**Introduction**

K-means was used to cluster a number of variables from a combined dataset of the Gapminder and Transparency international dataset with the hope of identifying clusters ranging from free open (democratically free as well as economically free), low corruption societies to anocratic (states in some form of failure) to autocratic (states that are not free or open to business).

The variables used in the study were:

1.      ‘incomeperperson’ :average income per person.

2.      ‘armedforcesrate’ : armed forces rate as a % of population.

3.      'femaleemployrate’ :female employee rate.

4.      'internetuserate’: internet use rate.

5.      CPI2015’ from the transparency international dataset, the transparency international score for 2015.

6. 'PRS International Country Risk Guide’ score. Built by ICRG.

7. 'World Economic Forum EOS (Executive Opinion Survey)’ .

8. Polity score. How free a country is?

9. Life expectancy.

10. Level of alcohol consumption

11. Employrate.

**Work Done**

The work done consisted of a number of steps, these included:

1. Sourcing the data.
2. Merging the two datasets (transparency and Gapminder)
3. Standardize the variables. The variables were standardized with a mean of 0 and a standard deviation of 1.
4. Removal of observations with missing values
5. As the dataset was a relatively small number of observations, there was no need to split into training and test data sets. This was not done and the clustering analysis was ran on the dataset as is.
6. Cluster over a range of k (number of cluster values) to decide on optimum K value.
7. Plot a scree (elbow) plot of the average distance of points (within a cluster) from centroid of cluster for the different K values against K. From the plot ascertain the optimum number of clusters (in our case it appeared to be 3 or 5) by ascertaining where the elbow in our curve occurred (i.e after this point no further improvement in minimizing average distance of points from centroid of cluster was seen).
8. Validate whether our chosen K -means clustering was affective at partitioning our data, this was done in a number of ways:
   * + 1. By creating a data visualization of a scatterplot of the first two principal components (principal component analysis is used to reduce the number of dimensions while still holding most of the variance of the original dataset in the newly created dimensions) and colour or shade as the cluster one can get a sense of how effective the clustering was. If the clustering was effective the clusters should be fairly tight and well separated from each other. From this analysis 3 and 5 were found give the best clusters with k = 5 giving the optimal choice.
       2. By appending the cluster names to the original dataset we can see how effective the clustering has been at partitioning the dataset into meaningful clusters by examining the summary statistics of the different dimensions for the different clusters.
       3. Point 2 can be extended and made easier to understand by using a data-visualization to look at the distribution of values of different dimensions for the different clusters.
       4. Through the use of statistical inference tests such as ANOVA we can check if there is a statistically significant difference in particular dimension values through-out the different clusters.
9. Finally we create a bi-plot (an enhanced scatterplot) to try and extract meaning from our clustering exercise, what to the different clusters mean. A biplot is a form of exploratory data-visualization which is based upon a scatterplot of the first two principal components of a dataset. The observations are plotted as points while the original variables are represented as vectors. The colour of the observation represents what cluster an observation belongs to. Using these plots we cannot only see how the effectiveness of our clustering but what are the influences (from the original dimensions) that make up our cluster.

**k- Means clustering**

k-Means clustering is a partition type clustering technique used to produce a fixed number of clusters (k, the number of clusters). When generating the optimal value for K, the clustering is run a number of times for different values of K and based on a goodness of clustering metric (in our case average distance of points (within a cluster) from centroid of cluster).

K-means clustering consists of 3 steps, these are

Step one, the initial centroids of the clusters are selected, with the most basic method being to choose k observations from the dataset randomly. Upon completion of this preliminary step (to select 3 clustering seeds), k-means involves iterating over the 2 remaining steps.

The first step also allocates all observation to its closest (in our case we calculate this using Euclidean distance) centroid.

The second step involves the generation of new centroids by taking the mean value of all of the samples assigned to each previous centroid. The third step calculates the difference between the old and the new centroids are then computed and k-means reiterates these two previous steps over and over again till such a time as this value is less than a threshold value, i.e till such a time as the centroids do not move significantly and ca be considered to be stabilized.

**PCA (Principal Component Analysis)**

Dimension reduction techniques are a series of techniques which are designed to represent a dataset by transforming it to a lower number of dimensions but maintaining most of the information held within the original dataset. Principal Component Analysis (PCA) components are derived through the diagonalization of a covariance matrix (therefore it will only work with numerical data, not with categorical data). The new components are uncorrelated. The transformation of the old dataset into a new dimensional space is represented by the equation below, where y represents an individual observation, x is the original observation and W is an orthogonal matrix created from the covariance matrix.

y =WT

The orthogonal matrix W is the d (dimensional samples) x k dimensional matrix created by choosing k eigenvectors sorted in decreasing order of eigenvalues by the largest eigenvalues. The eigenvectors and eigenvalues are calculated from the covariance matrix.

W = d X k

These components are then used to represent the data in a smaller, easier to understand dimension space.

**Selecting K number of clusters using the elbow method**

In my example I clustered over a range of k (number of cluster values) to decide on optimum K value. The range of K was 1-10.

A scree (elbow) plot was then plotted of the average distance of points (within a cluster) from centroid of cluster for the different K values against K. From the plot ascertain the optimum number of clusters (in our case it appeared to be 3 or 5) by ascertaining where the elbow in our curve occurred (i.e after this point no further improvement in minimizing average distance of points from centroid of cluster was seen).

The scree plot is shown below we set x and y axis at 0,0, and can see a number of candidate elbow points these are 2,3 and 5.

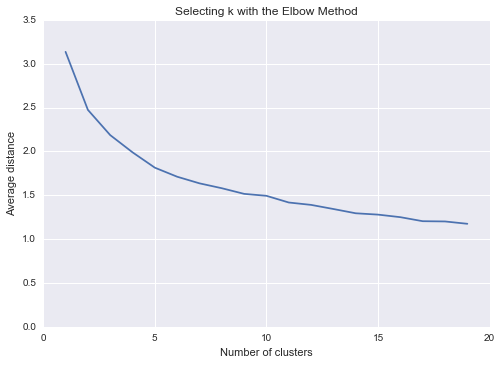


Figure. Scree plot to find optimal number of K Clusters.

**Validating our clustering K values through the use of PCA and data visualization**

PCA is ran on our original dataset. From the PCA scree plot, it shows that the first two components hold more than 80% of the variance in the dataset.

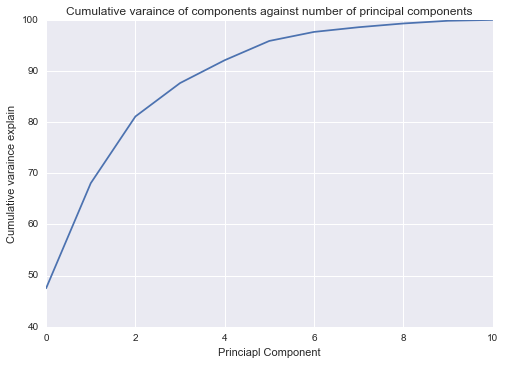


Figure. PCA scree plot.

By creating a data visualization of a scatterplot of the first two principal components (principal component analysis is used to reduce the number of dimensions while still holding most of the variance of the original dataset in the newly created dimensions) and colour or shade as the cluster one can get a sense of how effective the clustering was. If the clustering was effective the clusters should be fairly tight and well separated from each other. From this analysis 3 and 5 were found give the best clusters with k = 5 giving the optimal choice. These are shown below for:

1. k=2
2. k=3
3. k=5

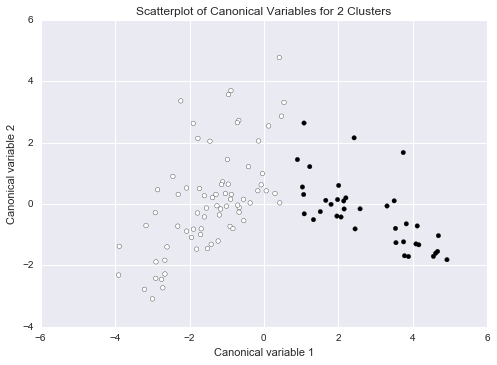


Figure. Scatterplot of Canonical Variables (PCA2,PCA1) for 2 clusters.

