Kinney_DSC550_Final

January 26, 2020

0.1 D. Kinney DSC 550 Final Project

0.1.1 Part 1: Graph Analysis

---

[1]:
```
import warnings
warnings.filterwarnings("ignore")
```

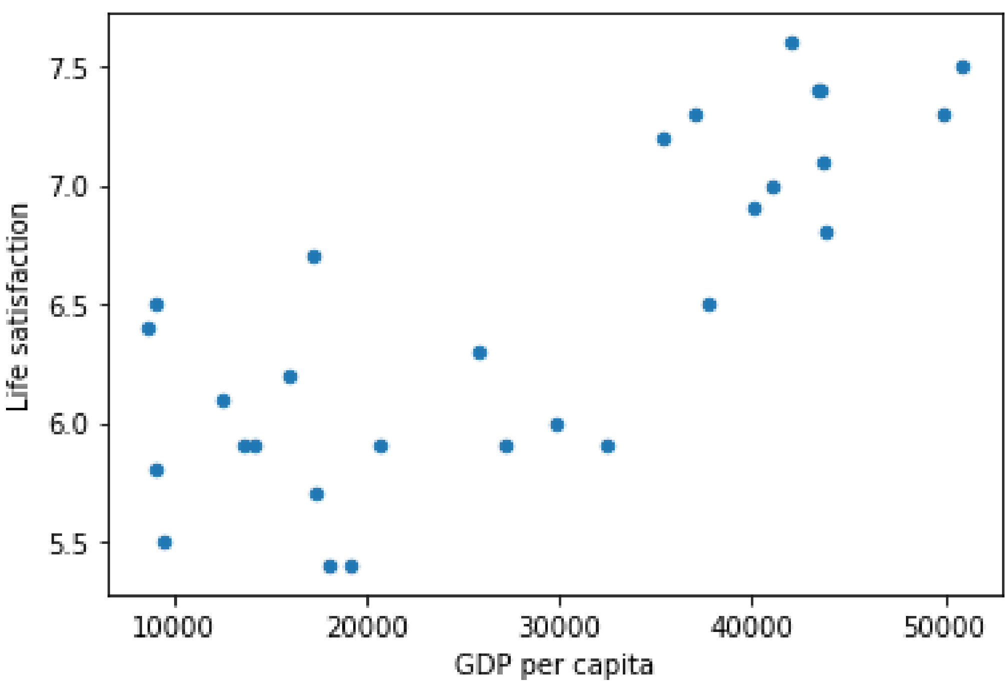
[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model
import statsmodels.formula.api as smf

from pandas.plotting import scatter_matrix
from plotnine import *

pd.set_option('display.max_columns', None)

%matplotlib inline
```

[3]:
```
def prepare_country_stats(oecd_bli, gdp_per_capita):
    oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli = oecd_bli.pivot(index="Country", columns="Indicator",␣
 ↪values="Value")
    gdp_per_capita.rename(columns={"2015": "GDP per capita"}, inplace=True)
    gdp_per_capita.set_index("Country", inplace=True)
    full_country_stats = pd.merge(left=oecd_bli, right=gdp_per_capita,
                                  left_index=True, right_index=True)
    full_country_stats.sort_values(by="GDP per capita", inplace=True)
    remove_indices = [0, 1, 6, 8, 33, 34, 35]
    keep_indices = list(set(range(36)) - set(remove_indices))
    return full_country_stats[["GDP per capita", 'Life satisfaction']].
 ↪iloc[keep_indices]
```

Step 1: Load data into dataframe

[4]:

```python
# Load the data
oecd_bli = pd.read_csv("data/oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("data/gdp_per_capita.
↪csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus' GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```



```
[[6.25984414]]
```

Step 2: check the dimension of the table

[5]:
```python
print("The dimension of the table is: ", oecd_bli.shape)
```

```
The dimension of the table is:  (2368, 17)
```

Step 3: Look at the data

[6]:
```python
print(oecd_bli.head(5))
```

```
  LOCATION         Country INDICATOR                  Indicator MEASURE  \
0      AUS       Australia   JE_LMIS  Labour market insecurity       L
1      AUT         Austria   JE_LMIS  Labour market insecurity       L
2      BEL         Belgium   JE_LMIS  Labour market insecurity       L
3      CAN          Canada   JE_LMIS  Labour market insecurity       L
4      CZE  Czech Republic   JE_LMIS  Labour market insecurity       L

  Measure INEQUALITY Inequality Unit Code        Unit  PowerCode Code  \
0   Value        TOT      Total        PC  Percentage               0
1   Value        TOT      Total        PC  Percentage               0
2   Value        TOT      Total        PC  Percentage               0
3   Value        TOT      Total        PC  Percentage               0
4   Value        TOT      Total        PC  Percentage               0

  PowerCode  Reference Period Code  Reference Period  Value  Flag Codes  Flags
0     Units                    NaN               NaN    5.4         NaN    NaN
1     Units                    NaN               NaN    3.5         NaN    NaN
2     Units                    NaN               NaN    3.7         NaN    NaN
3     Units                    NaN               NaN    6.0         NaN    NaN
4     Units                    NaN               NaN    3.1         NaN    NaN
```

[7]:
```python
oecd_bli.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2368 entries, 0 to 2367
Data columns (total 17 columns):
LOCATION                 2368 non-null object
Country                  2368 non-null object
INDICATOR                2368 non-null object
Indicator                2368 non-null object
MEASURE                  2368 non-null object
Measure                  2368 non-null object
INEQUALITY               2368 non-null object
Inequality               2368 non-null object
Unit Code                2368 non-null object
Unit                     2368 non-null object
```

```
PowerCode Code          2368 non-null int64
PowerCode               2368 non-null object
Reference Period Code   0 non-null float64
Reference Period        0 non-null float64
Value                   2368 non-null float64
Flag Codes              0 non-null float64
Flags                   0 non-null float64
dtypes: float64(5), int64(1), object(11)
memory usage: 314.6+ KB
```

Looking at the results of the "info" method, there are a number of empty columns that can be removed. There are also some with the same value throughout.

[8]:
```
# Remove empty columns
oecd_bli.dropna(axis=1, inplace=True)

# Looks liks some other variables have the same value from top to bottom,
# so really don't need them...
print(oecd_bli['MEASURE'].value_counts())
print(oecd_bli['PowerCode Code'].value_counts())
oecd_bli.drop(['MEASURE', 'Measure', 'PowerCode Code'], axis = 1, inplace=True)

# Remove space from Unit Code
oecd_bli.rename(columns={'Unit Code': 'UnitCode'})

print(oecd_bli.info())
```

```
L    2368
Name: MEASURE, dtype: int64
0    2368
Name: PowerCode Code, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2368 entries, 0 to 2367
Data columns (total 10 columns):
LOCATION      2368 non-null object
Country       2368 non-null object
INDICATOR     2368 non-null object
Indicator     2368 non-null object
INEQUALITY    2368 non-null object
Inequality    2368 non-null object
Unit Code     2368 non-null object
Unit          2368 non-null object
PowerCode     2368 non-null object
Value         2368 non-null float64
dtypes: float64(1), object(9)
memory usage: 185.1+ KB
None
```

[9]:
```
oecd_bli.sample(10)
```

[9]:
```
     LOCATION          Country INDICATOR  \
1948      GBR   United Kingdom   SW_LIFS
1916      CHE      Switzerland   SW_LIFS
1591      LVA           Latvia    HS_LEB
1061      CZE   Czech Republic   ES_STCS
2086      SVN         Slovenia   PS_REPH
547       CHE      Switzerland   JE_EMPL
1316      CHL            Chile  ES_EDUEX
192       BRA           Brazil   CG_SENG
1795      LUX       Luxembourg   HS_SFRH
1939      ISL          Iceland   SW_LIFS

                                        Indicator INEQUALITY Inequality  \
1948                            Life satisfaction        HGH       High
1916                            Life satisfaction        WMN      Women
1591                              Life expectancy         MN        Men
1061                               Student skills         MN        Men
2086                                Homicide rate        WMN      Women
547                               Employment rate         MN        Men
1316                            Years in education        WMN      Women
192   Stakeholder engagement for developing regulations        TOT      Total
1795                          Self-reported health         LW        Low
1939                            Life satisfaction        HGH       High

     Unit Code           Unit PowerCode  Value
1948   AVSCORE  Average score     Units    7.1
1916   AVSCORE  Average score     Units    7.5
1591        YR          Years     Units   69.8
1061   AVSCORE  Average score     Units  489.0
2086     RATIO          Ratio     Units    0.7
547         PC     Percentage     Units   84.0
1316        YR          Years     Units   17.7
192    AVSCORE  Average score     Units    2.2
1795        PC     Percentage     Units   65.0
1939   AVSCORE  Average score     Units    8.0
```

