HW2

September 26, 2016

1 HW2 - CSE 519

Name: Piyush Shyam Banginwar
SBU ID: 110944214
E-mail: pbanginwar@cs.stonybrok.edu

1.1 Preprocessing and Data Cleanup

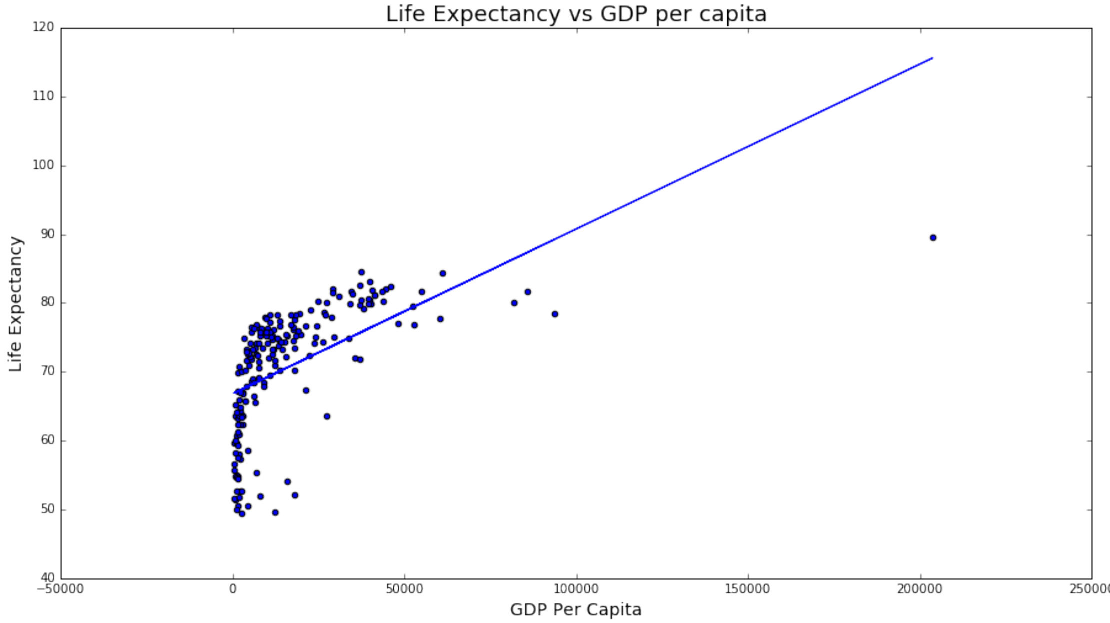
```
In [55]: #Importing all necessary pkgs
         import pandas as pd
         import numpy as np
         import math
         import scipy.stats as sct
         import matplotlib.pyplot as plt; plt.rcdefaults()
         import operator
         import random
         %matplotlib inline

         #'country-data.csv' has been modified by replacing 'unknown', 'NA' values
         df = pd.read_csv('country-data.csv', ',')

         #replacing NaN values with mean of the columns
         for x in df:
             if df[str(x)].dtype != "object":
                 # 'np.nanmean' finds mean by ignoring NaN values, we then replace
                 mean = np.nanmean(df[str(x)])
                 for i,y in enumerate(df[str(x)]):
                     if y=='nan' or y=="nan" or math.isnan(y):
                         df[str(x)][i] = mean

         #Defining some useful terms which can be used later
         gdp_per_capita = df.GDP/df.Population
         population_density = df.Population/df.Area
         labor_force_ratio = df.Labor_Force/df.Population
         internet_users_ratio = df.Internet_Users/df.Population
```

```
/home/piyush/anaconda2/lib/python2.7/site-packages/ipykernel/__main__.py:21: Settir
A value is trying to be set on a copy of a slice from a DataFrame
```

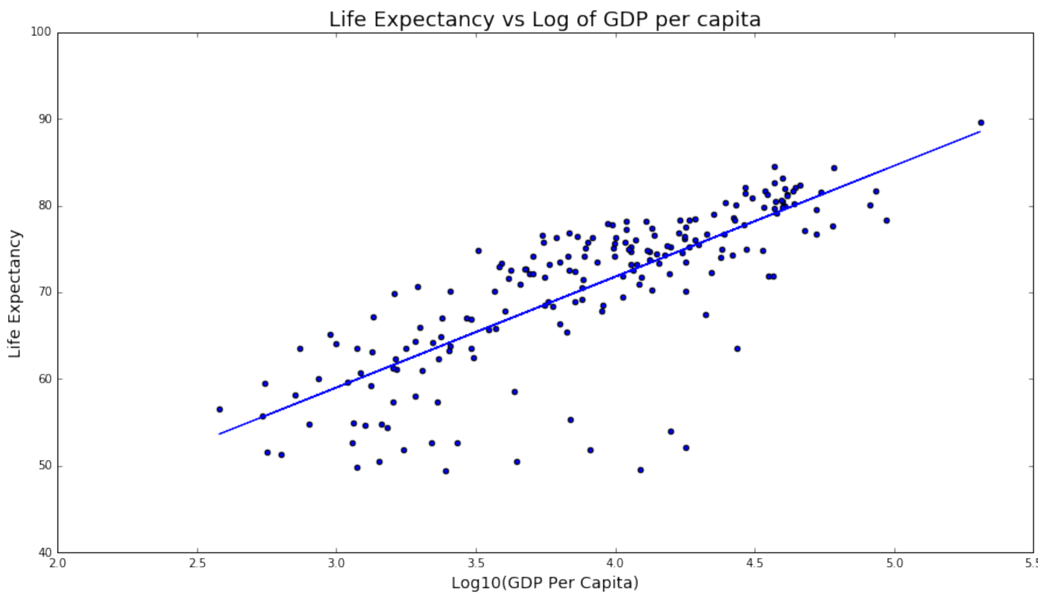
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i