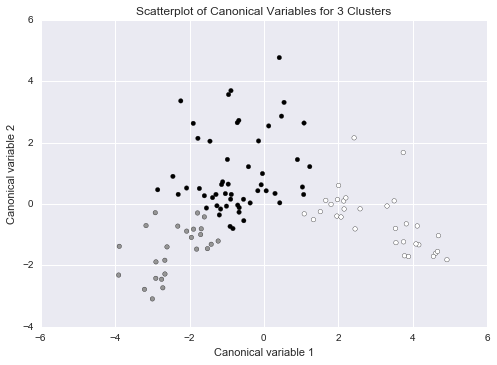


Figure. Scatterplot of Canonical Variables (PCA2,PCA1) for 3 clusters.

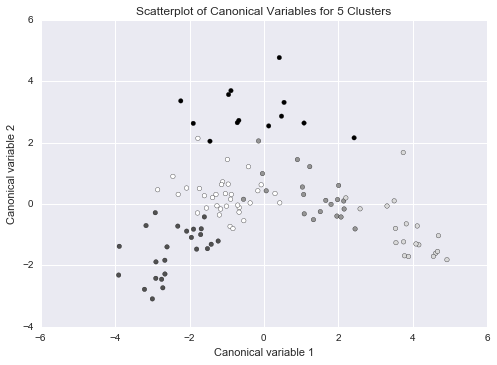


Figure. Scatterplot of Canonical Variables (PCA2,PCA1) for 5 clusters.

From the analysis of the scatterplots we can see that k = 3 or k = 5 gives the optimal splitting of the data. This is due to the observations within the clusters being tightly grouped and well separated from each other.

With k=2 while the two groups are well separated from each other they are not tightly grouped together and this would represent a sub optimal clustering solution.

**Examining the summary statistics of the different dimensions for the different clusters**

The means for the data dimensions for the different clustering k values (k=3,5) are shown below:

K=3

The mean values are shown below:

*'''*

*print(clustergrp)*

*index incomeperperson alcconsumption armedforcesrate \*

*cluster*

*0 108.781250 1.235069 0.615147 0.038096*

*1 100.423077 -0.407125 -0.154069 0.255742*

*2 108.307692 -0.705835 -0.448967 -0.558371*

*femaleemployrate internetuserate lifeexpectancy employrate \*

*cluster*

*0 0.132927 1.260777 0.909442 -0.020729*

*1 -0.698979 -0.291682 -0.000986 -0.610994*

*2 1.234355 -0.968362 -1.117341 1.247499*

*CPI2015 World Economic Forum EOS \*

*cluster*

*0 1.349325 1.234625*

*1 -0.449433 -0.382780*

*2 -0.761841 -0.753979*

*PRS International Country Risk Guide polityscore*

*cluster*

*0 1.311189 0.529000*

*1 -0.478346 -0.058928*

*2 -0.657079 -0.533222*

*'''*

That K=3 does a pretty good job at partitioning the dataset with the mean values of the different dimensions very different for all 3 clusters.

From our analysis of the mean values of the different dimensions that

Cluster 0, Western democracies is categorized as high polity score (democratic), high CPI 2015 (low levels of corruption), High EOS and PRS scores (low economic risk and good economic investment opportunity). Relatively low armed forces rates, average employee rates and high average incomes. These countries are mainly.

Cluster 1 is categorized as having high armed forces rates, average polity scores (broken democracies to failed states), PRS and EPS scores (somewhat risky investments with moderate to low economic outlook), low employee rates and high armed forces rates.

Cluster 2 is categorized as having low armed forces rates, low polity scores (autocratic states), low PRS and EPS scores (risky investments low economic outlook), high employee rates and low average incomes.

However there seems to be somewhat of a problem when we examine the distributions of the polityscore dimensions with the clusters showing overlapping regions of polity scores between the different clusters

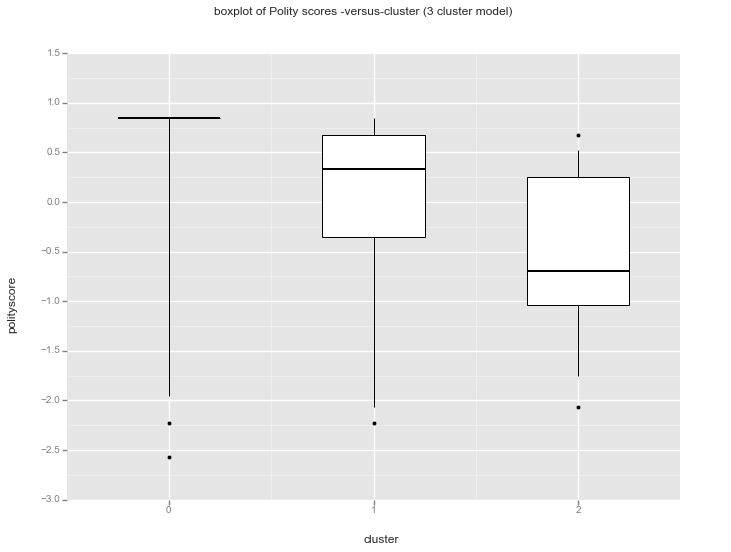


Figure. Boxplot of polity scores for the different clusters (k=3).

K=5 again does a pretty good job at partitioning the dataset with the mean values of the different dimensions very different for all 5 clusters. As we have more clusters we get a better splitting of our data.

*'''*

*Clustering variable means by cluster*

*index incomeperperson alcconsumption armedforcesrate \*

*cluster*

*0 106.210526 1.925011 0.360945 -0.191434*

*1 111.153846 -0.304801 -1.218275 1.514747*

*2 108.291667 -0.727085 -0.492013 -0.600894*

*3 102.484848 -0.503862 -0.074974 -0.272969*

*4 98.809524 0.069748 1.107717 0.351189*

*femaleemployrate internetuserate lifeexpectancy employrate \*

*cluster*

*0 0.255068 1.530205 1.029620 0.226588*

*1 -1.745974 -0.216228 0.192075 -1.116786*

*2 1.287686 -0.991900 -1.184508 1.312131*

*3 -0.244027 -0.502937 -0.209858 -0.197278*

*4 -0.238104 0.673314 0.633036 -0.703234*

*CPI2015 World Economic Forum EOS \*

*cluster*

*0 1.736917 1.701811*

*1 -0.439791 -0.032307*

*2 -0.777429 -0.777062*

*3 -0.532239 -0.524499*

*4 0.425622 0.192549*

*PRS International Country Risk Guide polityscore*

*cluster*

*0 1.718800 0.372376*

*1 -0.447996 -1.310539*

*2 -0.655673 -0.591411*

*3 -0.558362 0.279227*

*4 0.348993 0.711488*

*'''*

Based on the data observed we can see that we get a much better split and there is much less generalization as is the case when k-3.

Cluster 0 tier 1 western democracies, politically and economically free, low levels of corruption, high income per person and average employee rates.

Cluster 4 tier 2 western democracies, for the most part politically (politically free) and economically free (though have lower Transparency EOS and PRS scores), fairly low levels of corruption, good income per person and below average employee rates and average income per person.