Using pandas.pivot_table, transform dataframe into a more human-friendly format…

[10]:
```
df_table = pd.pivot_table(oecd_bli, values='Value', index='Country',␣
 ↪columns=['INDICATOR'])

# I also need a 'Country' column. I know this is probably not the right way to␣
 ↪go about this...
df_table['country'] = df_table.index.astype('str')
```

```python
# Drop this row, it's not a country...
indexNames = df_table[df_table['country'] == 'OECD - Total'].index
df_table.drop(indexNames , inplace=True)

print(df_table.sample(5))

# For reference, create a dictionary of Indicators
print("LIST OF INDICATOR KEYS AND DESCRIPTIONS")
print("======================================")
df_indicators = oecd_bli.groupby('INDICATOR')['Indicator'].agg('min')
print(df_indicators.sort_values())
```

```
INDICATOR        CG_SENG  CG_VOTO  EQ_AIRP   EQ_WATER    ES_EDUA   ES_EDUEX  \
Country
Australia            2.7     91.0      5.0  92.666667  81.000000  20.966667
New Zealand          2.5     80.0      5.0  89.000000  78.666667  17.700000
Slovak Republic      3.0     60.0     21.0  84.666667  91.333333  15.766667
Spain                1.8     70.0     11.0  72.333333  59.000000  17.900000
Turkey               1.5     86.0     20.0  65.000000  39.000000  18.300000

INDICATOR        ES_STCS  HO_BASE  HO_HISH  HO_NUMR     HS_LEB  HS_SFRH  \
Country
Australia          411.2      NaN     20.0      NaN  82.500000    87.25
New Zealand        506.2      NaN     26.0      2.4  81.700000    89.25
Slovak Republic    463.4      1.2     23.0      1.1  77.266667    68.60
Spain              492.4      0.1     21.0      1.9  83.400000    74.00
Turkey             426.8      8.0     20.0      1.0  78.000000    70.40

INDICATOR        IW_HADI   IW_HNFW     JE_EMPL  JE_LMIS   JE_LTUR  JE_PEARN  \
Country
Australia        32759.0  427064.0  73.000000    5.922  1.306667   49126.0
New Zealand          NaN  388514.0  77.000000    4.700  0.736667   40043.0
Slovak Republic  20474.0       NaN  66.000000   21.376  4.773333   24328.0
Spain            23999.0  373548.0  62.333333   23.792  7.710000   38507.0
Turkey               NaN       NaN  51.666667   12.060  2.660000       NaN

INDICATOR        PS_FSAFEN   PS_REPH  SC_SNTWS  SW_LIFS    WL_EWLH    WL_TNOW  \
Country
Australia        64.133333  1.100000     95.25    7.350  12.840000  14.350000
New Zealand      66.266667  1.300000     96.25    7.300  15.036667  14.883333
Slovak Republic  63.700000  0.800000     91.50    6.425   4.073333        NaN
Spain            82.166667  0.600000     92.75    6.225   3.963333  15.860000
Turkey           59.833333  1.366667     86.00    5.520  31.043333  14.653333

INDICATOR                country
Country
Australia              Australia
```

```
New Zealand          New Zealand
Slovak Republic  Slovak Republic
Spain                      Spain
Turkey                    Turkey
LIST OF INDICATOR KEYS AND DESCRIPTIONS
=======================================
INDICATOR
EQ_AIRP                                            Air pollution
HO_BASE                     Dwellings without basic facilities
ES_EDUA                                Educational attainment
WL_EWLH                      Employees working very long hours
JE_EMPL                                       Employment rate
PS_FSAFEN                   Feeling safe walking alone at night
PS_REPH                                         Homicide rate
IW_HADI               Household net adjusted disposable income
IW_HNFW                                  Household net wealth
HO_HISH                                   Housing expenditure
JE_LMIS                               Labour market insecurity
HS_LEB                                        Life expectancy
SW_LIFS                                     Life satisfaction
JE_LTUR                           Long-term unemployment rate
JE_PEARN                                    Personal earnings
SC_SNTWS                          Quality of support network
HO_NUMR                                      Rooms per person
HS_SFRH                                  Self-reported health
CG_SENG      Stakeholder engagement for developing regulations
ES_STCS                                        Student skills
WL_TNOW              Time devoted to leisure and personal care
CG_VOTO                                         Voter turnout
EQ_WATER                                        Water quality
ES_EDUEX                                   Years in education
Name: Indicator, dtype: object
```

[11]:

```
print("Describe Data")
print(df_table.describe())
```

```
Describe Data
INDICATOR    CG_SENG   CG_VOTO    EQ_AIRP   EQ_WATER    ES_EDUA   ES_EDUEX  \
count      38.000000  40.00000  40.000000  40.000000  39.000000  39.000000
mean        2.160526  69.57500  13.325000  82.333333  77.717949  17.547863
std         0.577291  12.21157   5.770782  10.492977  15.136134   1.412720
min         1.200000  47.00000   3.000000  55.333333  37.666667  14.100000
25%         1.725000  60.75000   9.750000  74.250000  75.000000  16.550000
50%         2.200000  69.50000  14.000000  83.833333  82.000000  17.666667
75%         2.575000  79.00000  16.500000  91.083333  87.833333  18.350000
max         3.200000  91.00000  28.000000  98.666667  94.000000  20.966667
```

```
INDICATOR      ES_STCS    HO_BASE    HO_HISH    HO_NUMR     HS_LEB    HS_SFRH  \
count        39.000000  37.000000  38.000000  37.000000  40.000000  37.000000
mean        485.707692   5.075676  20.657895   1.632432  79.567500  67.493243
std          33.787972   8.448320   2.528500   0.431441   4.669642  14.331584
min         398.200000   0.000000  15.000000   0.900000  57.500000  33.000000
25%         475.800000   0.300000  19.000000   1.200000  77.916667  60.800000
50%         492.800000   0.900000  21.000000   1.600000  81.366667  70.200000
75%         506.800000   6.700000  22.750000   1.900000  82.366667  76.000000
max         528.800000  37.000000  26.000000   2.600000  84.066667  89.250000

INDICATOR         IW_HADI         IW_HNFW    JE_EMPL    JE_LMIS    JE_LTUR  \
count           29.000000       27.000000  40.000000  33.000000  38.000000
mean         27807.310345   289780.185185  68.533333   7.706970   2.855789
std           7055.262661   165673.432787   7.882253   6.234572   3.622899
min          16275.000000    70160.000000  43.333333   0.662000   0.050000
25%          21453.000000   180100.000000  65.833333   4.392000   1.011667
50%          29333.000000   259667.000000  69.666667   5.396000   1.776667
75%          31304.000000   379777.000000  74.000000   8.784000   3.196667
max          45284.000000   769053.000000  85.666667  29.200000  16.643333

INDICATOR        JE_PEARN  PS_FSAFEN    PS_REPH   SC_SNTWS    SW_LIFS  \
count           35.000000  40.000000  40.000000  40.000000  40.000000
mean         39817.514286  68.463333   3.481667  90.193333   6.577208
std          13108.329748  13.960934   6.459861   4.384954   0.762724
min          15314.000000  35.866667   0.166667  78.333333   4.700000
25%          25971.500000  60.108333   0.600000  88.300000   5.938333
50%          40863.000000  70.483333   0.950000  91.350000   6.510000
75%          49400.500000  78.500000   2.166667  93.062500   7.243750
max          63062.000000  90.033333  27.000000  98.000000   7.660000

INDICATOR      WL_EWLH    WL_TNOW
count        38.000000  22.000000
mean          7.789649  15.048939
std           7.585983   0.672978
min           0.140000  13.826667
25%           3.150833  14.560833
50%           4.981667  14.885000
75%          10.571667  15.600833
max          31.043333  16.336667
```

[12]:
```
corr_matrix = df_table.corr()
corr_matrix["SW_LIFS"].sort_values(ascending=False)
```

[12]:
```
INDICATOR
SW_LIFS     1.000000
JE_PEARN    0.731418
IW_HADI     0.713008
EQ_WATER    0.682587
```

```
JE_EMPL      0.678344
SC_SNTWS     0.667896
HS_SFRH      0.656817
PS_FSAFEN    0.600163
HO_NUMR      0.597502
HS_LEB       0.568044
CG_VOTO      0.368598
ES_EDUEX     0.324655
ES_EDUA      0.293395
IW_HNFW      0.292887
HO_HISH      0.286334
WL_TNOW      0.199424
ES_STCS      0.197223
CG_SENG      0.180861
WL_EWLH     -0.195136
PS_REPH     -0.259378
JE_LMIS     -0.452874
HO_BASE     -0.528167
EQ_AIRP     -0.551376
JE_LTUR     -0.567002
Name: SW_LIFS, dtype: float64
```

Step 4: Think about some questions that might help you predict what indicators most influence the Life Satisfaction score: The central point of this dataset is the so-called, **"Life Satisfaction Index"**. In other words, do indicators in the categories of housing, income, jobs, community, education, environment, civic engagement, health, etc. really lead to a better, more satisfied life? Let's focus on a few high-level categories to see how the indicators correlate with the LSI...

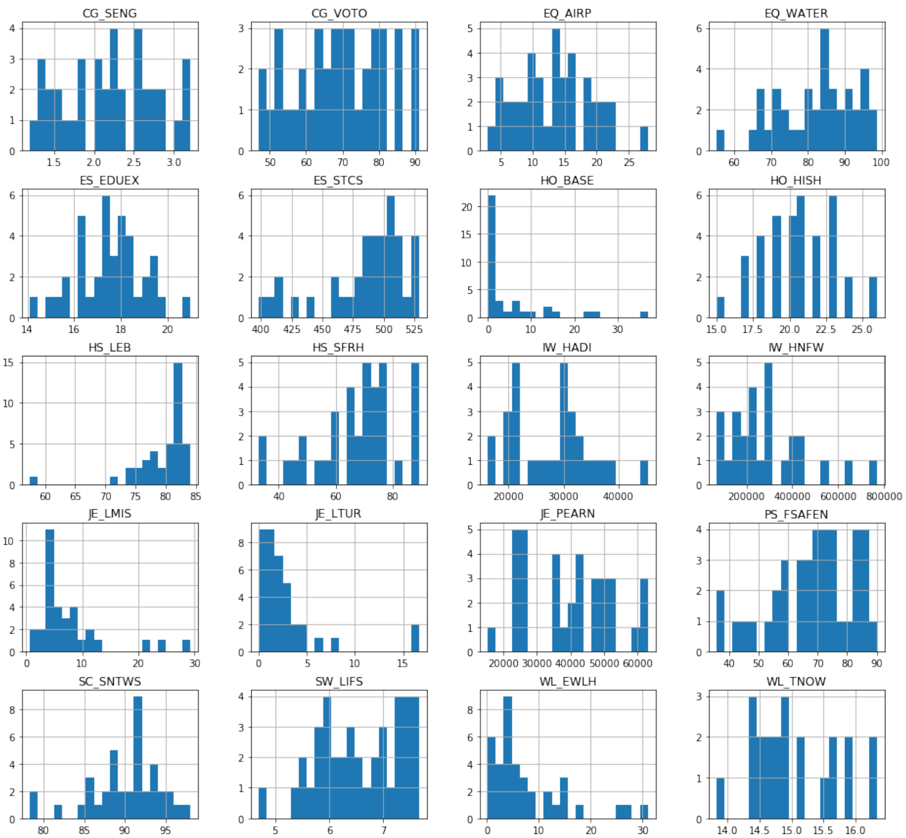
- **Wealth** Net Wealth, Labor Market Insecurity, Employment rate
- **Environment** Air pollution, Homicide rate, Water quality
- **Health** Life expectancy, Self-reported health, Long work hours

My observations are based on the **graph analysis** below.