1.2 Q1 : Informative Plots

In [56]:
```
# 1. Scatter Plots
#Plotting a scatter plot between GDP Per Capita and Life Expectancy along
plt.figure(figsize=(15, 8))
plt.scatter(gdp_per_capita, df.Life_Expectancy)
plt.plot(gdp_per_capita, np.poly1d(np.polyfit(gdp_per_capita, df.Life_Expe
plt.ylabel('Life Expectancy', {'fontsize':14})
plt.xlabel('GDP Per Capita', {'fontsize':14})
plt.title('Life Expectancy vs GDP per capita', {'fontsize':18})
description = "The plot shows the relation between Life Expectancy and GDP
plt.annotate(description, xy=(0, -0.2), size=14, xycoords='axes fraction',
plt.show()

#Avoiding outliers for the above by taking logarithm base 10 values
gdp_list = np.log10(gdp_per_capita)
plt.figure(figsize=(15, 8))
plt.scatter(gdp_list, df.Life_Expectancy)
plt.plot(gdp_list, np.poly1d(np.polyfit(gdp_list, df.Life_Expectancy, 1))
plt.ylabel('Life Expectancy', {'fontsize':14})
plt.xlabel('Log10(GDP Per Capita)', {'fontsize':14})
plt.title('Life Expectancy vs Log of GDP per capita', {'fontsize':18})
description = "The plot shows the relation between Life Expectancy and Log
plt.annotate(description, xy=(0, -0.23), size=14, xycoords='axes fraction'
plt.show()
```



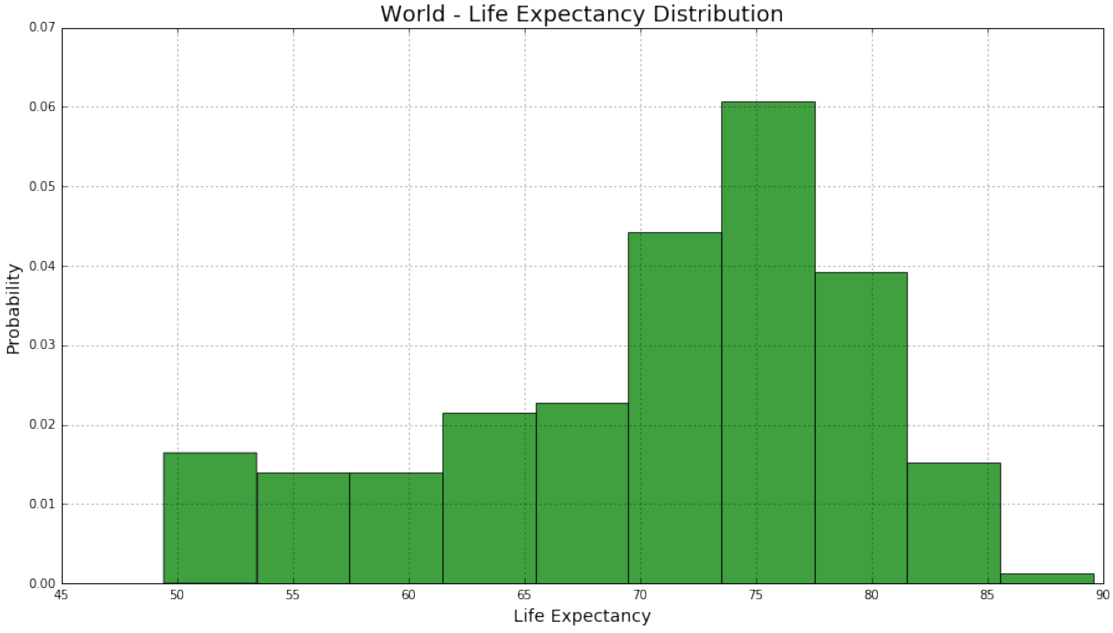
The plot shows the relation between Life Expectancy and GDP Per Capita.
We can see here that most countries are lying in the same region and that the graph is skewed due to outliers, which needs to be handled.



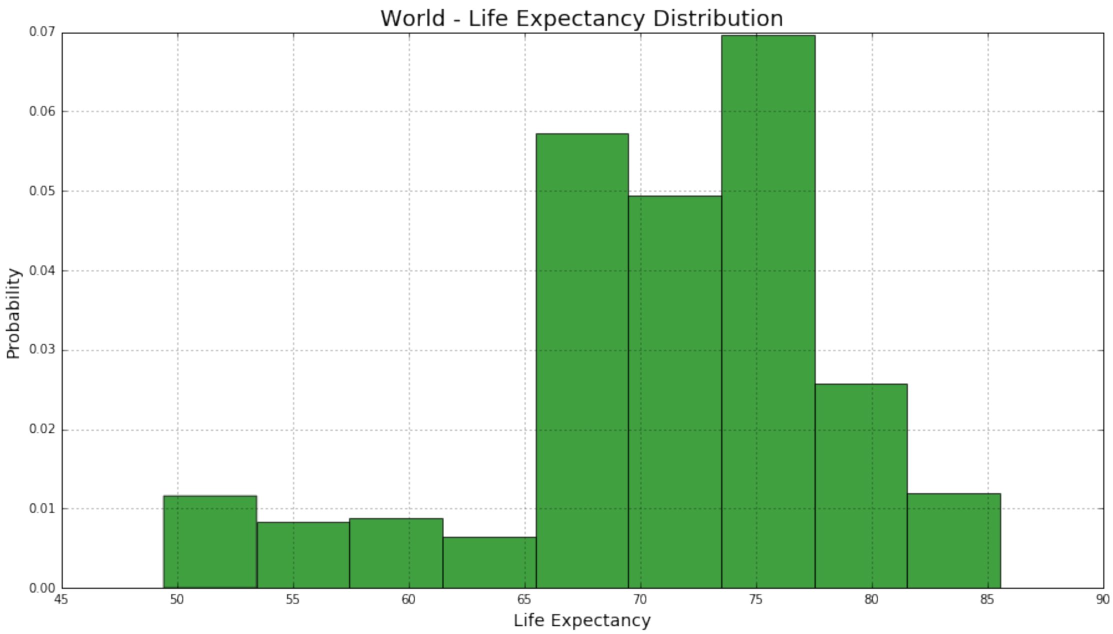
The plot shows the relation between Life Expectancy and Logarithm of GDP Per Capita base 10.
This shows us an almost linear relationship between the GDP Per Capita of a country and average life expectancy of its citizens.
We can make a general inference here that a nation where its citizens are earning well, will be having access to good healthcare facilities, better environment, food etc. increasing their life expectancy.

```
In [57]: # 2. Histogram
```

```
#Plotting a histogram for distribution of Life Expectancy of Countries
plt.figure(figsize=(15, 8))
n, bins, patches = plt.hist([df.Life_Expectancy], 10, normed=1, facecolor=
plt.xlabel('Life Expectancy', {'fontsize':14})
plt.ylabel('Probability', {'fontsize':14})
plt.title('World - Life Expectancy Distribution', {'fontsize':18})
description = "The plot shows the probability distribution of Life Expecta
plt.annotate(description, xy=(0, -0.23), size=14, xycoords='axes fraction'
plt.grid(True)
plt.show()

#Plotting a histogram for distribution of Life Expectancy of Countries we
plt.figure(figsize=(15, 8))
n, bins, patches = plt.hist([df.Life_Expectancy], 10, normed=1, weights =
plt.xlabel('Life Expectancy', {'fontsize':14})
plt.ylabel('Probability', {'fontsize':14})
plt.title('World - Life Expectancy Distribution', {'fontsize':18})
description = "The plot shows the actual probability distribution of Life
plt.annotate(description, xy=(0, -0.23), size=14, xycoords='axes fraction'
plt.grid(True)
plt.show()
```