Cluster 2 failed states to autocratic governments, low armed forces rates, high employment rates, low income per person and not politically (polity score) or economically free (low EOS and PRS scores).

Cluster 3 emerging democracies, emerging democracies with high levels of corruption, low levels of economic freedom (low EOS and PRS scores), very low income per person and slightly below average employ rates.

Cluster 1 Strongly autocratic governments, below average PRS score, average EOS score, high armed forces rates, corrupt (low CPI 2015 transparency scores), very low employment rates, very high armed forces rates, low alcohol consumption rates, low female employment rates and below average income per person.

Again we look at the boxplot of our distributions of polityscore. However this time there does not seem to be a problem when we examine the distributions of the polityscore dimensions with the clusters not showing overlapping regions of polity scores between the different clusters

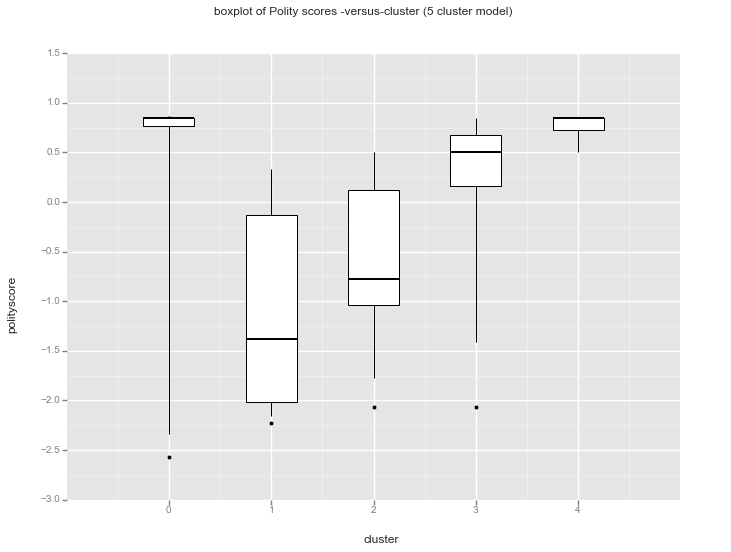


Figure. Boxplot of polity scores for the different clusters (k=5).

**Validating clusters in the data by examining cluster differences in Polityscore using ANOVA for k-means clustering (k=3,5)**

**K=3**

The analysis of variance (and standard deviation) within groups shows that equality of homogeneity assumption maybe violated. Running the ANOVA analysis using the general linear model function we can see that there is evidence obtained that there are differences amongst the different groups. The ANOVA analysis gave an F-statistic of 6.730 and a p value (Prob (F-statistic):  0.00206), which is less than our critical value of 0.05. This would indicate that there is a difference in means of the different groups. The model output is shown below the following code:

**'''**

**print (gpamod.summary())**

**OLS Regression Results**

**==============================================================================**

**Dep. Variable: polityscore R-squared: 0.154**

**Model: OLS Adj. R-squared: 0.131**

**Method: Least Squares F-statistic: 6.730**

**Date: Wed, 22 Jun 2016 Prob (F-statistic): 0.00206**

**Time: 22:37:15 Log-Likelihood: -104.11**

**No. Observations: 77 AIC: 214.2**

**Df Residuals: 74 BIC: 221.2**

**Df Model: 2**

**Covariance Type: nonrobust**

**===================================================================================**

**coef std err t P>|t| [95.0% Conf. Int.]**

**-----------------------------------------------------------------------------------**

**Intercept 0.5299 0.199 2.664 0.009 0.134 0.926**

**C(cluster)[T.1] -0.5851 0.256 -2.285 0.025 -1.095 -0.075**

**C(cluster)[T.2] -1.0770 0.296 -3.641 0.001 -1.666 -0.488**

**==============================================================================**

**Omnibus: 24.509 Durbin-Watson: 1.793**

**Prob(Omnibus): 0.000 Jarque-Bera (JB): 33.820**

**Skew: -1.444 Prob(JB): 4.53e-08**

**Kurtosis: 4.483 Cond. No. 4.02**

**==============================================================================**

**Warnings:**

**[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.**

**'''**

To check the pairwise comparisons, a between the different groups Tukey’s honest significant difference test is run in combination with the ANOVA as a post hoc test to show the pairs of  groups that the means that are significantly different.

Between the pairs we can only find evidence of statistically significant difference in means between cluster 0 and cluster 2 when (k=3).

**#########**

**###**

**#**

**''''**

**Multiple Comparison of Means - Tukey HSD,FWER=0.05**

**=============================================**

**group1 group2 meandiff lower upper reject**

**---------------------------------------------**

**0 1 -0.5851 -1.1976 0.0274 False**

**0 2 -1.077 -1.7845 -0.3696 True**

**1 2 -0.4919 -1.1422 0.1583 False**

**---------------------------------------------**

**Tukey HSD**

**'''**

**K=5**

When K=5 again we find evidence that within groups shows that equality of homogeneity assumption maybe violated. Running the ANOVA analysis using the general linear model function we can see that there is evidence obtained that there are differences amongst the different groups. The ANOVA analysis gave an F-statistic of 11.22 and a p value (Prob (F-statistic):  3.95e-07), which is less than our critical value of 0.05. This would indicate that there is a difference in means of the different groups. The model output is shown below the following code:

''''

print (polityscoremod.summary())

OLS Regression Results

==============================================================================

Dep. Variable: polityscore R-squared: 0.384

Model: OLS Adj. R-squared: 0.350

Method: Least Squares F-statistic: 11.22

Date: Sat, 25 Jun 2016 Prob (F-statistic): 3.95e-07

Time: 02:41:52 Log-Likelihood: -91.891

No. Observations: 77 AIC: 193.8

Df Residuals: 72 BIC: 205.5

Df Model: 4

Covariance Type: nonrobust

=======================================================================================

coef std err t P>|t| [95.0% Conf. Int.]

---------------------------------------------------------------------------------------

Intercept 0.3820 0.213 1.793 0.077 -0.043 0.807

C(cluster\_str)[T.1] -1.4672 0.337 -4.355 0.000 -2.139 -0.796

C(cluster\_str)[T.2] -0.9972 0.289 -3.456 0.001 -1.572 -0.422

C(cluster\_str)[T.3] -0.1227 0.282 -0.435 0.665 -0.685 0.439

C(cluster\_str)[T.4] 0.3947 0.307 1.287 0.202 -0.217 1.006

==============================================================================

Omnibus: 29.390 Durbin-Watson: 1.957

Prob(Omnibus): 0.000 Jarque-Bera (JB): 54.126

Skew: -1.421 Prob(JB): 1.76e-12

Kurtosis: 5.965 Cond. No. 5.97

==============================================================================

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

''''

To check the pairwise comparisons, a between the different groups Tukey’s honest significant difference test is run in combination with the ANOVA as a post hoc test to show the pairs of  groups that the means that are significantly different.

Between the pairs we can find evidence of statistically significant difference in means between

Cluster 0 and cluster 1

cluster 0 and cluster 2

cluster 1 and cluster 3

cluster 1 and cluster 4

cluster 2 and cluster 3

with

cluster 0 and cluster 4

cluster 0 and cluster 3 not showing any difference

when (k=5).