First, there does not seem to be any noticeable normal distribution amongst any of the indicators, although some–such as HS_LEB (Life Expectancy) exhibit *normal-ish* distribution on a skewed scale.

- **Wealth** - somewhat surprisingly, Net Wealth does not appear to be as important as labor market security and the employment rate. Having said that, removing the data points above $500,000 might tell a different story.
- **Environment** - Air and water quality seem to factor higher than the homicide rate, which shows almost no effect on the LSI.
- **Health** - Life expectancy seems like an obvious factor, but I was also satisfied to see long work hours affect the index as well.

[13]:
```
df_table.hist( bins = 20, figsize =( 20,15))
plt.show()
```



[14]:
```python
results = smf.ols('SW_LIFS ~ IW_HNFW + JE_LMIS + JE_EMPL', data=df_table).fit()
print("Money: Net Wealth, Labor Market Insecurity, Employment rate")
print(results.summary())
```

```
Money: Net Wealth, Labor Market Insecurity, Employment rate
                            OLS Regression Results
==============================================================================
Dep. Variable:                SW_LIFS   R-squared:                       0.327
Model:                            OLS   Adj. R-squared:                  0.235
Method:                 Least Squares   F-statistic:                     3.564
Date:                Sun, 26 Jan 2020   Prob (F-statistic):             0.0306
Time:                        09:45:22   Log-Likelihood:                -22.710
No. Observations:                  26   AIC:                             53.42
Df Residuals:                      22   BIC:                             58.45
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.7241      2.217      1.229      0.232      -1.873       7.321
IW_HNFW      1.13e-06    7.6e-07      1.486      0.151   -4.47e-07    2.71e-06
```

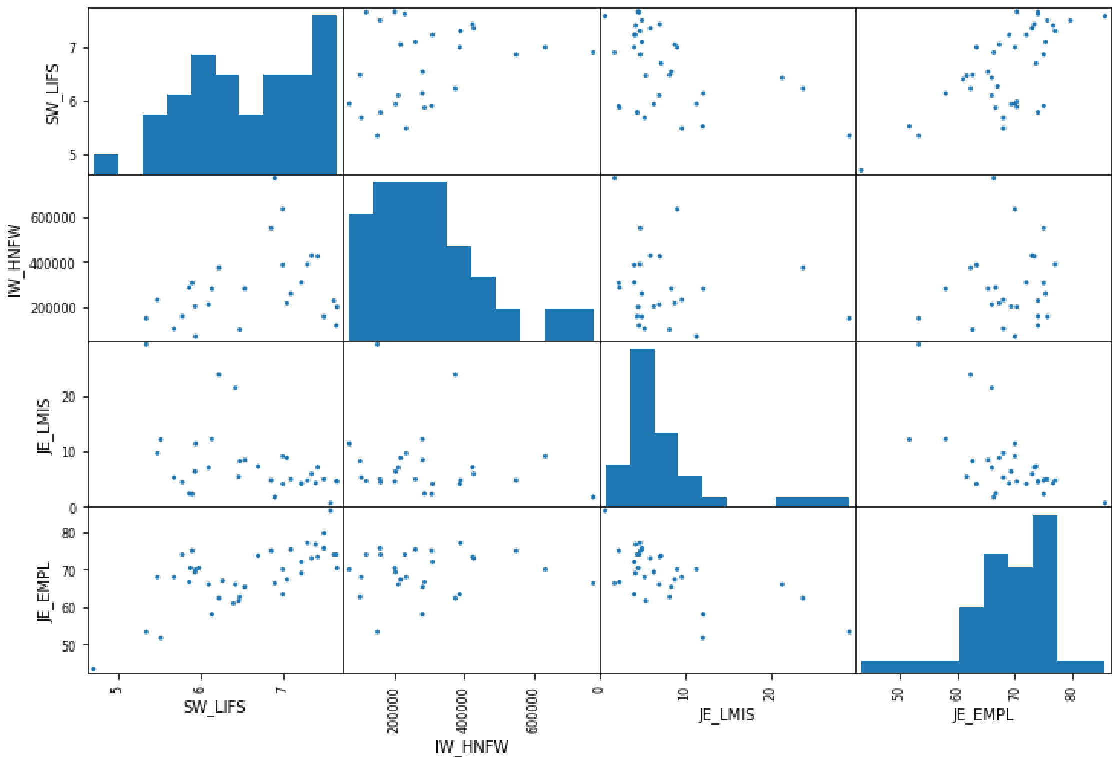
```
JE_LMIS       -0.0085      0.028     -0.304      0.764      -0.067       0.050
JE_EMPL        0.0523      0.030      1.763      0.092      -0.009       0.114
==============================================================================
Omnibus:                        0.953   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.621   Jarque-Bera (JB):                0.831
Skew:                          -0.167   Prob(JB):                        0.660
Kurtosis:                       2.190   Cond. No.                     6.02e+06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.02e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[15]:

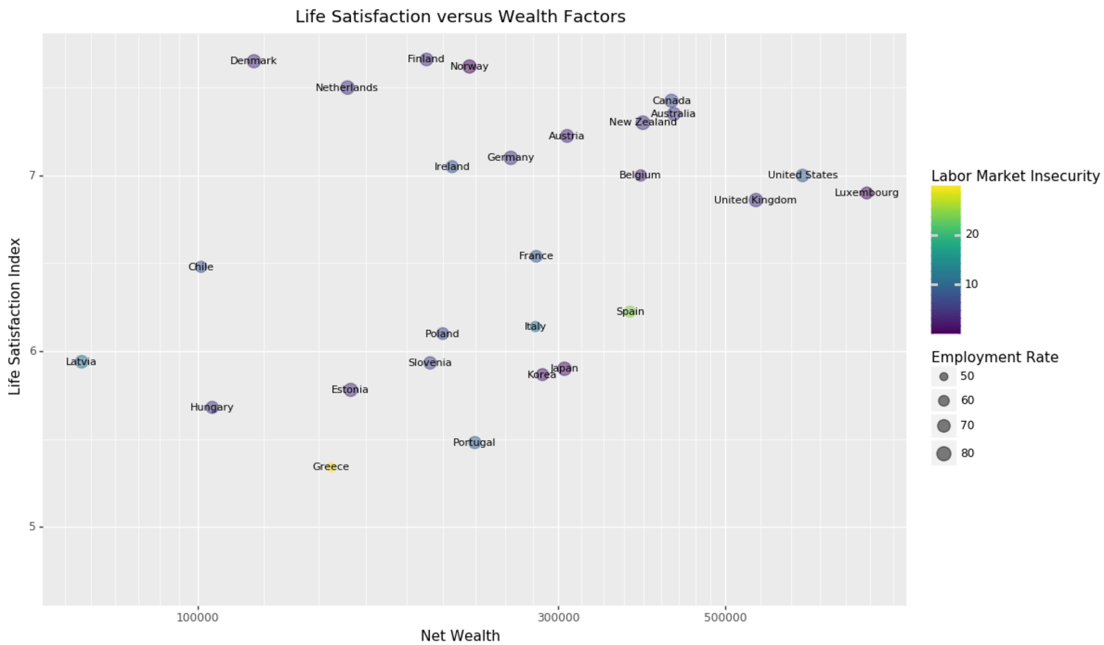
```python
attributes = ['SW_LIFS', 'IW_HNFW', 'JE_LMIS', 'JE_EMPL']
scatter_matrix(df_table[attributes], alpha=1.0, figsize=(12, 8))
```

[15]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6B1179A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6B0E66D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A9D3B50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A9FB3D0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AA1CC10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AA4B490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AA563A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AA81C40>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AAD2D90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AB07610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AB32E50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AB666D0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AB91F10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6ABC5790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6ABF1FD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AC26850>]],
      dtype=object)
```