The plot shows the probability distribution of Life Expectancy of world population.
The graph is plotted simply from the values of each country but does not take into account its population
which might be giving us an incorrect picture.

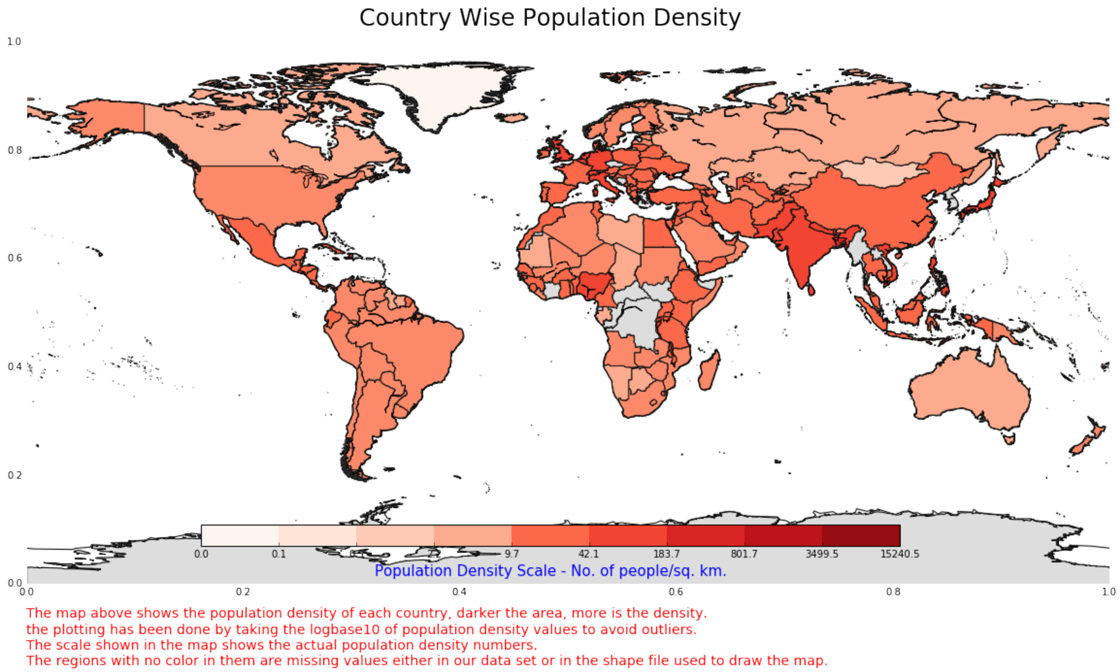


The plot shows the actual probability distribution of Life Expectancy of world population.
This graph takes into account the country population associated with each life expectancy entry
(by giving weights=country population to each life expectancy entry) we have and hence
gives us a more clear picture of the Life Expectancy Probablity distribution for world population

In [58]:
```
# 3. Data Map:

"""
References taken from:
http://ramiro.org/notebook/basemap-choropleth/
http://matplotlib.org/basemap/
http://www.naturalearthdata.com/downloads/10m-cultural-vectors/
"""
import matplotlib as mp
from mpl_toolkits.basemap import Basemap
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection

#Scaling down to avoid outliers in population density
values = np.log10(population_density)

#Creating a color map scheme to be used for plotting in map
num_colors = 10
cm = plt.get_cmap('Reds')
scheme = [cm(1.*i / num_colors) for i in range(num_colors)]
bins = np.linspace(values.min(), values.max(), num_colors)
df['bin'] = np.digitize(values, bins) - 1

#Creating the map
```

```
map = Basemap(lon_0=0)
fig = plt.figure(figsize=(20, 10))
fig.suptitle('Country Wise Population Density', fontsize=24, y=.95)
description = "The map above shows the population density of each country,
plt.annotate(description, xy=(0.0, -0.15), size=14, xycoords='axes fractio
ax = fig.add_subplot(111, axisbg='w', frame_on=False)
map.drawcoastlines()
map.drawmapboundary(color='w')
shapefile = 'mapshape/ne_10m_admin_0_countries_1'
map.readshapefile(shapefile, 'units', color='#444444', linewidth=.2)

for info, shape in zip(map.units_info, map.units):
    name = info['NAME']
    if name not in df.Name.values:
        color = '#dddddd'
    else:
        row = df.loc[df['Name'] == name]
        index = row.values[0][21]
        color = scheme[index]
    patches = [Polygon(np.array(shape), True)]
    pc = PatchCollection(patches)
    pc.set_facecolor(color)
    ax.add_collection(pc)

# Draw color legend.
ax_legend = fig.add_axes([0.25, 0.18, 0.5, 0.03], zorder=3)
cmap = mp.colors.ListedColormap(scheme)
cb = mp.colorbar.ColorbarBase(ax_legend, cmap=cmap, ticks=bins, boundaries
cb.ax.set_xticklabels([str(round(pow(10,round(i, 3)),1)) for i in bins])
cb.ax.set_xlabel('Population Density Scale - No. of people/sq. km.', {'for
```

```
Out[58]: <matplotlib.text.Text at 0xa5cb326c>
```



