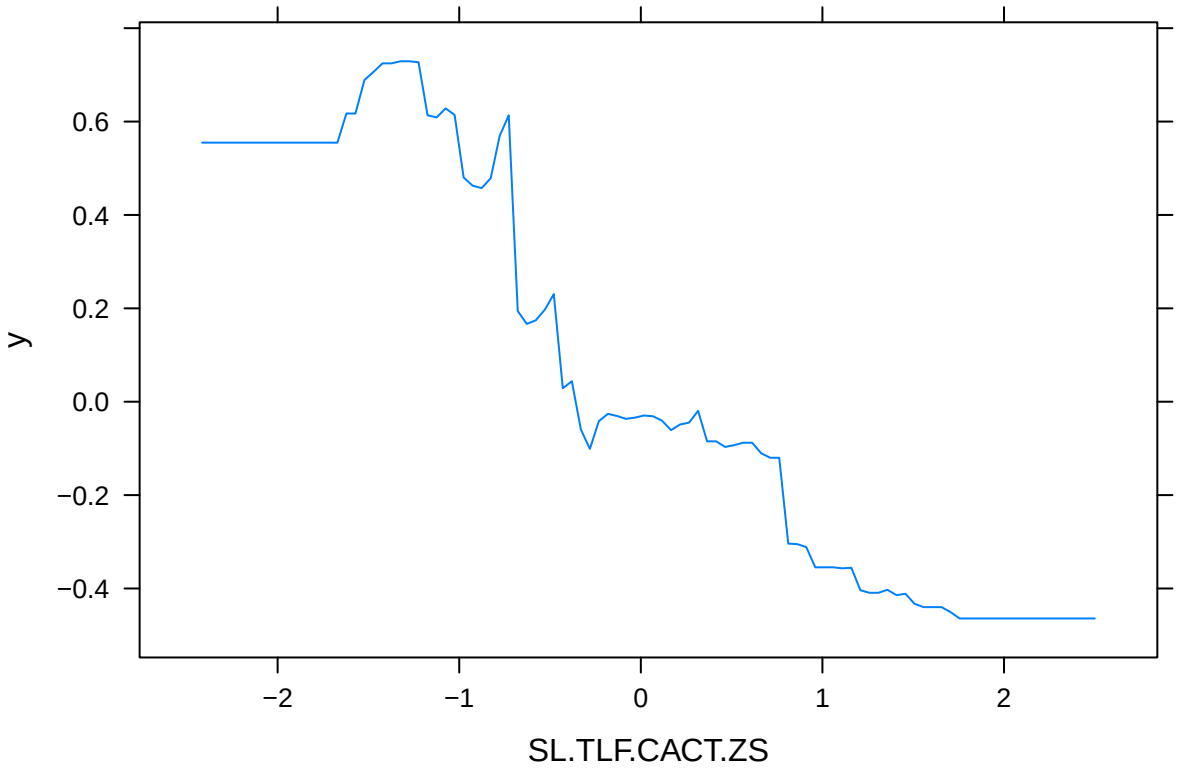
```
set.seed(101)
# Run Gradient boosting tree model with selected variables
boost.unemployment_rate <- gbm(SL.UEM.TOTL.ZS ~ .-SL.TLF.TOTL.IN, data = unemployment_rate[train,], dis

summary(boost.unemployment_rate)
```



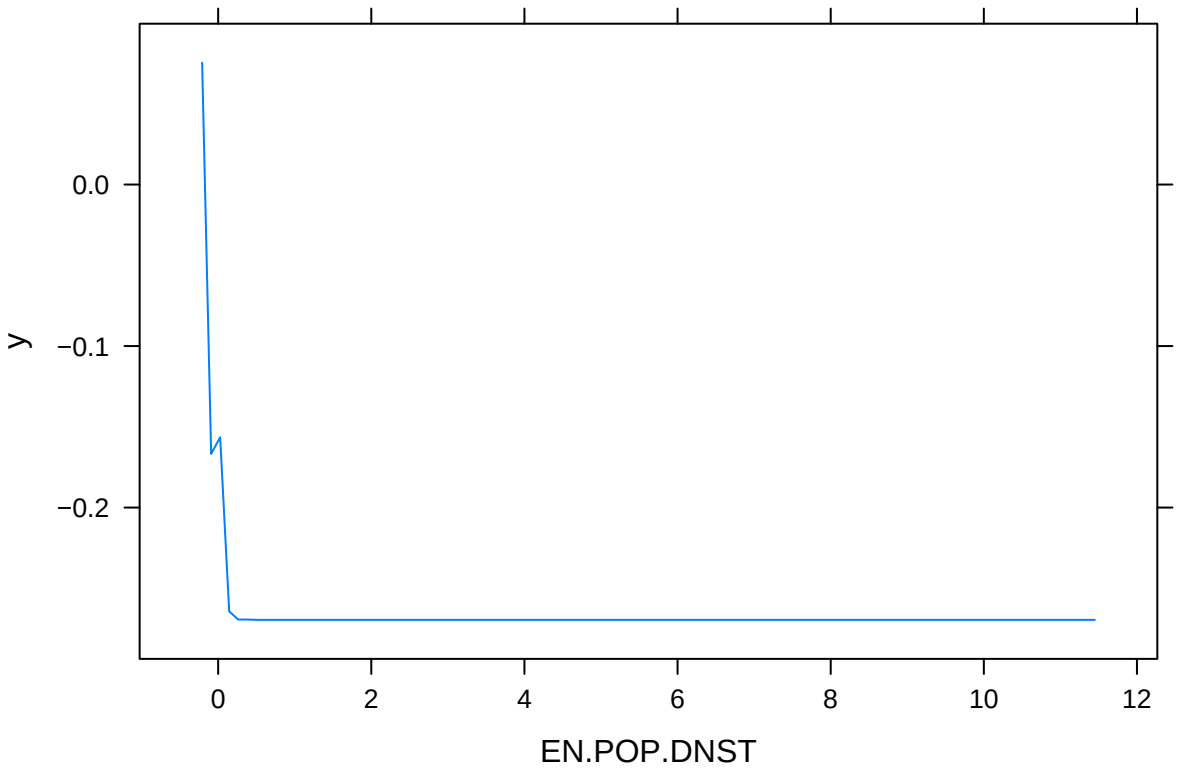

```
##                                      var      rel.inf
## SL.TLF.CACT.ZS             SL.TLF.CACT.ZS 17.909813199
## SL.EMP.WORK.ZS             SL.EMP.WORK.ZS 10.596623488
## EN.POP.DNST                   EN.POP.DNST  9.595224776
## SP.POP.TOTL                   SP.POP.TOTL  7.729773204
## Government.Integrity Government.Integrity  6.917544413
```

```
## NY.GDP.MKTP.KD.ZG       NY.GDP.MKTP.KD.ZG  5.734713314
## NY.GDP.MKTP.CD             NY.GDP.MKTP.CD  5.668188030