'''''

mc5 = multi.MultiComparison(sub2['polityscore'], sub2['cluster\_str'])

res5 = mc5.tukeyhsd()

print(res5.summary())

Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================

group1 group2 meandiff lower upper reject

---------------------------------------------

0 1 -1.4672 -2.41 -0.5245 True

0 2 -0.9972 -1.8045 -0.1898 True

0 3 -0.1227 -0.9115 0.666 False

0 4 0.3947 -0.4634 1.2529 False

1 2 0.47 -0.4408 1.3809 False

1 3 1.3445 0.4501 2.2389 True

1 4 1.862 0.9058 2.8181 True

2 3 0.8744 0.1242 1.6247 True

2 4 1.3919 0.569 2.2149 True

3 4 0.5175 -0.2872 1.3222 False

---------------------------------------------

'''''

**Bi-plot analysis using PCA of k=3 and k=5 clustering**

Bi-plots are improved scatterplots that use both points and vectors to embody the structure of our data. When used with principal component analysis the axes are the first and second principal components. Points are used to show the different observations along the principal components and the vectors are used to show the coefficients of the dimensions that make up the principal components.

Points that are near each other represent observations that have similar score for the principal components displayed in the plot. A vector is fixed along the course which is most similar to the variable characterised by the vector. This is the course/direction which has the highest squared multiple correlation with the principal components. The course of the vector is proportional to the squared multiple correlation between the fitted values for the variable and the actual variable.

The fitted values for a variable are due to the extending of the points in the space orthogonally onto the variable's vector. The observations whose points extend the farthest in the direction in which the vector points, are the observations that have the highest values of what the variable pertains to. Those points that extend in the opposite direction have the lowest, while those extending in the middle have an average value when compared to the rest.

Therefore, the vectors that point along a similar course or trajectory equate to variables that have similar response outlines, and they can be understood to have comparable connotation or meaning in the context set by the data.

To show the clusters cluster is represented by hue (colour of the dots).

**Biplot when K=3**

While some broad and general meaning by using k=3 we can see from the biplots quite clearly that the clusters are too broad and countries with very different characteristics are placed in the same clusters. For instance Greece and Italy are placed in the same cluster as Sri Lanka and Namibia (when k=3)! While on some variables like armed forces rate they are similar Greece, Greece and Italy have much higher political economic and economic freedoms (PRI and EOS scores) and less corruption (CPI 2015). This would give further evidence that k=5 gives more appropriate clusters. However Cluster 0 would seem to be by and large well formed representing western democracies quite well (high levels of economic and political freedoms,)

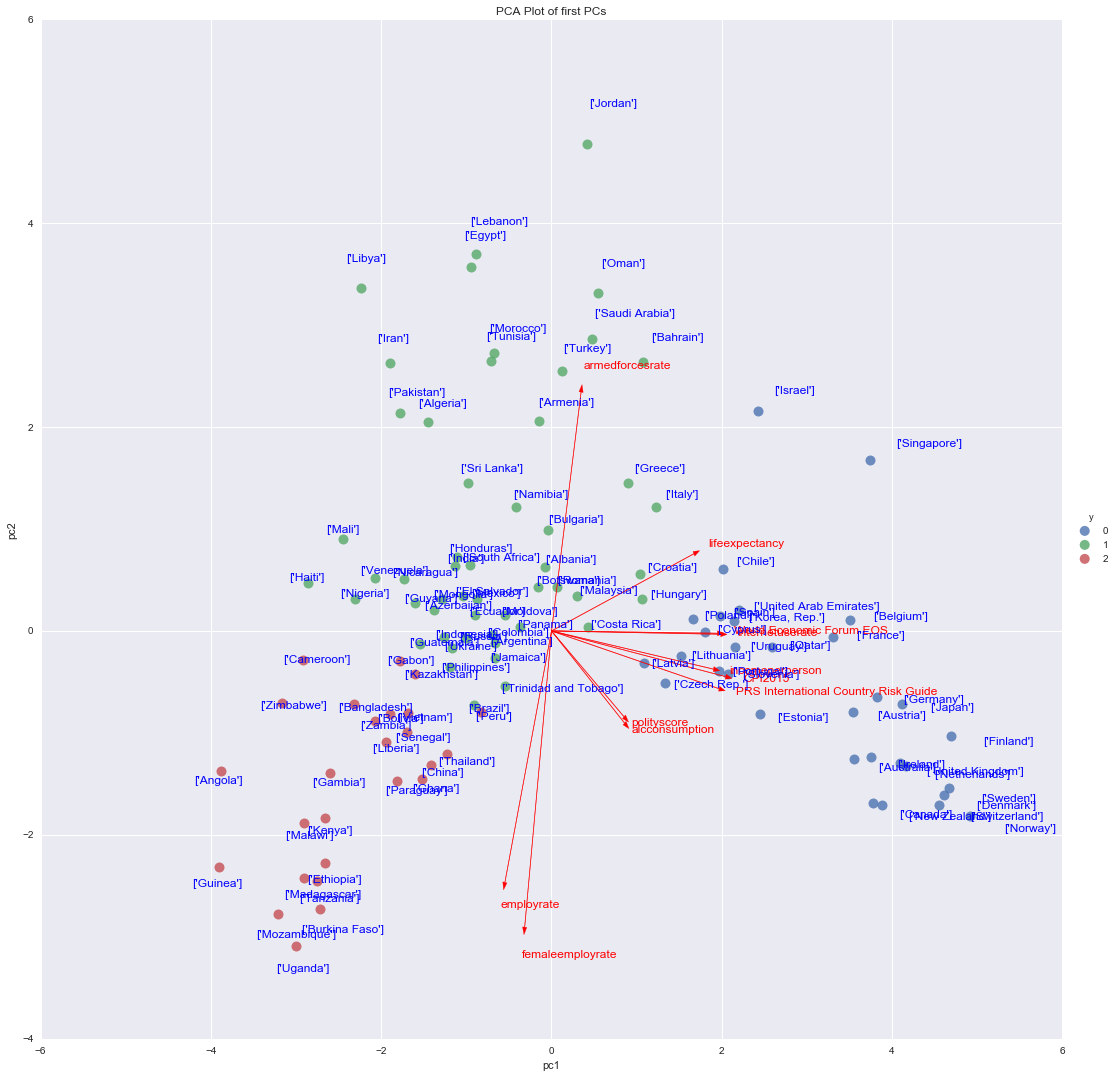


Figure. Biplot when k=3

**Biplot when k=5**

When we choose k=5, we can see that the clusters are much more well defined and intuitively make more sense. We can see now Sri Lanka and Namibia are placed in separate clusters to Greece and Italy and these clusters are more appropriate. Greece and Italy are placed in a cluster with other tier two Western democracies, Chile, Croatia, Poland, Israel , Lativa and Lithuania. Cluster 0 is again categorized as tier 1 Western democracies with political and economic freedoms and low levels of corruptions. 2 undemocratic states Qatar and UAE are included in this cluster due to their high levels of business freedoms (EOS and PRI) and low levels of corruption (CPI 2015).



Figure. Biplot when K=5

The other thing that should be noted is some of relationships between the vectors that point along a similar course or trajectory which indicate the variables that have similar response outlines. These would be alcohol consumption and polityscore and employrate and female employrate.

**Summary**

K means clustering with k=5 proved to be usefull tool for identifying not only democracies but also characteristics of a democracy.