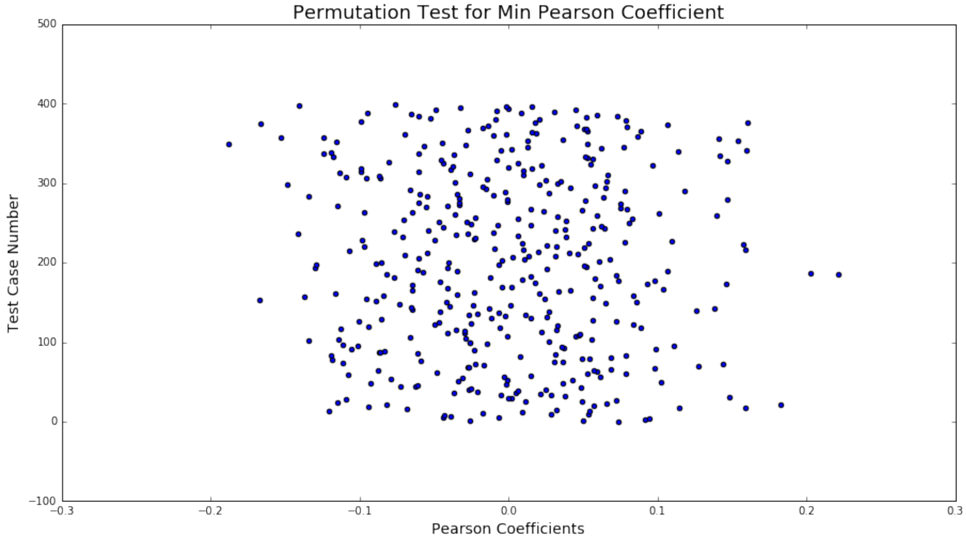
1.3 Q2: Pearson Correlation Analysis

```
In [59]: #Pearson coefficient calculation
         import json
         import random
         coeff_list = []
         for x in df:
             for y in df:
                 if x != y and df[str(x)].dtype != "object" and df[str(y)].dtype !=
                     #print sct.pearsonr(df[str(x)],df[str(y)]), " between", x, " a
                     value = sct.pearsonr(df[str(x)],df[str(y)])
                     coeff_list.append((value[0], value[1], x, y))

         answer = []
         answer.append(coeff_list[coeff_list.index(min(coeff_list, key=operator.ite
         answer.append(coeff_list[coeff_list.index(max(coeff_list, key=operator.ite

         jsonData = {
             "Email": "pbanginwar@cs.stonybrook.edu",
             "Name": "Piyush Shyam Banginwar",
             "Q2": {
                 "Negative": {
                     "Attr0": answer[0][3],
                     "Attr1": answer[0][2],
                     "coeff": round(answer[0][0],4),
```

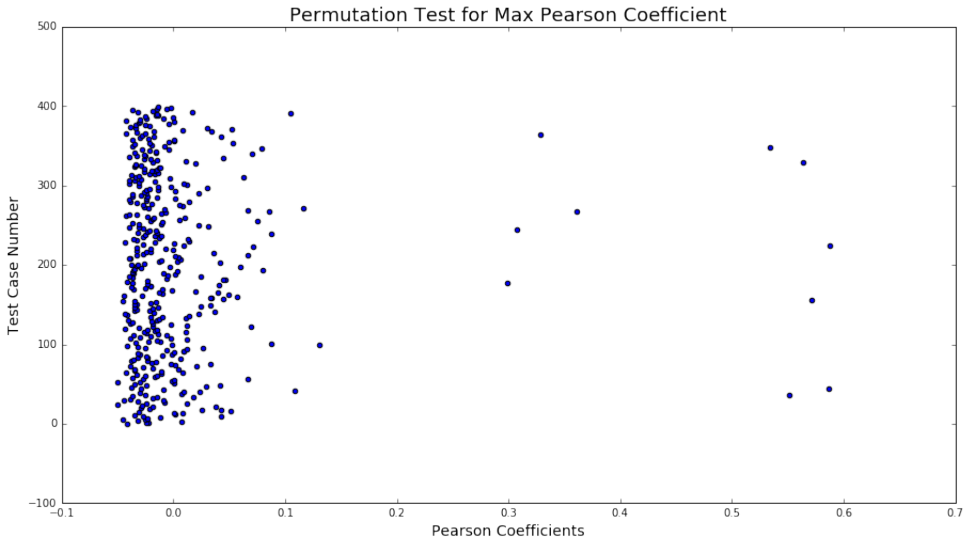
```
                "pValue": answer[0][1]
            },
            "Positive": {
                "Attr0": answer[1][3],
                "Attr1": answer[1][2],
                "coeff": round(answer[1][0],4),
                "pValue": answer[1][1]
            }
        },
        "SBUID": "110944214"
    }

    # Writing JSON data into a file called HW2.json
    with open('HW2.json', 'w') as f:
         json.dump(jsonData, f)
```

```
In [60]: #Permutation Test

         #1. For Min coefficient value
         coeff_list_random_min = [[],[]]
         for i in range(400):
             random_list = pd.DataFrame({'A':df[str(answer[0][3])].copy()})
             random_list.apply(random.shuffle, axis=0)
             value = sct.pearsonr(random_list.A,df[str(answer[0][2])])
             coeff_list_random_min[0].append(value[0])
             coeff_list_random_min[1].append(value[1])

         y_pos = np.arange(0,400)
         plt.figure(figsize=(15, 8))
         plt.xlabel('Pearson Coefficients', {'fontsize':14})
         plt.ylabel('Test Case Number', {'fontsize':14})
         plt.title('Permutation Test for Min Pearson Coefficient', {'fontsize':18})
         description = "The plot shows that almost all values we got from permutati
         plt.annotate(description, xy=(0, -0.18), size=14, xycoords='axes fraction'
         plt.scatter(coeff_list_random_min[0], y_pos)
         plt.show()

         #2. For Max coefficient value
         coeff_list_random_max = [[],[]]
         for i in range(400):
             random_list = pd.DataFrame({'A':df[str(answer[1][3])].copy()})
             random_list.apply(random.shuffle, axis=0)
             value = sct.pearsonr(random_list.A,df[str(answer[1][2])])
             coeff_list_random_max[0].append(value[0])
             coeff_list_random_max[1].append(value[1])

         y_pos = np.arange(0,400)
```

```
plt.figure(figsize=(15, 8))
plt.xlabel('Pearson Coefficients', {'fontsize':14})
plt.ylabel('Test Case Number', {'fontsize':14})
plt.title('Permutation Test for Max Pearson Coefficient', {'fontsize':18})
description = "The plot shows that almost all values we got from permutati
plt.annotate(description, xy=(0, -0.18), size=14, xycoords='axes fraction'
plt.scatter(coeff_list_random_max[0], y_pos)
plt.show()
```



The plot shows that almost all values we got from permutation tests lie between -0.2 to 0.2 and none are close to the actual value of -0.44.
We can thus infer that the Pearson Coefficient value we got was significant.