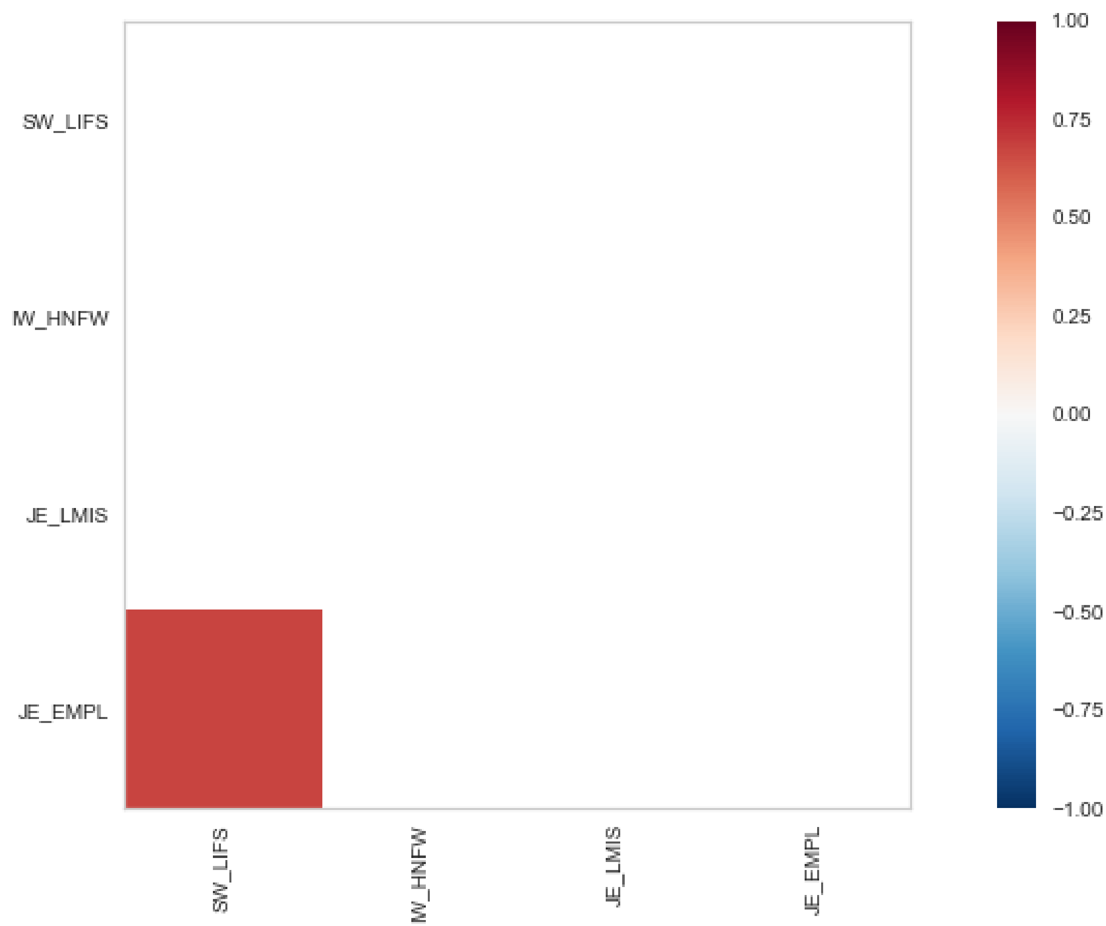


[16]:
```
(ggplot(df_table, aes(x='IW_HNFW', y='SW_LIFS', color='JE_LMIS',␣
↪size='JE_EMPL')) +
        geom_point(alpha=0.5) +
        scale_x_log10() +
        geom_text(aes(x='IW_HNFW', y='SW_LIFS', label='country'),
                  color="black",
                  size=8,
                  data=df_table) +
         theme(figure_size = (12.0, 8.0)) +
        labs(title="Life Satisfaction versus Wealth Factors",x="Net␣
↪Wealth",y="Life Satisfaction Index",size="Employment Rate",color="Labor␣
↪Market Insecurity")
)
```



[16]: <ggplot: (174595026035)>

[17]:
```
import yellowbrick
from yellowbrick.features import Rank2D
from yellowbrick.features import ParallelCoordinates
from yellowbrick.style import set_palette

#set up the figure size
plt.rcParams['figure.figsize'] = (15, 7)
num_features = ['SW_LIFS', 'IW_HNFW', 'JE_LMIS', 'JE_EMPL']
# extract the numpy arrays from the data frame
X = df_table[num_features].as_matrix()

# instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=num_features, algorithm='pearson')
visualizer.fit(X)                    # Fit the data to the visualizer
visualizer.transform(X)                 # Transform the data

plt.show()
```



[18]:
```python
results = smf.ols('SW_LIFS ~ EQ_AIRP + PS_REPH + EQ_WATER', data=df_table).fit()
print("Environment: Air pollution, Homicide rate, Water quality")
print(results.summary())
```

```
Environment: Air pollution, Homicide rate, Water quality
                            OLS Regression Results
==============================================================================
Dep. Variable:                SW_LIFS   R-squared:                       0.512
Model:                            OLS   Adj. R-squared:                  0.471
Method:                 Least Squares   F-statistic:                     12.57
Date:                Sun, 26 Jan 2020   Prob (F-statistic):           8.92e-06
Time:                        09:45:24   Log-Likelihood:                -31.080
No. Observations:                  40   AIC:                             70.16
Df Residuals:                      36   BIC:                             76.92
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

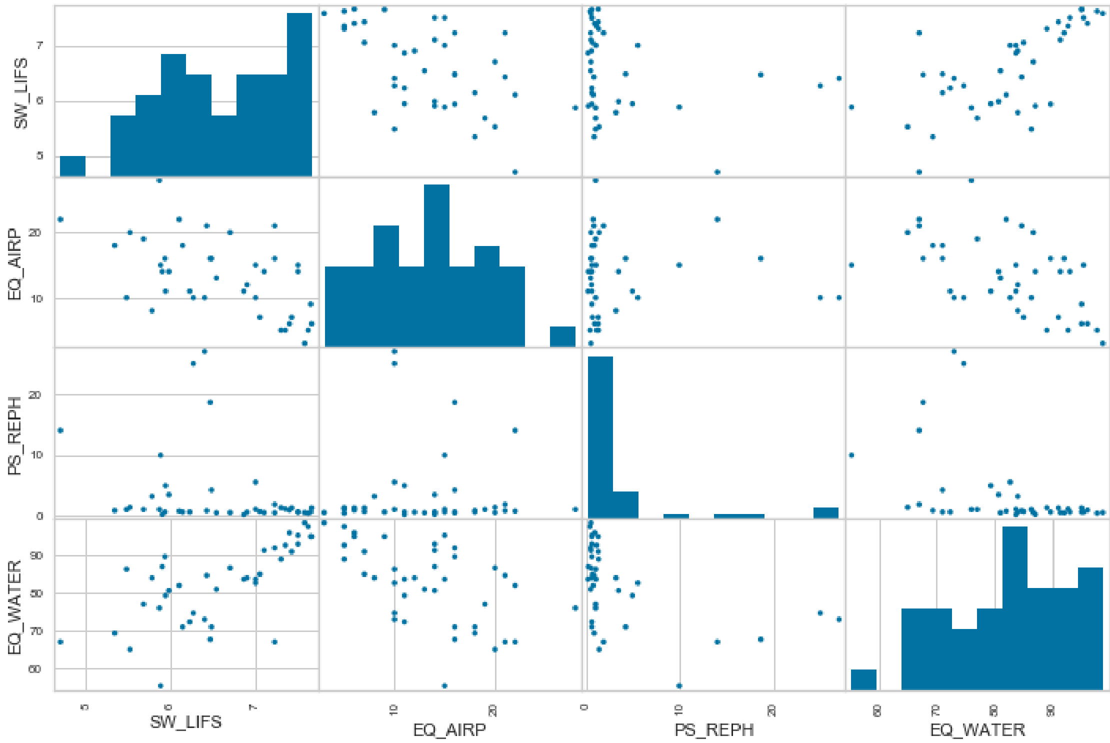
```
Intercept      3.8394      1.218      3.152      0.003      1.369      6.310
EQ_AIRP       -0.0343      0.020     -1.751      0.088     -0.074      0.005
PS_REPH       -0.0015      0.017     -0.093      0.926     -0.035      0.032
EQ_WATER       0.0389      0.012      3.193      0.003      0.014      0.064
==============================================================================
Omnibus:                        3.892   Durbin-Watson:                   1.767
Prob(Omnibus):                  0.143   Jarque-Bera (JB):                2.842
Skew:                          -0.368   Prob(JB):                        0.241
Kurtosis:                       4.079   Cond. No.                     1.17e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.17e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[19]:

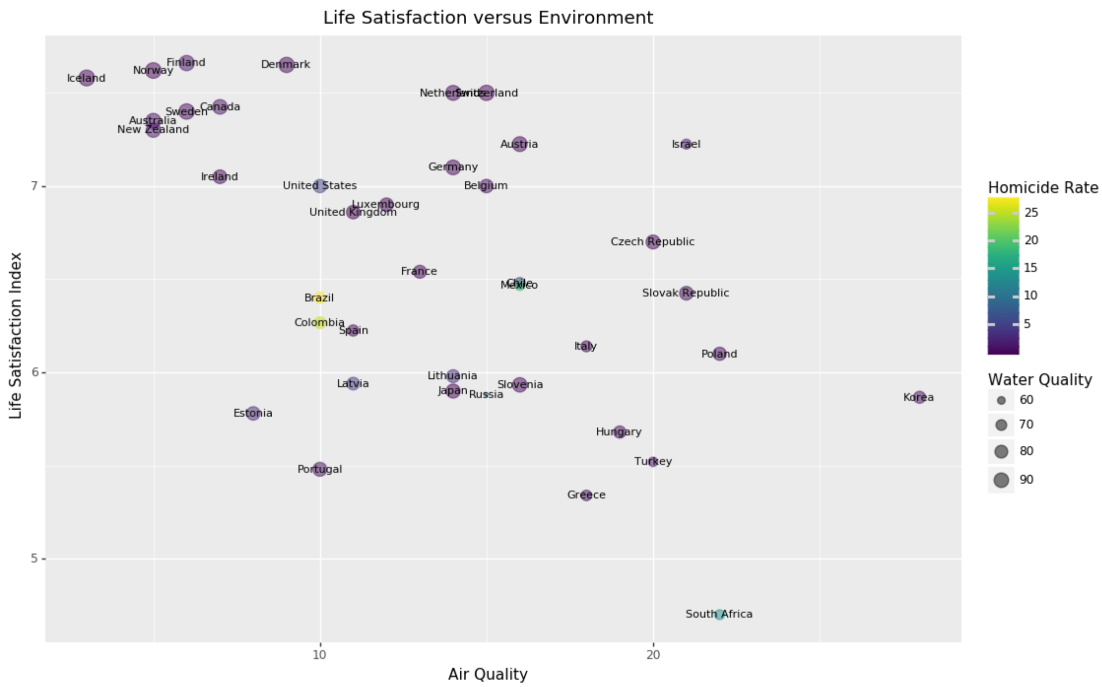
```
attributes = ['SW_LIFS', 'EQ_AIRP', 'PS_REPH', 'EQ_WATER']
scatter_matrix(df_table[attributes], alpha=1.0, figsize=(12, 8))
```