**Code**

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jun 03 12:27:51 2016

@author: Peter

"""

import os

import pandas

import numpy

import sklearn

import matplotlib

import matplotlib.pyplot as plt

import sys; print(sys.path)

from seaborn import \*

import seaborn as sns

import ggplot

from ggplot import \*

import scipy

from pandas import Series, DataFrame

import pandas as pd

import numpy as np

import os

import matplotlib.pylab as plt

from sklearn.cross\_validation import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report

import sklearn.metrics

# Feature Importance

from sklearn.ensemble import ExtraTreesClassifier

import pydot

import graphviz

apath='C:\Users\Peter\Desktop\Gapminder'

print(apath)

os.chdir('C:\Users\Peter\Desktop\Gapminder')

##check the directory has changed

os.getcwd()

##read in the file

data = pandas.read\_csv('gapminder.csv', low\_memory=False)

##lets convert the data to numeric

data['incomeperperson'] = data['incomeperperson'].convert\_objects(convert\_numeric=True)

data['alcconsumption'] = data['alcconsumption'].convert\_objects(convert\_numeric=True)

data['armedforcesrate'] = data['armedforcesrate'].convert\_objects(convert\_numeric=True)

data['breastcancerper100th'] = data['breastcancerper100th'].convert\_objects(convert\_numeric=True)

data['co2emissions'] = data['co2emissions'].convert\_objects(convert\_numeric=True)

data['femaleemployrate'] = data['femaleemployrate'].convert\_objects(convert\_numeric=True)

data['hivrate'] = data['hivrate'].convert\_objects(convert\_numeric=True)

data['internetuserate'] = data['internetuserate'].convert\_objects(convert\_numeric=True)

data['lifeexpectancy'] = data['lifeexpectancy'].convert\_objects(convert\_numeric=True)

data['oilperperson'] = data['oilperperson'].convert\_objects(convert\_numeric=True)

data['polityscore'] = data['polityscore'].convert\_objects(convert\_numeric=True)

data['relectricperperson'] = data['relectricperperson'].convert\_objects(convert\_numeric=True)

data['suicideper100th'] = data['suicideper100th'].convert\_objects(convert\_numeric=True)

data['employrate'] = data['employrate'].convert\_objects(convert\_numeric=True)

data['urbanrate'] = data['urbanrate'].convert\_objects(convert\_numeric=True)

bins = [0, 1000, 5000, 10000, 20000,50000,200000]

group\_names = ['Very Low Income,0-1000', 'Low Income,1000-5000', 'Okay Income,5000-10000', 'Good Income,10000-20000','Great Income,20000-50000','50,000-200,000']

categories = pandas.cut(data['incomeperperson'], bins, labels=group\_names)

data['categories'] = pandas.cut(data['incomeperperson'], bins, labels=group\_names)

##data.dtypes chk

##now encode european countries

##

##Ok lets see what the best features are

##note to one self HIV rates missing froma lot of countries

datatransparency = pandas.read\_csv('CPI\_2015\_DATA.csv', low\_memory=False)

##w['female'] = w['female'].map({'female': 1, 'male': 0})

datatransparency.columns.values

data.columns.values

## Dont use map

## datatransparency['Country']= datatransparency['Country'].map(

## {"The United States Of America":"United States",

## "C“te dïIvoire":"Cote d'Ivoire",

## "Korea (South)":"Korea, Rep.",

## "Korea (North)":"Korea, Dem. Rep.",

## "Czech Republic":"Czech Rep.",

## "Democratic Republic of the Congo":"Congo, Dem. Rep.",

## "The FYR of Macedonia": "Macedonia, FYR",

## "Hong Kong":"Hong Kong, China"

## })

def country\_consistent (row):

if row['Country'] == "The United States Of America" :

return "United Sates"

elif row['Country'] == "C“te dïIvoire" :

return "Cote d'Ivoire"

elif row['Country'] == "Korea (South)" :

return "Korea, Rep."

elif row['Country'] == "Korea (North)" :

return "Korea, Dem. Rep."

elif row['Country'] == "Korea (South)" :

return "Korea, Rep."

elif row['Country'] == "Czech Republic" :

return "Czech Rep."

elif row['Country'] == "Democratic Republic of the Congo" :

return "Congo, Dem. Rep."

elif row['Country'] == "The FYR of Macedonia" :

return "Macedonia, FYR"

elif row['Country'] == "Hong Kong" :

return "Hong Kong, China"

else :

return row['Country']

datatransparency['Country'] = datatransparency.apply (lambda row: country\_consistent(row),axis=1)

##calculate the age of NATO countries

##data['Years\_In\_Nato'] = data.apply (lambda row: AGE\_YEARS (row),axis=1)

##

##ok after eyeballing in excel they all look ok

##merge the two datasets

datafullset=data.merge(datatransparency,left\_on='country',right\_on='Country',how='left')

datafullset.columns.values

datafullset.count

## 'country', 'incomeperperson', 'alcconsumption', 'armedforcesrate',

## 'breastcancerper100th', 'co2emissions', 'femaleemployrate',

## 'hivrate', 'internetuserate', 'lifeexpectancy', 'oilperperson',

## 'polityscore', 'relectricperperson', 'suicideper100th',

## 'employrate', 'urbanrate', 'categories', 'European', 'African',

## 'Asian', 'Mid\_East', 'North\_American', 'Carribean\_Central\_America',

## 'OPEC', 'Arab\_League', 'ASEAN\_ARF', 'South\_American',

## 'Is\_Nato\_Country', 'Year\_Joined\_Nato', 'Eu\_Member', 'Years\_In\_Nato',

## 'NATO\_EU\_MEMBERSHIP', 'polityscore\_cat', 'Rank', 'CPI2015',

## 'Country', 'Region', 'wbcode', 'World Bank CPIA',

## 'World Economic Forum EOS', 'Bertelsmann Foundation TI',

## 'African Dev Bank', 'IMD World Competitiveness Yearbook',

## 'Bertelsmann Foundation SGI', 'World Justice Project ROL',

## 'PRS International Country Risk Guide',

## 'Economist Intelligence Unit', 'IHS Global Insight',

## 'PERC Asia Risk Guide', 'Freedom House NIT', 'CPI2015(2)', 'Rank2',

## 'Number of Sources', 'Std Deviation of Sources', 'Standard Error',

## 'Minimum', 'Maximum', 'Lower CI', 'Upper CI', 'Country2'

## we are going to not include information

data2=datafullset[['country','incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]

data2.count

data\_clean2\_pre=data2.dropna()

data\_clean2\_pre.count

data\_clean2 = data\_clean2\_pre[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]

## drop all na values cant handle nulls

data\_clean2.count

from sklearn import preprocessing

## standardize the dataset

data\_clean2['incomeperperson']=preprocessing.scale(data\_clean2['incomeperperson'].astype('float64'))

data\_clean2['alcconsumption']=preprocessing.scale(data\_clean2['alcconsumption'].astype('float64'))

data\_clean2['armedforcesrate']=preprocessing.scale(data\_clean2['armedforcesrate'].astype('float64'))

data\_clean2['femaleemployrate']=preprocessing.scale(data\_clean2['femaleemployrate'].astype('float64'))

data\_clean2['internetuserate']=preprocessing.scale(data\_clean2['internetuserate'].astype('float64'))

data\_clean2['lifeexpectancy']=preprocessing.scale(data\_clean2['lifeexpectancy'].astype('float64'))

data\_clean2['employrate']=preprocessing.scale(data\_clean2['employrate'].astype('float64'))

data\_clean2['CPI2015']=preprocessing.scale(data\_clean2['CPI2015'].astype('float64'))

data\_clean2['World Economic Forum EOS']=preprocessing.scale(data\_clean2['World Economic Forum EOS'].astype('float64'))

data\_clean2['PRS International Country Risk Guide']=preprocessing.scale(data\_clean2['PRS International Country Risk Guide'].astype('float64'))

data\_clean2['polityscore']=preprocessing.scale(data\_clean2['polityscore'].astype('float64'))

##

###

###check the standardization worked

###

###

from sklearn import preprocessing

from sklearn.cluster import KMeans

##I will not split the dataset

# k-means cluster analysis for 1-9 clusters

from scipy.spatial.distance import cdist

clusters=range(1,20)

meandist=[]