The plot shows that almost all values we got from permutation tests lie between -0.1 to 0.2 and none are close to the actual value of -0.98.
We can thus infer that the Pearson Coefficient value we got was significant.

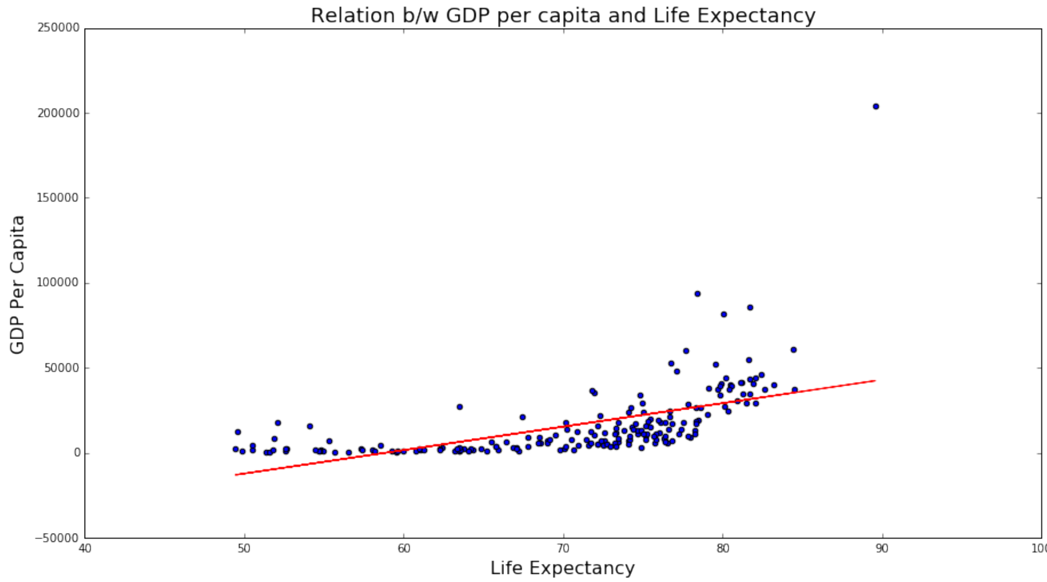
1.4 Q3: Linear Regression Model

In [61]:
```
#Linear Regression Model
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing

coeff_list = []
for x in df:
    if df[str(x)].dtype != "object":
        (a,b) = np.polyfit(df[str(x)], gdp_per_capita, deg=1)
        err =  np.sqrt(mean_squared_error(np.polyval([a,b], df[str(x)]), g
        coeff_list.append((a, b, err, x))
answer = []
answer.append(coeff_list[coeff_list.index(min(coeff_list, key=operator.ite
answer.append(coeff_list[coeff_list.index(max(coeff_list, key=operator.ite

a = answer[0][0]
b = answer[0][1]
x = answer[0][3]
min_error_regr_list = np.polyval([a,b], df[str(x)])
forecast_diff = gdp_per_capita - min_error_regr_list
mostAbove = df.Name[forecast_diff.argmax()]
mostBelow = df.Name[forecast_diff.argmin()]

print "The country most above the forecast is:", mostAbove
print "The country most below the forecast is:", mostBelow

#Write the answer to JSON file
a_dict = {
    "Q3": {
        "Attr": answer[0][3],
        "SquaredError": round(answer[0][2],4),
        "TCMostAboveForecast": mostAbove,
        "TCMostBelowForecast": mostBelow
    }}
with open('HW2.json') as f:
    data = json.load(f)
data.update(a_dict)
with open('HW2.json', 'w') as f:
    json.dump(data, f)

plt.figure(figsize=(15, 8))
description = """Explanation: The linear regression model with least mean
The country most below the forecast is: Solomon Islands, this can be be at
plt.annotate(description, xy=(0, -0.45), size=15, xycoords='axes fraction'
plt.xlabel('Life Expectancy', {'fontsize':16})
plt.ylabel('GDP Per Capita', {'fontsize':16})
```

```
plt.title('Relation b/w GDP per capita and Life Expectancy', {'fontsize':1
plt.plot(df.Life_Expectancy, a * df.Life_Expectancy + b, color='red')
plt.scatter(df.Life_Expectancy, gdp_per_capita)
```

```
The country most above the forecast is: Monaco
The country most below the forecast is: Solomon Islands
```

```
Out[61]: <matplotlib.collections.PathCollection at 0x9e3b466c>
```



Explanation: The linear regression model with least mean square error has been the one with Life Expectancy.
Here, the country most above the forecast is Monaco and this seems pretty correct.
As per Wikipedia: Monaco has the world's second highest GDP nominal per capita and the lowest poverty rate.
This can be attributed to its gambling industry,favorable tax policies making it a tax haven and large tourism industry.

The country most below the forecast is: Solomon Islands, this can be be attributed to its underdeveloped state.
It's economy mainly depends on fisheries and timber exports with most of the high end goods and petroleum products
being imported into the country. Only 60% of school age children have access to primary education, this reflects
its poor economic state and hence the forecast makes sense.