## Government.Spending   Government.Spending  5.495894454
## SP.URB.TOTL.IN.ZS       SP.URB.TOTL.IN.ZS  4.589401623
## Business.Freedom         Business.Freedom  4.319256720
## NY.GDP.PCAP.CD             NY.GDP.PCAP.CD  4.306416569
## Labor.Freedom               Labor.Freedom  3.372836665
## Tax.Burden                     Tax.Burden  3.332203335
## Investment.Freedom     Investment.Freedom  3.059357182
## Financial.Freedom       Financial.Freedom  2.704730576
## Monetary.Freedom         Monetary.Freedom  1.745272154
## Trade.Freedom               Trade.Freedom  1.494982105
## Property.Rights           Property.Rights  1.427768193
```

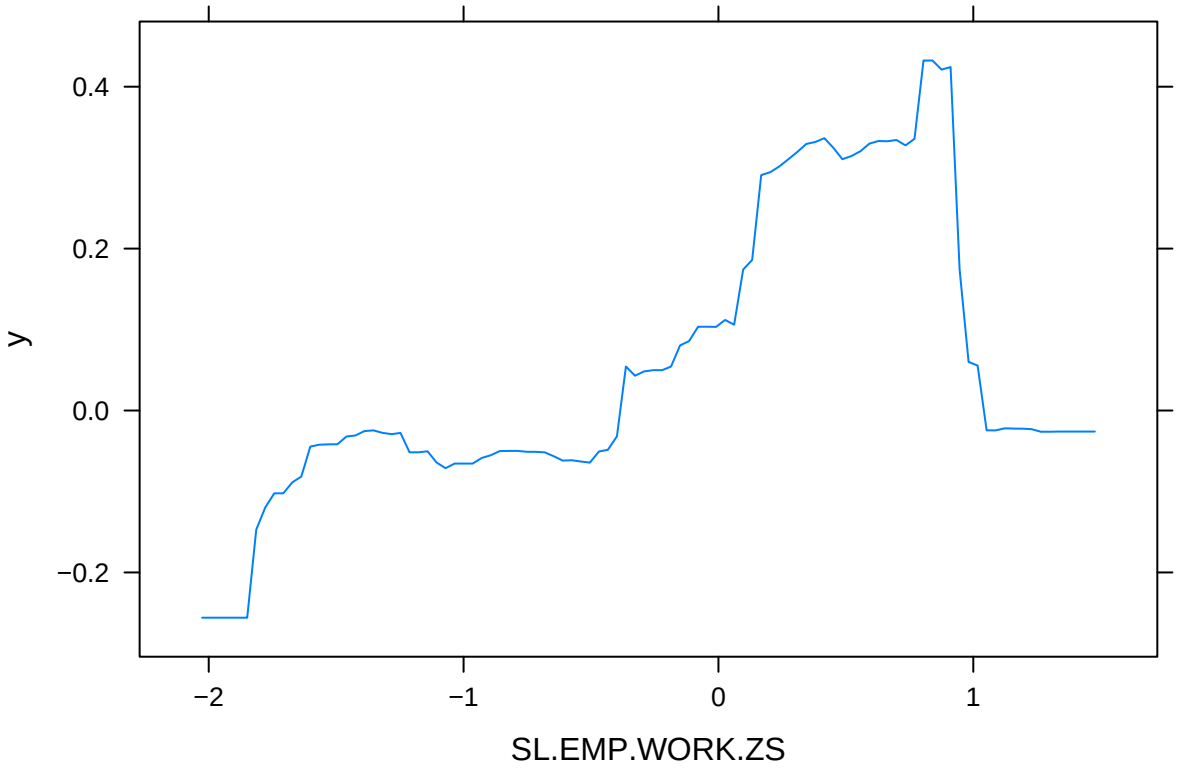
```r
par(mfrow=c(1,2))

# Plot top 3 important variables
plot(boost.unemployment_rate,i="SL.TLF.CACT.ZS")
```




```r
plot(boost.unemployment_rate,i="EN.POP.DNST")
```




```
plot(boost.unemployment_rate,i="SL.EMP.WORK.ZS")
```




```
# Make a prediction on the test set.
n.tree = seq(from=100,to=5000, by=100)
yhat.boost=predict(boost.unemployment_rate,newdata=unemployment_rate[-train,],n.trees=n.tree)
```

```r
dim(yhat.boost)
```

```
## [1] 401  50
```

```r
berr= with(unemployment_rate[-train,],apply((yhat.boost-unemployment_rate[test, ]$SL.UEM.TOTL.ZS)^2,2,m

# Make a plot
plot(n.tree,berr, pch=19, ylab="Mean Squared Error", xlab ="# Trees", main ="Boosting Test Error" )
abline(h=min(berr), col="red")

# Get RSME of the model
print("RMSE:")
```

```
## [1] "RMSE:"
```

```r
sqrt(mean((yhat.boost-unemployment_rate[test, ]$SL.UEM.TOTL.ZS)^2))
```

```
## [1] 0.4259104833
```