[19]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A85F520>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A6A38B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A73BE20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A797B20>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A6D05B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6B00C6D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6B00CDC0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A71DB50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A6394C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A3EA250>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A6570D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A6EF550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A247C70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6B01B8B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A8FD400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A934670>]],
      dtype=object)
```

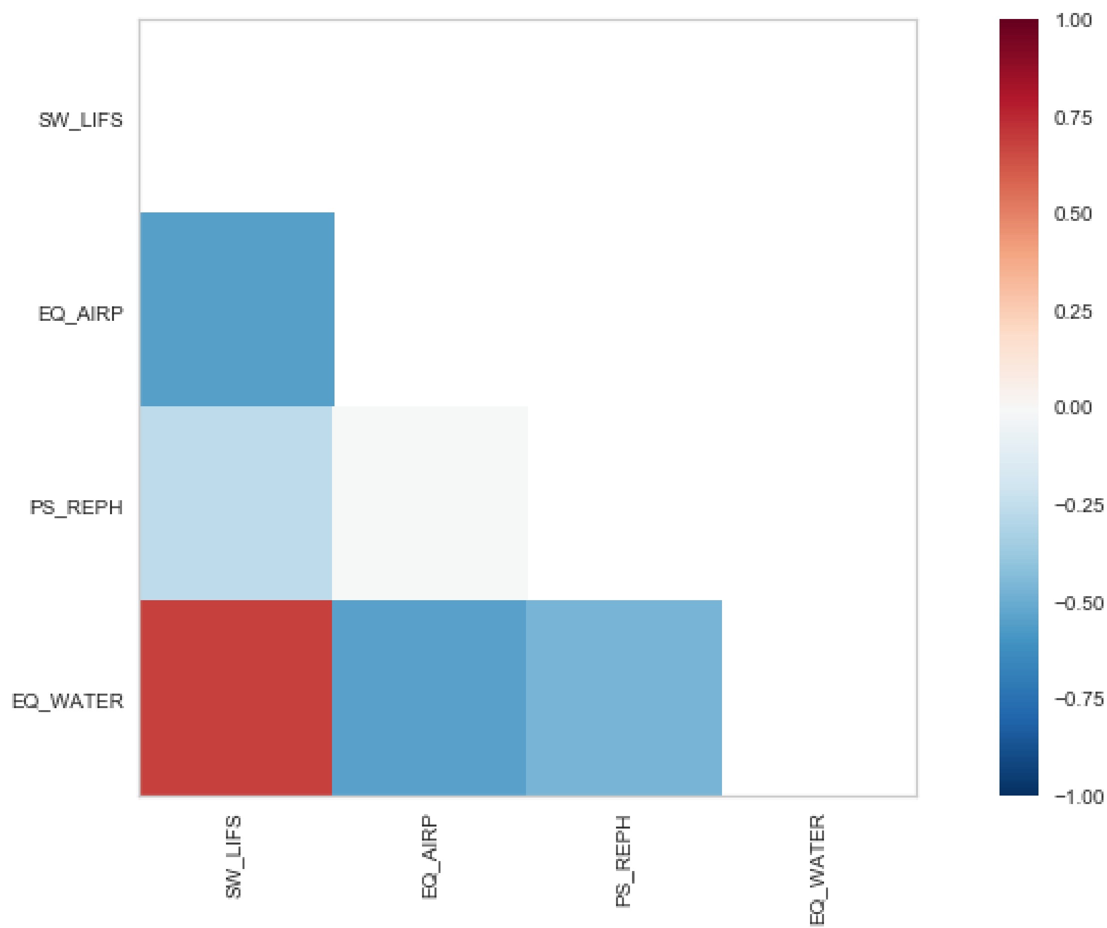


[20]:
```
(ggplot(df_table, aes(x='EQ_AIRP', y='SW_LIFS', color='PS_REPH',␣
↪size='EQ_WATER')) +
        geom_point(alpha = 0.5) +
         geom_text(aes(x='EQ_AIRP', y='SW_LIFS', label='country'),
                  color="black",
                  size=8,
                  data=df_table) +
        theme(figure_size = (12.0, 8.0)) +
        labs(title="Life Satisfaction versus Environment",x="Air␣
↪Quality",y="Life Satisfaction Index",size="Water Quality",color="Homicide␣
↪Rate")
)
```



[20]: <ggplot: (174595291127)>

[21]:
```
#set up the figure size
plt.rcParams['figure.figsize'] = (15, 7)
num_features = ['SW_LIFS', 'EQ_AIRP', 'PS_REPH', 'EQ_WATER']
# extract the numpy arrays from the data frame
X = df_table[num_features].as_matrix()

# instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=num_features, algorithm='pearson')
visualizer.fit(X)                    # Fit the data to the visualizer
visualizer.transform(X)              # Transform the data

plt.show()
```



[22]:
```
results = smf.ols('SW_LIFS ~ HS_LEB + HS_SFRH + WL_EWLH', data=df_table).fit()
print("Health: Life expectancy, Self-reported health, Long work hours")
print(results.summary())
```

```
Health: Life expectancy, Self-reported health, Long work hours
                            OLS Regression Results
==============================================================================
Dep. Variable:                SW_LIFS   R-squared:                       0.484
Model:                            OLS   Adj. R-squared:                  0.434
Method:                 Least Squares   F-statistic:                     9.707
Date:                Sun, 26 Jan 2020   Prob (F-statistic):           0.000114
Time:                        09:45:26   Log-Likelihood:                -26.111
No. Observations:                  35   AIC:                             60.22
Df Residuals:                      31   BIC:                             66.44
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

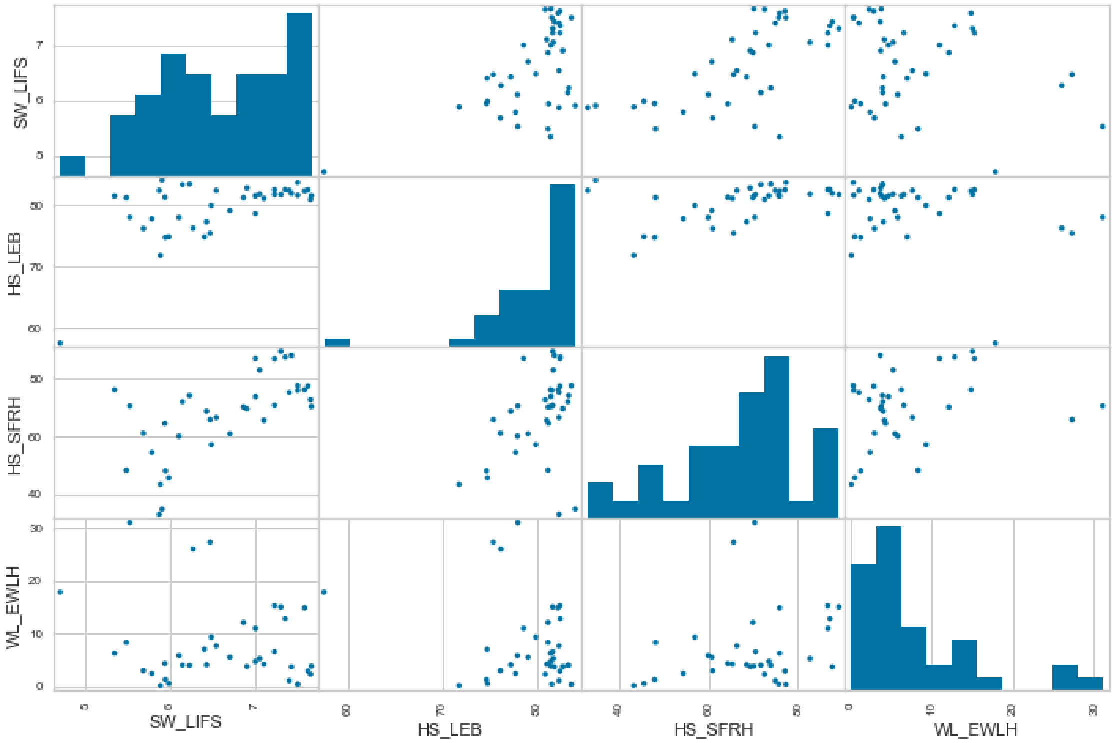
```
Intercept      2.3807      3.176      0.750      0.459     -4.096      8.858
HS_LEB         0.0227      0.046      0.499      0.621     -0.070      0.116
HS_SFRH        0.0382      0.011      3.403      0.002      0.015      0.061
WL_EWLH       -0.0250      0.014     -1.737      0.092     -0.054      0.004
==============================================================================
Omnibus:                        8.285   Durbin-Watson:                   2.096
Prob(Omnibus):                  0.016   Jarque-Bera (JB):                6.935
Skew:                          -0.902   Prob(JB):                       0.0312
Kurtosis:                       4.227   Cond. No.                     3.70e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.7e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[23]:

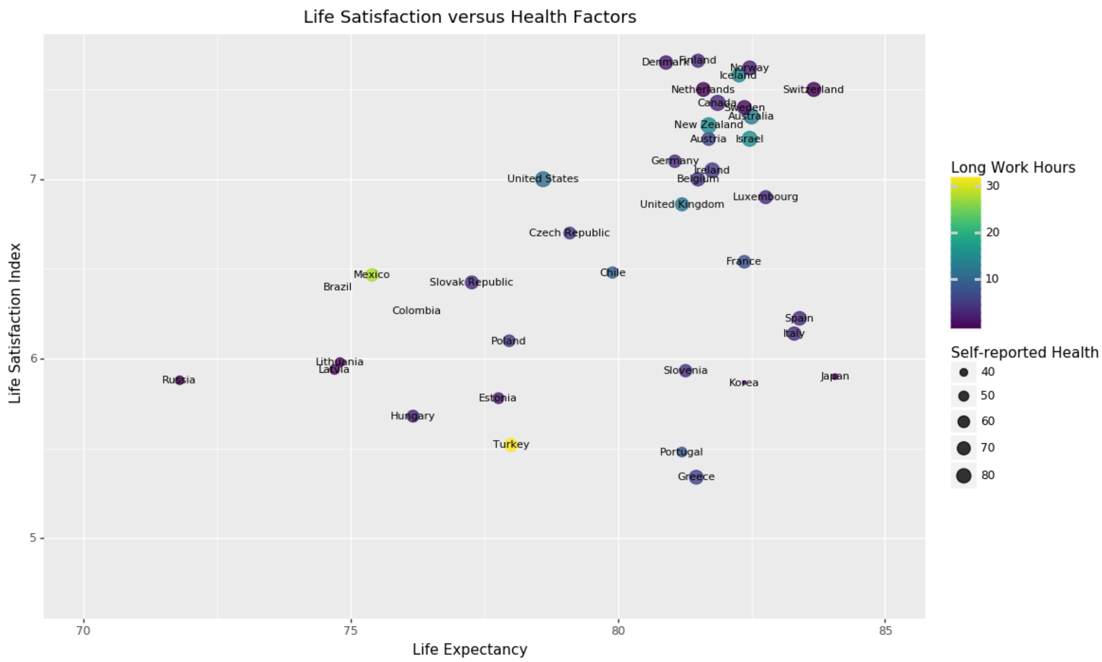
```
attributes = ['SW_LIFS', 'HS_LEB', 'HS_SFRH', 'WL_EWLH']
scatter_matrix(df_table[attributes], alpha=1.0, figsize=(12, 8))
```

[23]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A70AA30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A429EE0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A68D3D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A4C9850>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A895CD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A509040>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A5090A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A5CA4C0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A53AF10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A4383D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A434850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A7C2400>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A8B5D00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AD24FD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6A3CB790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000028A6AE55580>]],
      dtype=object)
```