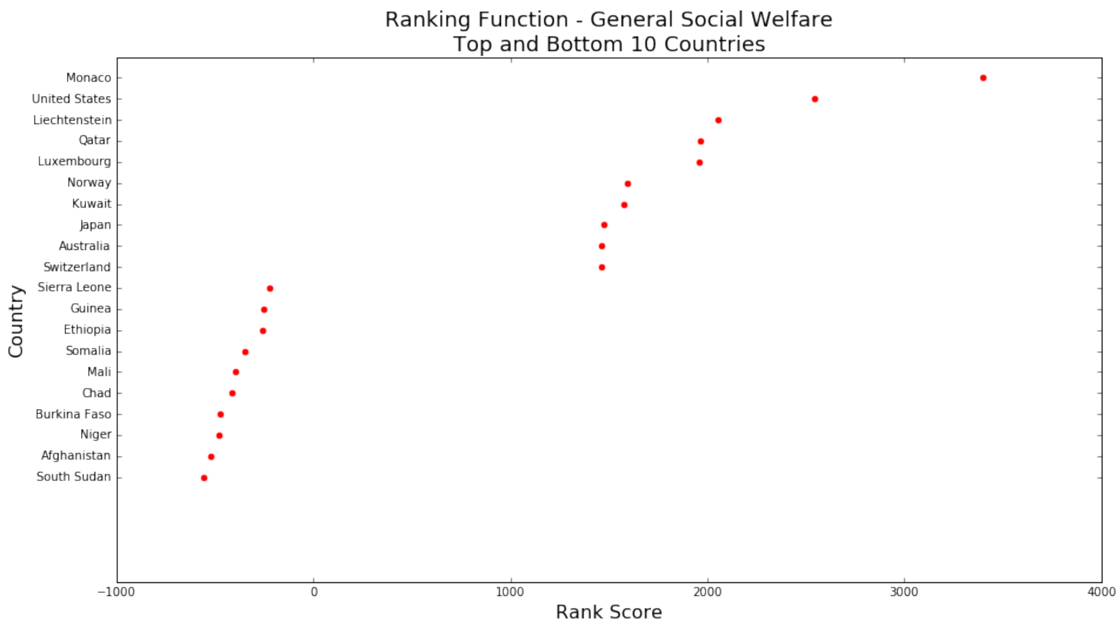
1.5 Q4: Ranking/Scoring Function

```
In [62]: #Reference: http://scikit-learn.org/stable/modules/generated/sklearn.prepr

         #Scaling all variables to a range of (0,cap), where cap = 10000
         #This is done by: X_std = (X - X.min) / (X.max - X.min)
         #                 X_scaled = X_std * (max - min) + min
         #Using an API referred above to do the computation
         cap = 10000
         gdp_per_capita_scaled = preprocessing.minmax_scale(gdp_per_capita, (0,cap)
         population_density_scaled = preprocessing.minmax_scale(population_density,
         life_expectancy_scaled = preprocessing.minmax_scale(df.Life_Expectancy, (0
         literacy_rate_scaled = preprocessing.minmax_scale(df.Literacy_Rate, (0,cap
```

```
renewable_water_ratio_scaled = preprocessing.minmax_scale(df.Renewable_Wat
labor_force_ratio_scaled = preprocessing.minmax_scale(labor_force_ratio,
internet_users_ratio_scaled = preprocessing.minmax_scale(internet_users_ra
migration_rate_scaled = preprocessing.minmax_scale(df.Net_Migration_Rate,

#The values in Health Expenditure/GDP, Military Expenditure/GDP is the per
# where, the actual values are:
actual_health_expenditure = df.Health_Expenditure * df.GDP / 100
actual_military_expenditure = df.Military_Expenditure * df.GDP / 100
health_expenditure_scaled = preprocessing.minmax_scale(actual_health_exper
military_expenditure_scaled = preprocessing.minmax_scale(actual_military_e

rank = (((0.3) * gdp_per_capita_scaled) +
         ((0.05) * life_expectancy_scaled) +
         ((0.1) * literacy_rate_scaled) +
         ((0.05) * health_expenditure_scaled) +
         ((0.05) * military_expenditure_scaled) +
         ((0.05) * renewable_water_ratio_scaled) +
         ((0.1) * internet_users_ratio_scaled) +
        ((-0.1) * population_density_scaled) +
        ((-0.1) * migration_rate_scaled) +
        ((-0.1) * labor_force_ratio_scaled))

mostAbove = df.Name[rank.argmax()]
mostBelow = df.Name[rank.argmin()]
print "Highest Rank: ",mostAbove
print "Lowest Rank: ",mostBelow

#lets create a list of bottom and top 10 countries and plot them
new_list = pd.DataFrame({'A':rank,'B':df.Name})
sorted_new_list = new_list.sort_values(by='A',axis=0)
rank_list = [[],[]]
i=0
for index, row in sorted_new_list.iterrows():
    if i<10:
        #print row['B']
        rank_list[0].append(row['A'])
        rank_list[1].append(row['B'])
    if i>186:
        rank_list[0].append(row['A'])
        rank_list[1].append(row['B'])
    i+=1

#x_pos = np.arange(0,len(df.Name))
x_pos = np.arange(0,len(rank_list[1]))
plt.figure(figsize=(15, 8))
description = """The plot shows the top and bottom 10 countries ranked by
Explanation: I have considered numerous factors to calculate the rank/scor
```

```
The assignment of weights is based on intuition about how significant(and
Following factors(alongwith weights) have been considered to calculate the
GDP Per Capita - +0.3\nLife Expectancy - +0.05\nLiteracy Rate - +0.1\nHeal
plt.annotate(description, xy=(0, -0.90), size=15, xycoords='axes fraction'
plt.xlabel('Rank Score', {'fontsize':16})
plt.ylabel('Country', {'fontsize':16})
plt.title('Ranking Function - General Social Welfare\nTop and Bottom 10 Co
plt.scatter(rank_list[0],x_pos, color='red')
plt.yticks(x_pos, rank_list[1])
plt.show()
```

```
Highest Rank:  Monaco
Lowest Rank:  South Sudan
```



The plot shows the top and bottom 10 countries ranked by the Rank/Scoring Function.
The outcomes are quite close to actual country rankings and we can infer that the
scoring function has done a decent job of ranking them.

Explanation: I have considered numerous factors to calculate the rank/scoring function,
and have scaled all factors to a range of (0-10,000) to avoid scaling issues between factors.
Along with this, weights between (0,1) have been assigned to each factor with the total weight being 1.0.
The assignment of weights is based on intuition about how significant(and in what way - positive/negaive)
a role a factor has to play towards social welfare of citizens of a country.