##

#

##

for k in clusters:

model=KMeans(n\_clusters=k)

model.fit(data\_clean2)

clusassign=model.predict(data\_clean2)

meandist.append(sum(np.min(cdist(data\_clean2, model.cluster\_centers\_, 'euclidean'), axis=1))

/ data\_clean2.shape[0])

"""

Plot average distance from observations from the cluster centroid

to use the Elbow Method to identify number of clusters to choose

"""

plt.plot(clusters, meandist)

plt.xlim(0)

plt.ylim(0)

plt.xlabel('Number of clusters')

plt.ylabel('Average distance')

plt.title('Selecting k with the Elbow Method')

####From the scree plot lets see what we can see

#principal component analysis

## lets create a scree plot to see how much of the variance our

###

model2=KMeans(n\_clusters=2)

model2.fit(data\_clean2)

clusassign2=model2.predict(data\_clean2)

# plot clusters

from sklearn.decomposition import PCA

pca\_2 = PCA(2)

plot\_columns = pca\_2.fit\_transform(data\_clean2)

plt.scatter(x=plot\_columns[:,0], y=plot\_columns[:,1], c=model2.labels\_,)

plt.xlabel('Canonical variable 1')

plt.ylabel('Canonical variable 2')

plt.title('Scatterplot of Canonical Variables for 2 Clusters')

plt.show()

# Interpret 3 cluster solution

model3=KMeans(n\_clusters=3)

model3.fit(data\_clean2)

clusassign=model3.predict(data\_clean2)

# plot clusters

pca\_2 = PCA(2)

plot\_columns = pca\_2.fit\_transform(data\_clean2)

plt.scatter(x=plot\_columns[:,0], y=plot\_columns[:,1], c=model3.labels\_,)

plt.xlabel('Canonical variable 1')

plt.ylabel('Canonical variable 2')

plt.title('Scatterplot of Canonical Variables for 3 Clusters')

plt.show()

## Interpret 5 cluster solution

model5=KMeans(n\_clusters=5)

model5.fit(data\_clean2)

clusassign5=model2.predict(data\_clean2)

# plot clusters

pca\_2 = PCA(2)

plot\_columns = pca\_2.fit\_transform(data\_clean2)

plt.scatter(x=plot\_columns[:,0], y=plot\_columns[:,1], c=model5.labels\_,)

plt.xlabel('Canonical variable 1')

plt.ylabel('Canonical variable 2')

plt.title('Scatterplot of Canonical Variables for 5 Clusters')

plt.show()

##now try t-sme

X\_tsne3 = TSNE(learning\_rate=100).fit\_transform(data\_clean2)

plt.scatter(X\_tsne3[:, 0], X\_tsne3[:, 1], c=model3.labels\_)

plt.title("T-sne Plot categories Polity scores for 3 cluster")

X\_tsne5 = TSNE(learning\_rate=100).fit\_transform(data\_clean2)

plt.scatter(X\_tsne5[:, 0], X\_tsne5[:, 1], c=model5.labels\_)

plt.title("T-sne Plot categories Polity scores for 5 cluster")

##these are not effective at finding linear or non linear combinations of variables

# k=3 or k=5 gives the best split

# create a unique identifier variable from the index for the

# cluster training data to merge with the cluster assignment variable

data\_clean2.reset\_index(level=0, inplace=True)

# create a list that has the new index variable

cluslist=list(data\_clean2['index'])

# create a list of cluster assignments

labels=list(model3.labels\_)

# combine index variable list with cluster assignment list into a dictionary

newlist=dict(zip(cluslist, labels))

newlist

# convert newlist dictionary to a dataframe

newclus=DataFrame.from\_dict(newlist, orient='index')

newclus

# rename the cluster assignment column

newclus.columns = ['cluster']

newclus['cluster']

# now do the same for the cluster assignment variable

# create a unique identifier variable from the index for the

# cluster assignment dataframe

# to merge with cluster training data

newclus.reset\_index(level=0, inplace=True)

# merge the cluster assignment dataframe with the cluster training variable dataframe

# by the index variable

merged\_clust\_names=pd.merge(data\_clean2, newclus, on='index')

merged\_clust\_names.head(n=100)

merged\_clust\_names.columns.names

# cluster frequencies

###

#

#

###

data\_clean2\_pre.reset\_index(level=0, inplace=True)

countrylist=list(data\_clean2\_pre['country'])

countrylistindex=list(data\_clean2\_pre['index'])

newcountrylist=dict(zip(countrylistindex,countrylist))

newcountry=DataFrame.from\_dict(newcountrylist,orient='index')

newcountry.columns = ['country']

newcountry['country']

newcountry.reset\_index(level=0, inplace=True)

newcountry['country']

###

#

#

###

merged\_clust\_names\_country=pd.merge(merged\_clust\_names, newcountry, on='index')

merged\_clust\_names\_country[['country']]

##we can quickly see from applying

clustergrp = merged\_clust\_names\_country.groupby('cluster').mean()

print ("Clustering variable means by cluster")

print(clustergrp)

'''

print(clustergrp)

index incomeperperson alcconsumption armedforcesrate \

cluster

0 108.781250 1.235069 0.615147 0.038096

1 100.423077 -0.407125 -0.154069 0.255742

2 108.307692 -0.705835 -0.448967 -0.558371

femaleemployrate internetuserate lifeexpectancy employrate \

cluster

0 0.132927 1.260777 0.909442 -0.020729

1 -0.698979 -0.291682 -0.000986 -0.610994

2 1.234355 -0.968362 -1.117341 1.247499

CPI2015 World Economic Forum EOS \

cluster

0 1.349325 1.234625

1 -0.449433 -0.382780

2 -0.761841 -0.753979

PRS International Country Risk Guide polityscore

cluster

0 1.311189 0.529000

1 -0.478346 -0.058928

2 -0.657079 -0.533222

'''

# validate clusters in training data by examining cluster differences in GPA using ANOVA

# first have to merge GPA with clustering variables and cluster assignment data

import patsy

import pandas

import statsmodels

import statsmodels.formula.api as smf

import statsmodels.stats.multicomp as multi

##sub1.apply(lambda x: pd.to\_numeric('cluster', errors='ignore'))

polity\_score\_train, polity\_score\_test = train\_test\_split(merged\_clust\_names\_country, test\_size=.3, random\_state=123)

sub1 = polity\_score\_train[['cluster','polityscore']]

sub1.dtypes

sub1['cluster\_str'] = sub1['cluster'].astype(str)

polityscoremod= smf.ols(formula='polityscore ~ C(cluster)', data=sub1).fit()

print (polityscoremod.summary())

'''

print (gpamod.summary())

OLS Regression Results

==============================================================================

Dep. Variable: polityscore R-squared: 0.154

Model: OLS Adj. R-squared: 0.131

Method: Least Squares F-statistic: 6.730

Date: Wed, 22 Jun 2016 Prob (F-statistic): 0.00206

Time: 22:37:15 Log-Likelihood: -104.11

No. Observations: 77 AIC: 214.2

Df Residuals: 74 BIC: 221.2

Df Model: 2

Covariance Type: nonrobust

===================================================================================

coef std err t P>|t| [95.0% Conf. Int.]