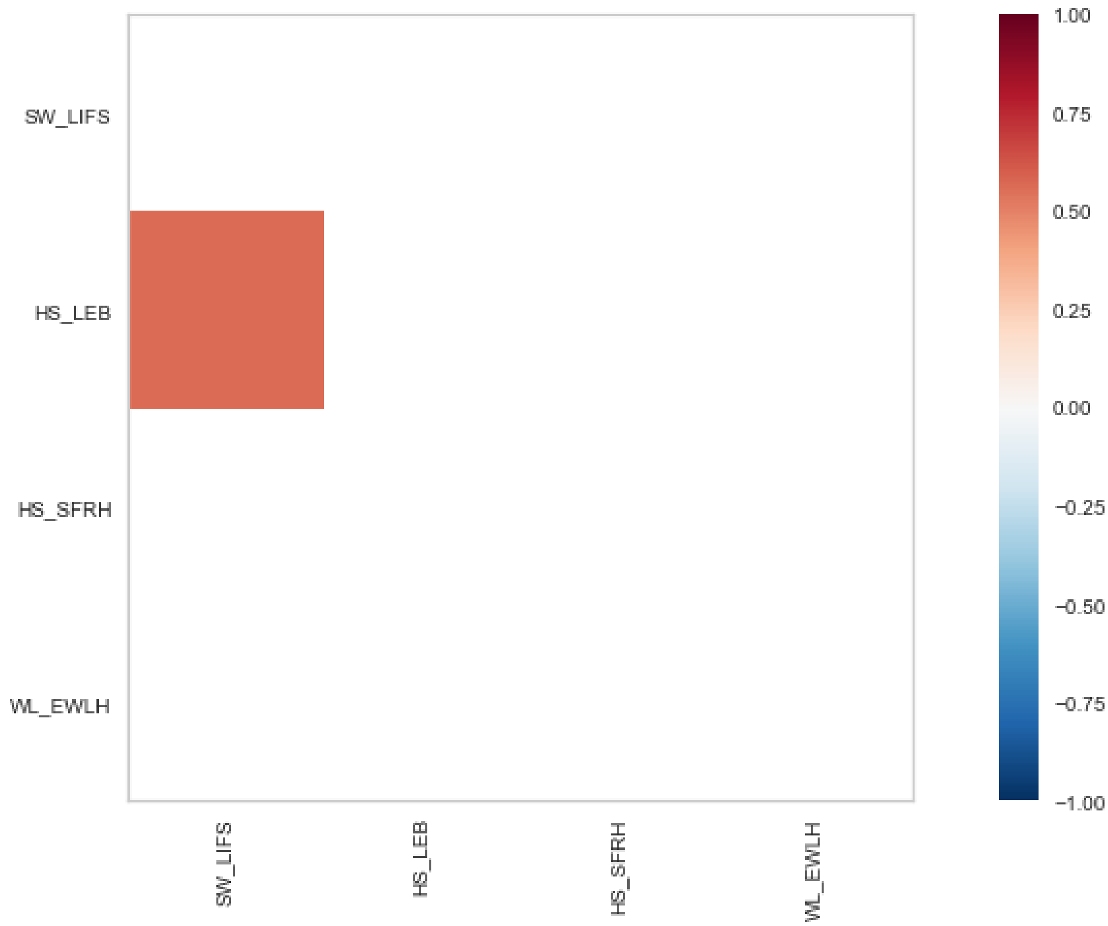


[24]:
```
(ggplot(df_table[df_table['HS_LEB'] > 70], aes(x='HS_LEB', y='SW_LIFS',␣
↪color='WL_EWLH', size='HS_SFRH')) +
        geom_point(alpha = 0.8) +
        scale_x_continuous(limits=[70,85]) +
        geom_text(aes(x='HS_LEB', y='SW_LIFS', label='country'),
                  color="black",
                  size=8,
                  data=df_table) +
        theme(figure_size = (12.0, 8.0)) +
        labs(title="Life Satisfaction versus Health Factors",x="Life␣
↪Expectancy",y="Life Satisfaction Index",size="Self-reported␣
↪Health",color="Long Work Hours")
)
```



[24]: <ggplot: (174594488278)>

[25]:
```
#set up the figure size
plt.rcParams['figure.figsize'] = (15, 7)
num_features = ['SW_LIFS', 'HS_LEB', 'HS_SFRH', 'WL_EWLH']
# extract the numpy arrays from the data frame
X = df_table[num_features].as_matrix()

# instantiate the visualizer with the Covariance ranking algorithm
visualizer = Rank2D(features=num_features, algorithm='pearson')
visualizer.fit(X)                    # Fit the data to the visualizer
visualizer.transform(X)                 # Transform the data

plt.show()
```



Part 2: Dimensionality and Feature Reduction

---

[26]:
```
# What kind of data are we dealing with?
df_table.describe()
```

[26]:

INDICATOR	CG_SENG	CG_VOTO	EQ_AIRP	EQ_WATER	ES_EDUA	ES_EDUEX	\
count	38.000000	40.00000	40.000000	40.000000	39.000000	39.000000	
mean	2.160526	69.57500	13.325000	82.333333	77.717949	17.547863	
std	0.577291	12.21157	5.770782	10.492977	15.136134	1.412720	
min	1.200000	47.00000	3.000000	55.333333	37.666667	14.100000	
25%	1.725000	60.75000	9.750000	74.250000	75.000000	16.550000	
50%	2.200000	69.50000	14.000000	83.833333	82.000000	17.666667	
75%	2.575000	79.00000	16.500000	91.083333	87.833333	18.350000	
max	3.200000	91.00000	28.000000	98.666667	94.000000	20.966667	

```
INDICATOR      ES_STCS    HO_BASE    HO_HISH    HO_NUMR     HS_LEB    HS_SFRH  \
count        39.000000  37.000000  38.000000  37.000000  40.000000  37.000000
mean        485.707692   5.075676  20.657895   1.632432  79.567500  67.493243
std          33.787972   8.448320   2.528500   0.431441   4.669642  14.331584
min         398.200000   0.000000  15.000000   0.900000  57.500000  33.000000
25%         475.800000   0.300000  19.000000   1.200000  77.916667  60.800000
50%         492.800000   0.900000  21.000000   1.600000  81.366667  70.200000
75%         506.800000   6.700000  22.750000   1.900000  82.366667  76.000000
max         528.800000  37.000000  26.000000   2.600000  84.066667  89.250000

INDICATOR        IW_HADI         IW_HNFW    JE_EMPL    JE_LMIS    JE_LTUR  \
count          29.000000       27.000000  40.000000  33.000000  38.000000
mean        27807.310345   289780.185185  68.533333   7.706970   2.855789
std          7055.262661   165673.432787   7.882253   6.234572   3.622899
min         16275.000000    70160.000000  43.333333   0.662000   0.050000
25%         21453.000000   180100.000000  65.833333   4.392000   1.011667
50%         29333.000000   259667.000000  69.666667   5.396000   1.776667
75%         31304.000000   379777.000000  74.000000   8.784000   3.196667
max         45284.000000   769053.000000  85.666667  29.200000  16.643333

INDICATOR       JE_PEARN  PS_FSAFEN    PS_REPH   SC_SNTWS    SW_LIFS  \
count          35.000000  40.000000  40.000000  40.000000  40.000000
mean        39817.514286  68.463333   3.481667  90.193333   6.577208
std         13108.329748  13.960934   6.459861   4.384954   0.762724
min         15314.000000  35.866667   0.166667  78.333333   4.700000
25%         25971.500000  60.108333   0.600000  88.300000   5.938333
50%         40863.000000  70.483333   0.950000  91.350000   6.510000
75%         49400.500000  78.500000   2.166667  93.062500   7.243750
max         63062.000000  90.033333  27.000000  98.000000   7.660000

INDICATOR    WL_EWLH    WL_TNOW
count      38.000000  22.000000
mean        7.789649  15.048939
std         7.585983   0.672978
min         0.140000  13.826667
25%         3.150833  14.560833
50%         4.981667  14.885000
75%        10.571667  15.600833
max        31.043333  16.336667
```