Following factors(alongwith weights) have been considered to calculate the scoring function:
GDP Per Capita - +0.3
Life Expectancy - +0.05
Literacy Rate - +0.1
Health Expenditure - +0.05
Military Expenditure - +0.05
Renewable Water Resources - +0.05
Internet Users - +0.1
Population Density - -0.1
Migration Rate - -0.1
Labor Force Ratio - -0.1

1.6 Q5: Distance Function

```
In [63]: #Using the Rank calculated in earlier question to calcualte the distance b
         # Logic used: Distance(a,b) = absolute difference between ranks of a & b.
         #Where ranks will be normalized before usage.

         #Normalizing ranks computed earlier
         rank_normalized = preprocessing.normalize(rank)

         #Distance Function
         def distance(x,y):
             return abs(rankFn(x) - rankFn(y))
         def rankFn(x):
             return rank_normalized[0][df.Name.tolist().index(x)]

         distance_answer = []
         for x in df.Name:
             distance_matrix = []
             for y in df.Name:
                 if x != y:
                     #print x,y
                     distance_val = distance(x,y)
                     distance_matrix.append((distance_val, x, y))
             distance_answer.append((x, distance_matrix[distance_matrix.index(min(d

         print "Using the Rank calculated in earlier question to calcualte the dist
         print "\nTable Format:\n'Country Name' 'Nearest Country' 'Farthest Country
         print np.asarray(distance_answer)
```

```
/home/piyush/anaconda2/lib/python2.7/site-packages/sklearn/utils/validation.py:386:
  DeprecationWarning)


Using the Rank calculated in earlier question to calcualte the distance between two
Logic used: Distance(a,b) = absolute difference between ranks of a & b.
Where ranks will be normalized before usage.

Table Format:
'Country Name' 'Nearest Country' 'Farthest Country'
[['Afghanistan' 'South Sudan' 'Monaco']
 ['Albania' 'Montenegro' 'Monaco']
 ['Algeria' 'Nicaragua' 'Monaco']
 ['Andorra' 'France' 'Monaco']
 ['Angola' 'Eritrea' 'Monaco']
 ['Antigua and Barbuda' 'Cyprus' 'Monaco']
 ['Argentina' 'Hungary' 'Monaco']
 ['Armenia' 'Bosnia and Herzegovina' 'Monaco']
 ['Australia' 'Switzerland' 'South Sudan']
 ['Austria' 'Iceland' 'Monaco']
```

```
['Azerbaijan' 'Bosnia and Herzegovina' 'Monaco']
['Bahamas, The' 'Czech Republic' 'Monaco']
['Bahrain' 'Belarus' 'Monaco']
['Bangladesh' 'Togo' 'Monaco']
['Barbados' 'Chile' 'Monaco']
['Belarus' 'Bahrain' 'Monaco']
['Belgium' 'Ireland' 'Monaco']
['Belize' 'Botswana' 'Monaco']
['Benin' 'Sierra Leone' 'Monaco']
['Bhutan' 'Rwanda' 'Monaco']
['Bolivia' 'Tunisia' 'Monaco']
['Bosnia and Herzegovina' 'Azerbaijan' 'Monaco']
['Botswana' 'Nicaragua' 'Monaco']
['Brazil' 'Maldives' 'Monaco']
['Brunei' 'Canada' 'South Sudan']
['Bulgaria' 'Panama' 'Monaco']
['Burkina Faso' 'Niger' 'Monaco']
['Burundi' 'Pakistan' 'Monaco']
['Cabo Verde' 'South Africa' 'Monaco']
['Cambodia' 'Laos' 'Monaco']
['Cameroon' 'Yemen' 'Monaco']
['Canada' 'Switzerland' 'South Sudan']
['Central African Republic' 'Mozambique' 'Monaco']
['Chad' 'Mali' 'Monaco']
['Chile' 'Barbados' 'Monaco']
['China' 'Singapore' 'Monaco']
['Colombia' 'Georgia' 'Monaco']
['Comoros' 'Sudan' 'Monaco']
['Congo, Democratic Republic of the' 'Bangladesh' 'Monaco']
['Congo, Republic of the' 'Morocco' 'Monaco']
['Costa Rica' 'Mexico' 'Monaco']
["Cote d'Ivoire" 'Gambia, The' 'Monaco']
['Croatia' 'Latvia' 'Monaco']
['Cuba' 'Panama' 'Monaco']
['Cyprus' 'Antigua and Barbuda' 'Monaco']
['Czech Republic' 'Bahamas, The' 'Monaco']
['Denmark' 'San Marino' 'Monaco']
['Djibouti' 'Yemen' 'Monaco']
['Dominica' 'Malaysia' 'Monaco']
['Dominican Republic' 'Paraguay' 'Monaco']
['Ecuador' 'Thailand' 'Monaco']
['Egypt' 'Belize' 'Monaco']
['El Salvador' 'Jordan' 'Monaco']
['Equatorial Guinea' 'Mexico' 'Monaco']
['Eritrea' 'Zimbabwe' 'Monaco']
['Estonia' 'Syria' 'Monaco']
['Ethiopia' 'Guinea' 'Monaco']
['Fiji' 'Guyana' 'Monaco']
```

```
['Finland' 'Ireland' 'Monaco']
['France' 'Andorra' 'Monaco']
['Gabon' 'Solomon Islands' 'Monaco']
['Gambia, The' 'Nigeria' 'Monaco']
['Georgia' 'Colombia' 'Monaco']
['Germany' 'Brunei' 'South Sudan']
['Ghana' 'India' 'Monaco']
['Greece' 'Saudi Arabia' 'Monaco']
['Greenland' 'Israel' 'Monaco']
['Grenada' 'Macedonia' 'Monaco']
['Guatemala' 'Kenya' 'Monaco']
['Guinea' 'Ethiopia' 'Monaco']
['Guinea-Bissau' 'Central African Republic' 'Monaco']