-----------------------------------------------------------------------------------

Intercept 0.5299 0.199 2.664 0.009 0.134 0.926

C(cluster)[T.1] -0.5851 0.256 -2.285 0.025 -1.095 -0.075

C(cluster)[T.2] -1.0770 0.296 -3.641 0.001 -1.666 -0.488

==============================================================================

Omnibus: 24.509 Durbin-Watson: 1.793

Prob(Omnibus): 0.000 Jarque-Bera (JB): 33.820

Skew: -1.444 Prob(JB): 4.53e-08

Kurtosis: 4.483 Cond. No. 4.02

==============================================================================

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

'''

print ('means for PolityScore by cluster')

m1= sub1.groupby('cluster').mean()

print (m1)

print ('standard deviations for polityscore by cluster')

m2= sub1.groupby('cluster').std()

print (m2)

mc1 = multi.MultiComparison(sub1['polityscore'], sub1['cluster'])

res1 = mc1.tukeyhsd()

print(res1.summary())

####

#######

#######

####

#########

###

#

''''

Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================

group1 group2 meandiff lower upper reject

---------------------------------------------

0 1 -0.5851 -1.1976 0.0274 False

0 2 -1.077 -1.7845 -0.3696 True

1 2 -0.4919 -1.1422 0.1583 False

---------------------------------------------

Tukey HSD

'''

import matplotlib.pyplot as plt

from matplotlib.patches import Polygon

### lets visualize this

sub1pivot=sub1.copy

sub2=pd.DataFrame(sub1[['cluster\_str','polityscore']])

sub2['idx'] = sub2.groupby('cluster\_str').cumcount()

sub2.reset\_index()

sub2.dtypes

pivoted = sub2.pivot(columns='cluster\_str', values='polityscore')

pivoted.columns.names

pivoted.reset\_index()

pivoted[['0','1','2']]

pivoted.dtypes

pivotedplt=pivoted[['0','1','2']].reset\_index()

ggplot(sub1, aes(x='cluster', y='polityscore')) + geom\_boxplot() +ggtitle("boxplot of Polity scores -versus-cluster (3 cluster model)")

###

#

###

'''

Okay there seems to be a problem with this we are only seeing a statistically significant at the p = 0.05 level

We can even see from the boxplot there is an overlap of the plotly scores.

'''

##From the scree plot lets see what we can see

pca = PCA(n\_components=11)

pca.fit(data\_clean2[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']])

#The amount of variance that each PC explains

varianceexplainedbyPCACOMP= pca.explained\_variance\_ratio\_

##

PCACUMPLOT=np.cumsum(np.round(pca.explained\_variance\_ratio\_, decimals=4)\*100)

plt.plot(PCACUMPLOT)

plt.title("Cumulative varaince of components against number of principal components")

plt.xlabel("Princiapl Component")

plt.ylabel("Cumulative varaince explain")

## wow the first two components hold more than 80% of the variance

##

tran\_pca = pca.fit(data\_clean2[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]).transform(data\_clean2[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']])

df\_pca = pd.DataFrame(tran\_pca)

df\_pca.columns = [['pc1', 'pc2', 'pc3', 'pc4','pc5','pc6','pc7','pc8','pc9','pc10','pc11']]

df\_pca['y'] = merged\_clust\_names\_country[['cluster']]

df\_pca.head()

##lets create a biplot

import seaborn as sns

np\_cluster=merged\_clust\_names\_country[['cluster']].as\_matrix()

# Scatter plot based and assigne color based on 'label - y'

sns.lmplot('pc1', 'pc2', data=df\_pca, fit\_reg = False, size = 15, hue='y', scatter\_kws={"s": 100})

# set the maximum variance of the first two PCs

# this will be the end point of the arrow of each \*\*original features\*\*

xvector = pca.components\_[0]

yvector = pca.components\_[1]

X=data\_clean2[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]

# value of the first two PCs, set the x, y axis boundary

xs = pca.transform(X)[:,0]

ys = pca.transform(X)[:,1]

for i in range(len(xvector)):

# arrows project features (ie columns from csv) as vectors onto PC axes

# we can adjust length and the size of the arrow

plt.arrow(0, 0, xvector[i]\*max(xs), yvector[i]\*max(ys),

color='r', width=0.005, head\_width=0.05)

plt.text(xvector[i]\*max(xs)\*1.1, yvector[i]\*max(ys)\*1.1,

list(X.columns.values)[i], color='r')

np\_df = data\_clean2\_pre[['country']].as\_matrix()

##np\_df[0] rember numpy arrays are 0 indexed

for i in range(len(xs)):

plt.text(xs[i]\*1.08, ys[i]\*1.08, np\_df[i],color='b') # index number of each observations

plt.title('PCA Plot of first PCs')

########

###

##lets try 5 clusters

#

#

##

#######

## Interpret 5 cluster solution

model5=KMeans(n\_clusters=5)

model5.fit(data\_clean2)

clusassign5=model2.predict(data\_clean2)

# create a list that has the new index variable

cluslist5=list(data\_clean2['index'])

# create a list of cluster assignments

labels5=list(model5.labels\_)

# combine index variable list with cluster assignment list into a dictionary

newlist5=dict(zip(cluslist5, labels5))

newlist5

# convert newlist dictionary to a dataframe

newclus5=DataFrame.from\_dict(newlist5, orient='index')

newclus5

# rename the cluster assignment column

newclus5.columns = ['cluster']

newclus5['cluster']

# now do the same for the cluster assignment variable

# create a unique identifier variable from the index for the

# cluster assignment dataframe

# to merge with cluster training data

newclus5.reset\_index(level=0, inplace=True)

newclus5.columns.values

# merge the cluster assignment dataframe with the cluster training variable dataframe

# by the index variable

merged\_clust\_names5=pd.merge(data\_clean2, newclus5,on='index')

merged\_clust\_names5.head(n=100)

merged\_clust\_names5.columns.names

# cluster frequencies

###

#

#

###

##data\_clean2\_pre.reset\_index(level=0, inplace=True)

countrylist=list(data\_clean2\_pre['country'])

countrylistindex=list(data\_clean2\_pre['index'])

newcountrylist=dict(zip(countrylistindex,countrylist))

newcountry=DataFrame.from\_dict(newcountrylist,orient='index')

newcountry.columns = ['country']

newcountry['country']

newcountry.reset\_index(level=0, inplace=True)

newcountry['country']

###

#

#

###

merged\_clust\_names\_country5=pd.merge(merged\_clust\_names5, newcountry, on='index')

merged\_clust\_names\_country5[['country']]

##we can quickly see from applying

clustergrp5 = merged\_clust\_names\_country5.groupby('cluster').mean()

print ("Clustering variable means by cluster")

print(clustergrp5)

'''

Clustering variable means by cluster

index incomeperperson alcconsumption armedforcesrate \

cluster

0 106.210526 1.925011 0.360945 -0.191434

1 111.153846 -0.304801 -1.218275 1.514747

2 108.291667 -0.727085 -0.492013 -0.600894

3 102.484848 -0.503862 -0.074974 -0.272969

4 98.809524 0.069748 1.107717 0.351189

femaleemployrate internetuserate lifeexpectancy employrate \

cluster

0 0.255068 1.530205 1.029620 0.226588

1 -1.745974 -0.216228 0.192075 -1.116786

2 1.287686 -0.991900 -1.184508 1.312131

3 -0.244027 -0.502937 -0.209858 -0.197278

4 -0.238104 0.673314 0.633036 -0.703234

CPI2015 World Economic Forum EOS \

cluster

0 1.736917 1.701811

1 -0.439791 -0.032307

2 -0.777429 -0.777062

3 -0.532239 -0.524499

4 0.425622 0.192549

PRS International Country Risk Guide polityscore

cluster

0 1.718800 0.372376

1 -0.447996 -1.310539

2 -0.655673 -0.591411

3 -0.558362 0.279227

4 0.348993 0.711488

'''