Proposed Steps for Dimensionality and Feature Reduction

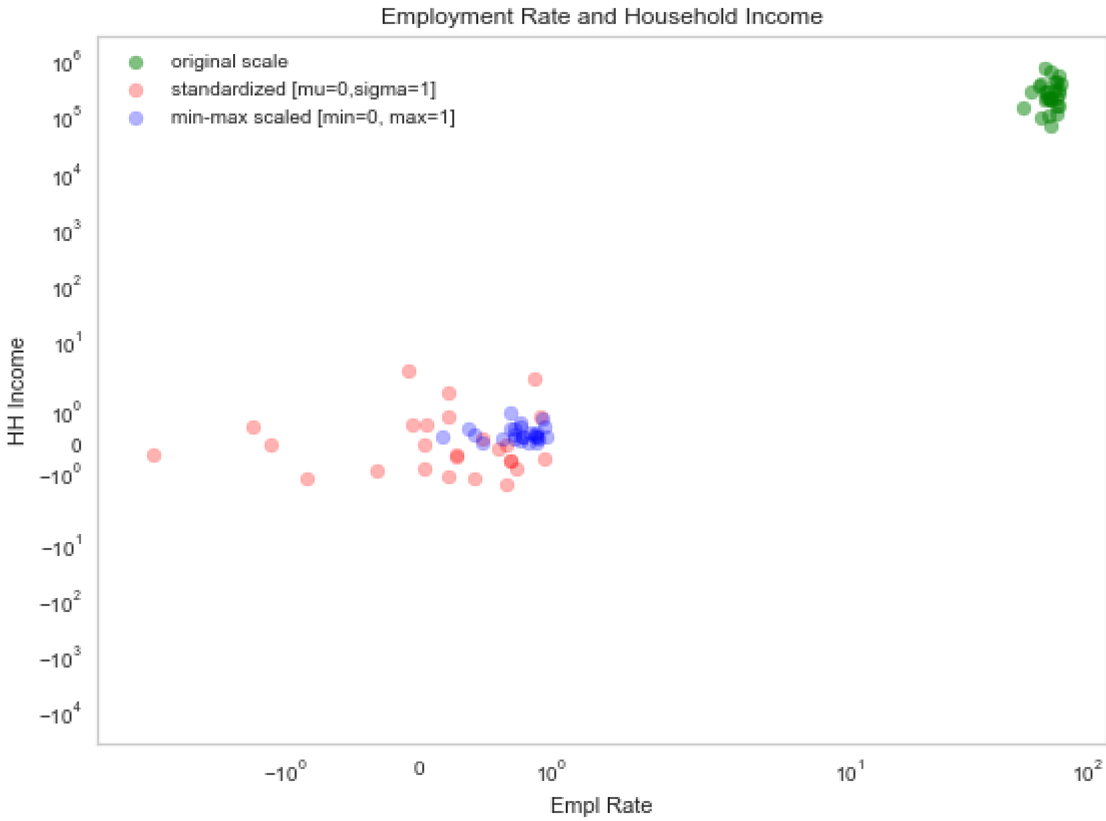
- Due to the quantitative nature of these variables, there exists wide discrepancies of units. For instance, Household Net Wealth can be in the thousands of US dollars, while Employment Rate is a percentage. I will apply a simple `min-max rescaler` to normalize all variables.
- The *variance* between features varies greatly, For example, CG_SENS ranges from 1.2 to 3.2 while SC_SNTWS ranges from 4.38 to 98. I will start by setting a variance threshold and eliminating features below the threshold. In theory, this should eliminate variables with low

variance, which likely will not contribute greatly to the model.
- I suspect that variables such as air pollution and water quality probably are highly correlated. I will apply a correlation matrix and will consider dropping one of the correlated features.
- That takes care of the easy stuff. Next I'd like to automatically select the best features to keep by leveraging scikit-learn's recursive feature elimination functionality.

VarianceThreshold

[27]:
```python
from sklearn import preprocessing

# Make a features dataset by dropping the target variable--SW_LIFS
features = df_table.drop(['country','SW_LIFS'], axis=1)
target = df_table['SW_LIFS']

# Standardizing
std_scale = preprocessing.StandardScaler().fit(features)
df_std = std_scale.transform(features)

#Min-max scaling
minmax_scale = preprocessing.MinMaxScaler().fit(features)
df_minmax = minmax_scale.transform(features)
```

[28]:
```python
def plot():
    plt.figure(figsize=(8,6))

    plt.scatter(df_table['JE_EMPL'], df_table['IW_HNFW'],
            color='green', label='original scale', alpha=0.5)

    plt.scatter(df_std[:,4], df_std[:,13], color='red',
            label='standardized [mu=0,sigma=1]', alpha=0.3)

    plt.scatter(df_minmax[:,4], df_minmax[:,13],
            color='blue', label='min-max scaled [min=0, max=1]', alpha=0.3)

    plt.title('Employment Rate and Household Income')
    plt.xlabel('Empl Rate')
    plt.ylabel('HH Income')
    plt.legend(loc='upper left')
    plt.xscale('symlog')
    plt.yscale('symlog')
    plt.grid()

    plt.tight_layout()

plot()
plt.show()
```



[29]:
```
from sklearn.feature_selection import VarianceThreshold

# Make a features dataset by dropping the target variable--SW_LIFS
features = df_table.drop(['country','SW_LIFS'], axis=1)
target = df_table['SW_LIFS']

# Create thresholder
thresholder = VarianceThreshold(threshold=5.0)

# Create high variance feature matrix
features_high_variance = thresholder.fit_transform(features)

# View high variance feature matrix
features_high_variance[0:3]

# View variances
thresholder.fit(features).variances_
```

[29]:
```
array([3.24494460e-01, 1.45394375e+02, 3.24693750e+01, 1.07350000e+02,
       2.23228139e+02, 1.94460370e+00, 1.11235456e+03, 6.94450840e+01,
       6.22506925e+00, 1.81110299e-01, 2.12604160e+01, 1.99843100e+02,
       4.80602922e+07, 2.64311054e+10, 6.05766667e+01, 3.76920085e+01,
```

```
1.27799922e+01, 1.66918929e+08, 1.90034989e+02, 4.06865528e+01,
1.87471222e+01, 5.60327426e+01, 4.32312511e-01])
```

[30]:
```python
# Create correlation matrix
corr_matrix = features.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
                         k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

# Drop features
features.drop(features.columns[to_drop], axis=1).head()
```

[30]:
```
INDICATOR  CG_SENG  CG_VOTO  EQ_AIRP   EQ_WATER    ES_EDUA   ES_EDUEX  \
Country
Australia      2.7     91.0      5.0  92.666667  81.000000  20.966667
Austria        1.3     80.0     16.0  92.000000  85.000000  17.000000
Belgium        2.0     89.0     15.0  83.666667  77.000000  19.300000
Brazil         2.2     79.0     10.0  73.000000  49.000000  16.166667
Canada         2.9     68.0      7.0  91.000000  91.333333  17.333333

INDICATOR  ES_STCS  HO_BASE  HO_HISH  HO_NUMR     HS_LEB  HS_SFRH  IW_HADI  \
Country
Australia    411.2      NaN     20.0      NaN  82.500000    87.25  32759.0
Austria      492.8      0.9     21.0      1.6  81.700000    70.60  33541.0
Belgium      503.8      1.9     21.0      2.2  81.500000    73.60  30364.0
Brazil       398.2      6.7      NaN      NaN  74.766667      NaN      NaN
Canada       523.2      0.2     22.0      2.6  81.866667    87.80  30854.0

INDICATOR   IW_HNFW    JE_EMPL  JE_LMIS   JE_LTUR  JE_PEARN  PS_FSAFEN  \
Country
Australia  427064.0  73.000000    5.922  1.306667   49126.0  64.133333
Austria    308325.0  72.000000    4.076  1.830000   50349.0  80.700000
Belgium    386006.0  63.333333    4.052  3.533333   49675.0  70.266667
Brazil          NaN  61.000000      NaN       NaN       NaN  35.866667
Canada     423849.0  73.333333    7.048  0.763333   47622.0  82.500000

INDICATOR    PS_REPH  SC_SNTWS    WL_EWLH    WL_TNOW
Country
Australia   1.100000     95.25  12.840000  14.350000
Austria     0.466667     92.00   6.590000  14.530000
Belgium     1.033333     92.00   4.703333  15.663333
Brazil     27.000000     89.25   7.006667        NaN
Canada      1.266667     93.25   3.673333  14.553333
```

[31]:
```python
import seaborn as sns
print(corr_matrix)
sns.heatmap(corr_matrix)
```

```
INDICATOR   CG_SENG   CG_VOTO   EQ_AIRP  EQ_WATER   ES_EDUA  ES_EDUEX  \
INDICATOR
CG_SENG    1.000000  0.028002  0.015333  0.052708  0.245220  0.037028
CG_VOTO    0.028002  1.000000  0.205751  0.161888  0.086394  0.301847
EQ_AIRP    0.015333  0.205751  1.000000  0.545403  0.012111  0.341326
EQ_WATER   0.052708  0.161888  0.545403  1.000000  0.401520  0.499011
ES_EDUA    0.245220  0.086394  0.012111  0.401520  1.000000  0.272450
ES_EDUEX   0.037028  0.301847  0.341326  0.499011  0.272450  1.000000
ES_STCS    0.067136  0.057214  0.072012  0.473621  0.627987  0.248241
HO_BASE    0.002465  0.231873  0.229820  0.586296  0.327014  0.500586
HO_HISH    0.070164  0.046796  0.204556  0.259928  0.052753  0.146336
HO_NUMR    0.132029  0.297270  0.599746  0.632610  0.238163  0.323064
HS_LEB     0.086465  0.147134  0.287432  0.502301  0.086281  0.314825
HS_SFRH    0.186154  0.422847  0.416346  0.295603  0.028642  0.257713
IW_HADI    0.009595  0.450701  0.384807  0.474136  0.069743  0.062532
IW_HNFW    0.313337  0.399376  0.197628  0.076700  0.030136  0.304318
JE_EMPL    0.149286  0.001314  0.525146  0.726957  0.504685  0.238336
JE_LMIS    0.046940  0.272671  0.203927  0.456311  0.236128  0.007565
JE_LTUR    0.199630  0.063474  0.284824  0.390481  0.202390  0.132076
JE_PEARN   0.019460  0.529236  0.450445  0.604712  0.218113  0.245365
PS_FSAFEN  0.042428  0.152899  0.364504  0.745049  0.449514  0.456138
PS_REPH    0.041994  0.136966  0.007333  0.464015  0.490610  0.503941
SC_SNTWS   0.043415  0.297239  0.679765  0.622848  0.276437  0.299917
WL_EWLH    0.009313  0.067688  0.115445  0.376035  0.634258  0.282485
WL_TNOW    0.302802  0.307280  0.088929  0.004498  0.337958  0.094854

INDICATOR   ES_STCS   HO_BASE   HO_HISH   HO_NUMR    HS_LEB   HS_SFRH  \
INDICATOR
CG_SENG    0.067136  0.002465  0.070164  0.132029  0.086465  0.186154
CG_VOTO    0.057214  0.231873  0.046796  0.297270  0.147134  0.422847
EQ_AIRP    0.072012  0.229820  0.204556  0.599746  0.287432  0.416346
EQ_WATER   0.473621  0.586296  0.259928  0.632610  0.502301  0.295603
ES_EDUA    0.627987  0.327014  0.052753  0.238163  0.086281  0.028642
ES_EDUEX   0.248241  0.500586  0.146336  0.323064  0.314825  0.257713
ES_STCS    1.000000  0.600704  0.126796  0.627171  0.436038  0.171635
HO_BASE    0.600704  1.000000  0.359024  0.570377  0.844605  0.480710
HO_HISH    0.126796  0.359024  1.000000  0.194369  0.277640  0.329240
HO_NUMR    0.627171  0.570377  0.194369  1.000000  0.587021  0.464200
HS_LEB     0.436038  0.844605  0.277640  0.587021  1.000000  0.404711
HS_SFRH    0.171635  0.480710  0.329240  0.464200  0.404711  1.000000
IW_HADI    0.028382  0.451797  0.095344  0.717301  0.459078  0.515910
IW_HNFW    0.045354  0.428187  0.191833  0.571859  0.374210  0.355329
JE_EMPL    0.478432  0.489324  0.215731  0.435517  0.522467  0.085938
```