['Guyana' 'Mongolia' 'Monaco']
['Haiti' 'Senegal' 'Monaco']
['Honduras' 'Burma' 'Monaco']
['Hungary' 'Argentina' 'Monaco']
['Iceland' 'Austria' 'Monaco']
['India' 'Ghana' 'Monaco']
['Indonesia' 'Korea, North' 'Monaco']
['Iran' 'Kosovo' 'Monaco']
['Iraq' 'Kiribati' 'Monaco']
['Ireland' 'Belgium' 'Monaco']
['Israel' 'United Arab Emirates' 'Monaco']
['Italy' 'New Zealand' 'Monaco']
['Jamaica' 'Tajikistan' 'Monaco']
['Japan' 'Australia' 'South Sudan']
['Jordan' 'El Salvador' 'Monaco']
['Kazakhstan' 'Serbia' 'Monaco']
['Kenya' 'Guatemala' 'Monaco']
['Kiribati' 'Iraq' 'Monaco']
['Korea, North' 'Iran' 'Monaco']
['Korea, South' 'Taiwan' 'Monaco']
['Kosovo' 'Iran' 'Monaco']
['Kuwait' 'Norway' 'South Sudan']
['Kyrgyzstan' 'Marshall Islands' 'Monaco']
['Laos' 'Sudan' 'Monaco']
['Latvia' 'Croatia' 'Monaco']
['Lebanon' 'Swaziland' 'Monaco']
['Lesotho' 'Timor-Leste' 'Monaco']
['Liberia' 'Nigeria' 'Monaco']
['Libya' 'Philippines' 'Monaco']
['Liechtenstein' 'Qatar' 'South Sudan']
['Lithuania' 'Trinidad and Tobago' 'Monaco']
['Luxembourg' 'Qatar' 'South Sudan']
['Macedonia' 'Turkey' 'Monaco']
['Madagascar' 'Uganda' 'Monaco']
['Malawi' 'Angola' 'Monaco']
```

```
['Malaysia' 'Romania' 'Monaco']
['Maldives' 'Suriname' 'Monaco']
['Mali' 'Chad' 'Monaco']
['Malta' 'Oman' 'Monaco']
['Marshall Islands' 'Kyrgyzstan' 'Monaco']
['Mauritania' 'Nepal' 'Monaco']
['Mauritius' 'Uzbekistan' 'Monaco']
['Mexico' 'Costa Rica' 'Monaco']
['Micronesia, Federated States of' 'Peru' 'Monaco']
['Moldova' 'Peru' 'Monaco']
['Monaco' 'United States' 'South Sudan']
['Mongolia' 'Guyana' 'Monaco']
['Montenegro' 'Venezuela' 'Monaco']
['Morocco' 'Congo, Republic of the' 'Monaco']
['Mozambique' 'Haiti' 'Monaco']
['Burma' 'Honduras' 'Monaco']
['Namibia' 'Western Sahara' 'Monaco']
['Nauru' 'Tunisia' 'Monaco']
['Nepal' 'Mauritania' 'Monaco']
['Netherlands' 'Tuvalu' 'Monaco']
['New Zealand' 'Italy' 'Monaco']
['Nicaragua' 'Botswana' 'Monaco']
['Niger' 'Burkina Faso' 'Monaco']
['Nigeria' 'Gambia, The' 'Monaco']
['Norway' 'Kuwait' 'South Sudan']
['Oman' 'Malta' 'Monaco']
['Pakistan' 'Togo' 'Monaco']
['Palau' 'Spain' 'Monaco']
['Panama' 'Cuba' 'Monaco']
['Papua New Guinea' 'Zimbabwe' 'Monaco']
['Paraguay' 'Dominican Republic' 'Monaco']
['Peru' 'Moldova' 'Monaco']
['Philippines' 'Libya' 'Monaco']
['Poland' 'Estonia' 'Monaco']
['Portugal' 'Barbados' 'Monaco']
['Qatar' 'Luxembourg' 'South Sudan']
['Romania' 'Malaysia' 'Monaco']
['Russia' 'Trinidad and Tobago' 'Monaco']
['Rwanda' 'Uganda' 'Monaco']
['Saint Kitts and Nevis' 'Antigua and Barbuda' 'Monaco']
['Saint Lucia' 'Samoa' 'Monaco']
['Saint Vincent and the Grenadines' 'Costa Rica' 'Monaco']
['Samoa' 'Saint Lucia' 'Monaco']
['San Marino' 'Taiwan' 'Monaco']
['Sao Tome and Principe' 'Comoros' 'Monaco']
['Saudi Arabia' 'Greece' 'Monaco']
['Senegal' 'Haiti' 'Monaco']
['Serbia' 'Turkey' 'Monaco']
```

```
['Seychelles' 'Uruguay' 'Monaco']
['Sierra Leone' 'Benin' 'Monaco']
['Singapore' 'Slovakia' 'Monaco']
['Slovakia' 'Singapore' 'Monaco']
['Slovenia' 'Spain' 'Monaco']
['Solomon Islands' 'Gabon' 'Monaco']
['Somalia' 'Mali' 'Monaco']
['South Africa' 'Honduras' 'Monaco']
['South Sudan' 'Afghanistan' 'Monaco']
['Spain' 'Palau' 'Monaco']
['Sri Lanka' 'Mongolia' 'Monaco']
['Sudan' 'Laos' 'Monaco']
['Suriname' 'Maldives' 'Monaco']
['Swaziland' 'Lebanon' 'Monaco']
['Sweden' 'United Kingdom' 'Monaco']
['Switzerland' 'Australia' 'South Sudan']
['Syria' 'Estonia' 'Monaco']
['Taiwan' 'San Marino' 'Monaco']
['Tajikistan' 'Vietnam' 'Monaco']
['Tanzania' 'Djibouti' 'Monaco']
['Thailand' 'Ecuador' 'Monaco']
['Timor-Leste' 'Morocco' 'Monaco']
['Togo' 'Bangladesh' 'Monaco']
['Tonga' 'Bulgaria' 'Monaco']
['Trinidad and Tobago' 'Lithuania' 'Monaco']
['Tunisia' 'Nauru' 'Monaco']
['Turkey' 'Macedonia' 'Monaco']
['Turkmenistan' 'Samoa' 'Monaco']
['Tuvalu' 'Netherlands' 'Monaco']
['Uganda' 'Rwanda' 'Monaco']
['Ukraine' 'Dominican Republic' 'Monaco']
['United Arab Emirates' 'Israel' 'Monaco']
['United Kingdom' 'Netherlands' 'Monaco']
['United States' 'Liechtenstein' 'South Sudan']
['Uruguay' 'Croatia' 'Monaco']
['Uzbekistan' 'Mauritius' 'Monaco']
['Vanuatu' 'Burma' 'Monaco']
['Venezuela' 'Montenegro' 'Monaco']
['Vietnam' 'Tajikistan' 'Monaco']
['Western Sahara' 'Namibia' 'Monaco']
['Yemen' 'Cameroon' 'Monaco']
['Zambia' "Cote d'Ivoire" 'Monaco']
['Zimbabwe' 'Papua New Guinea' 'Monaco']]
```