######

###

#

###

#####

##sub1.apply(lambda x: pd.to\_numeric('cluster', errors='ignore'))

polity\_score\_train\_cl5, polity\_score\_test\_cl5 = train\_test\_split(merged\_clust\_names\_country5, test\_size=.3, random\_state=123)

sub2 = polity\_score\_train\_cl5[['cluster','polityscore']]

sub2.dtypes

ggplot(sub2, aes(x='cluster', y='polityscore')) + geom\_boxplot() +ggtitle("boxplot of Polity scores -versus-cluster (5 cluster model)")

sub2['cluster\_str'] = sub2['cluster'].astype(str)

polityscoremod= smf.ols(formula='polityscore ~ C(cluster\_str)', data=sub2).fit()

print (polityscoremod.summary())

''''

print (polityscoremod.summary())

OLS Regression Results

==============================================================================

Dep. Variable: polityscore R-squared: 0.384

Model: OLS Adj. R-squared: 0.350

Method: Least Squares F-statistic: 11.22

Date: Sat, 25 Jun 2016 Prob (F-statistic): 3.95e-07

Time: 02:41:52 Log-Likelihood: -91.891

No. Observations: 77 AIC: 193.8

Df Residuals: 72 BIC: 205.5

Df Model: 4

Covariance Type: nonrobust

=======================================================================================

coef std err t P>|t| [95.0% Conf. Int.]

---------------------------------------------------------------------------------------

Intercept 0.3820 0.213 1.793 0.077 -0.043 0.807

C(cluster\_str)[T.1] -1.4672 0.337 -4.355 0.000 -2.139 -0.796

C(cluster\_str)[T.2] -0.9972 0.289 -3.456 0.001 -1.572 -0.422

C(cluster\_str)[T.3] -0.1227 0.282 -0.435 0.665 -0.685 0.439

C(cluster\_str)[T.4] 0.3947 0.307 1.287 0.202 -0.217 1.006

==============================================================================

Omnibus: 29.390 Durbin-Watson: 1.957

Prob(Omnibus): 0.000 Jarque-Bera (JB): 54.126

Skew: -1.421 Prob(JB): 1.76e-12

Kurtosis: 5.965 Cond. No. 5.97

==============================================================================

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

''''

##yep yep we can reject the NUll hypothesis at 0.05 level

mc5 = multi.MultiComparison(sub2['polityscore'], sub2['cluster\_str'])

res5 = mc5.tukeyhsd()

print(res5.summary())

'''''

mc5 = multi.MultiComparison(sub2['polityscore'], sub2['cluster\_str'])

res5 = mc5.tukeyhsd()

print(res5.summary())

Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================

group1 group2 meandiff lower upper reject

---------------------------------------------

0 1 -1.4672 -2.41 -0.5245 True

0 2 -0.9972 -1.8045 -0.1898 True

0 3 -0.1227 -0.9115 0.666 False

0 4 0.3947 -0.4634 1.2529 False

1 2 0.47 -0.4408 1.3809 False

1 3 1.3445 0.4501 2.2389 True

1 4 1.862 0.9058 2.8181 True

2 3 0.8744 0.1242 1.6247 True

2 4 1.3919 0.569 2.2149 True

3 4 0.5175 -0.2872 1.3222 False

---------------------------------------------

'''''

########

####

##

####

#######

df\_pca['y'] = merged\_clust\_names\_country5[['cluster']]

df\_pca.head(109)

##lets create a biplot

# Scatter plot based and assigne color based on 'label - y'

sns.lmplot('pc1', 'pc2', data=df\_pca, fit\_reg = False, size = 15, hue='y', scatter\_kws={"s": 100})

# set the maximum variance of the first two PCs

# this will be the end point of the arrow of each \*\*original features\*\*

xvector = pca.components\_[0]

yvector = pca.components\_[1]

X=data\_clean2[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]

# value of the first two PCs, set the x, y axis boundary

xs = pca.transform(X)[:,0]

ys = pca.transform(X)[:,1]

for i in range(len(xvector)):

# arrows project features (ie columns from csv) as vectors onto PC axes

# we can adjust length and the size of the arrow

plt.arrow(0, 0, xvector[i]\*max(xs), yvector[i]\*max(ys),

color='r', width=0.005, head\_width=0.05)

plt.text(xvector[i]\*max(xs)\*1.1, yvector[i]\*max(ys)\*1.1,

list(X.columns.values)[i], color='r')

np\_df = data\_clean2\_pre[['country']].as\_matrix()

##np\_df[0] rember numpy arrays are 0 indexed

for i in range(len(xs)):

plt.text(xs[i]\*1.08, ys[i]\*1.08, np\_df[i],color='b') # index number of each observations

plt.title('PCA Plot of first PCs PCA1 and PCA2')

#####

##

#

##

#####

'''

Besides looking at Just the PCA plots lets look at another dimension reduction technique t-sne and visualize our data that way

'''

from sklearn.manifold import TSNE

data2tsne=datafullset[['country','incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

'polityscore']]

##run categorization on polityscore

def polityscore\_cat (row):

if (row['polityscore'] >=6 and row['polityscore'] <= 10 ) :

return 1 ##democracy

elif (row['polityscore'] >=-5 and row['polityscore'] <= 5 ) :

return 2 ##anocracy

elif (row['polityscore'] >=-10 and row['polityscore'] <= -6 ) :

return 3 ##autocracy

else :

return 0 ##unknown

##calculate the age of NATO countries

##data['Years\_In\_Nato'] = data.apply (lambda row: AGE\_YEARS (row),axis=1)

data2tsne['polityscore\_cat'] = data2tsne.apply (lambda row: polityscore\_cat (row),axis=1)

##drop NA values

data2tsne=data2tsne.dropna()

##Explore

polityscoredata=data2tsne[['incomeperperson', 'alcconsumption', 'armedforcesrate',

'femaleemployrate',

'internetuserate', 'lifeexpectancy',

'employrate',

'CPI2015','World Economic Forum EOS','PRS International Country Risk Guide',

]]

polityscoretarget= data2tsne.polityscore\_cat

polityscoredata['incomeperperson']=preprocessing.scale(polityscoredata['incomeperperson'].astype('float64'))

polityscoredata['alcconsumption']=preprocessing.scale(polityscoredata['alcconsumption'].astype('float64'))

polityscoredata['armedforcesrate']=preprocessing.scale(polityscoredata['armedforcesrate'].astype('float64'))

polityscoredata['femaleemployrate']=preprocessing.scale(polityscoredata['femaleemployrate'].astype('float64'))

polityscoredata['internetuserate']=preprocessing.scale(polityscoredata['internetuserate'].astype('float64'))

polityscoredata['lifeexpectancy']=preprocessing.scale(polityscoredata['lifeexpectancy'].astype('float64'))

polityscoredata['employrate']=preprocessing.scale(polityscoredata['employrate'].astype('float64'))

polityscoredata['CPI2015']=preprocessing.scale(polityscoredata['CPI2015'].astype('float64'))

polityscoredata['World Economic Forum EOS']=preprocessing.scale(polityscoredata['World Economic Forum EOS'].astype('float64'))

polityscoredata['PRS International Country Risk Guide']=preprocessing.scale(polityscoredata['PRS International Country Risk Guide'].astype('float64'))

X\_tsne = TSNE(learning\_rate=100).fit\_transform(polityscoredata)

X\_pca = PCA().fit\_transform(polityscoredata)

plt.scatter(X\_tsne[:, 0], X\_tsne[:, 1], c=polityscoretarget)

plt.title("T-sne Plot categories Polity scores")

## -SNE can help us to decide whether classes are separable in some linear or nonlinear representation. Here we can see that the 3 classes of the Iris dataset can be separated quite easily

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=polityscoretarget)

plt.title("PCA 1 and 2")