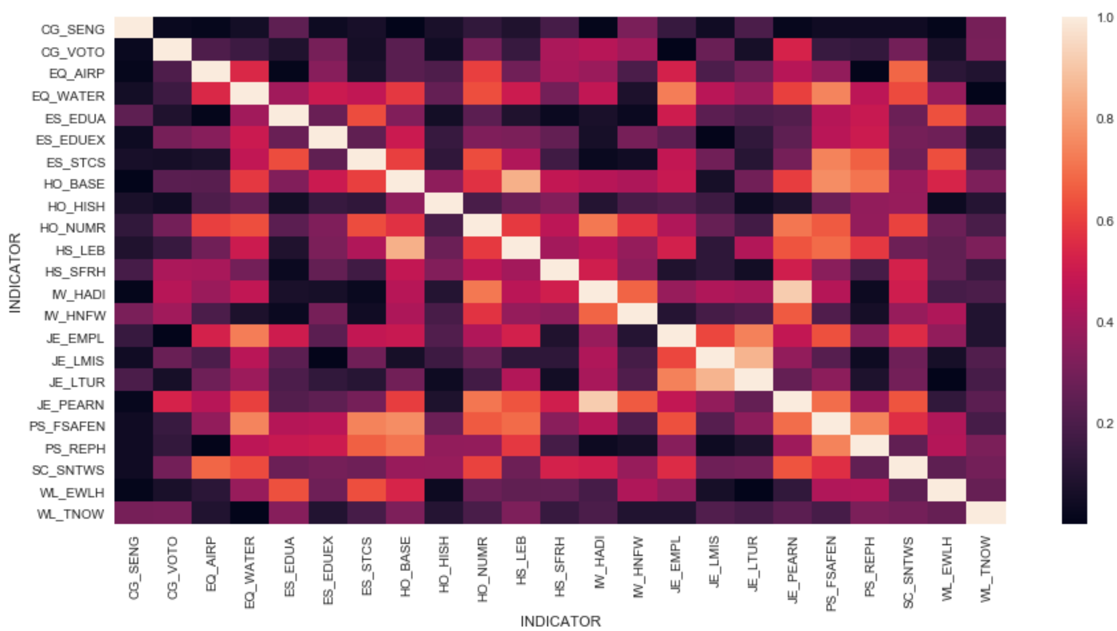
JE_LMIS	0.288360	0.063078	0.164635	0.264569	0.122740	0.125001
JE_LTUR	0.102944	0.292733	0.037768	0.173907	0.439576	0.050118
JE_PEARN	0.299373	0.596020	0.080009	0.711735	0.641399	0.513532
PS_FSAFEN	0.739562	0.765774	0.275512	0.655590	0.690902	0.345593
PS_REPH	0.664774	0.707629	0.364861	0.368320	0.585053	0.183617
SC_SNTWS	0.284479	0.383604	0.376605	0.606296	0.278339	0.525843
WL_EWLH	0.630973	0.534097	0.033516	0.274522	0.248687	0.252242
WL_TNOW	0.188257	0.318026	0.101389	0.193615	0.320410	0.147207

INDICATOR	IW_HADI	IW_HNFW	JE_EMPL	JE_LMIS	JE_LTUR	JE_PEARN \
INDICATOR						
CG_SENG	0.009595	0.313337	0.149286	0.046940	0.199630	0.019460
CG_VOTO	0.450701	0.399376	0.001314	0.272671	0.063474	0.529236
EQ_AIRP	0.384807	0.197628	0.525146	0.203927	0.284824	0.450445
EQ_WATER	0.474136	0.076700	0.726957	0.456311	0.390481	0.604712
ES_EDUA	0.069743	0.030136	0.504685	0.236128	0.202390	0.218113
ES_EDUEX	0.062532	0.304318	0.238336	0.007565	0.132076	0.245365
ES_STCS	0.028382	0.045354	0.478432	0.288360	0.102944	0.299373
HO_BASE	0.451797	0.428187	0.489324	0.063078	0.292733	0.596020
HO_HISH	0.095344	0.191833	0.215731	0.164635	0.037768	0.080009
HO_NUMR	0.717301	0.571859	0.435517	0.264569	0.173907	0.711735
HS_LEB	0.459078	0.374210	0.522467	0.122740	0.439576	0.641399
HS_SFRH	0.515910	0.355329	0.085938	0.125001	0.050118	0.513532
IW_HADI	1.000000	0.674728	0.378516	0.430542	0.415803	0.917182
IW_HNFW	0.674728	1.000000	0.099342	0.181178	0.210344	0.654348
JE_EMPL	0.378516	0.099342	1.000000	0.614939	0.735549	0.481038
JE_LMIS	0.430542	0.181178	0.614939	1.000000	0.857796	0.366901
JE_LTUR	0.415803	0.210344	0.735549	0.857796	1.000000	0.258938
JE_PEARN	0.917182	0.654348	0.481038	0.366901	0.258938	1.000000
PS_FSAFEN	0.448296	0.211345	0.634678	0.226821	0.354233	0.694537
PS_REPH	0.034060	0.056629	0.344470	0.037982	0.082079	0.392847
SC_SNTWS	0.510536	0.376644	0.552591	0.286091	0.294198	0.644078
WL_EWLH	0.187986	0.432014	0.364282	0.061697	0.004828	0.132684
WL_TNOW	0.197591	0.089883	0.087340	0.214011	0.188335	0.237182

INDICATOR	PS_FSAFEN	PS_REPH	SC_SNTWS	WL_EWLH	WL_TNOW
INDICATOR					
CG_SENG	0.042428	0.041994	0.043415	0.009313	0.302802
CG_VOTO	0.152899	0.136966	0.297239	0.067688	0.307280
EQ_AIRP	0.364504	0.007333	0.679765	0.115445	0.088929
EQ_WATER	0.745049	0.464015	0.622848	0.376035	0.004498
ES_EDUA	0.449514	0.490610	0.276437	0.634258	0.337958
ES_EDUEX	0.456138	0.503941	0.299917	0.282485	0.094854
ES_STCS	0.739562	0.664774	0.284479	0.630973	0.188257
HO_BASE	0.765774	0.707629	0.383604	0.534097	0.318026
HO_HISH	0.275512	0.364861	0.376605	0.033516	0.101389
HO_NUMR	0.655590	0.368320	0.606296	0.274522	0.193615
HS_LEB	0.690902	0.585053	0.278339	0.248687	0.320410

HS_SFRH	0.345593	0.183617	0.525843	0.252242	0.147207
IW_HADI	0.448296	0.034060	0.510536	0.187986	0.197591
IW_HNFW	0.211345	0.056629	0.376644	0.432014	0.089883
JE_EMPL	0.634678	0.344470	0.552591	0.364282	0.087340
JE_LMIS	0.226821	0.037982	0.286091	0.061697	0.214011
JE_LTUR	0.354233	0.082079	0.294198	0.004828	0.188335
JE_PEARN	0.694537	0.392847	0.644078	0.132684	0.237182
PS_FSAFEN	1.000000	0.739129	0.561306	0.437147	0.186304
PS_REPH	0.739129	1.000000	0.251516	0.442530	0.317128
SC_SNTWS	0.561306	0.251516	1.000000	0.246698	0.295831
WL_EWLH	0.437147	0.442530	0.246698	1.000000	0.263090
WL_TNOW	0.186304	0.317128	0.295831	0.263090	1.000000

[31]: <matplotlib.axes._subplots.AxesSubplot at 0x28a6b17a790>



[32]:
```python
# Load libraries
import warnings
from sklearn.datasets import make_regression
from sklearn.feature_selection import RFECV
from sklearn import linear_model

# Suppress an annoying but harmless warning
warnings.filterwarnings(action="ignore", module="scipy",
                        message="^internal gelsd")

# Generate features matrix, target vector, and the true coefficients
features, target = make_regression(n_samples = 10000,
```

```python
                                   n_features = 100,
                                   n_informative = 2,
                                   random_state = 1)

# Create a linear regression
ols = linear_model.LinearRegression()

# Recursively eliminate features
rfecv = RFECV(estimator=ols, step=1, scoring="neg_mean_squared_error")
rfecv.fit(features, target)
rfecv.transform(features)

# Once we have conducted RFE, we can see the number of features we should keep:
# Number of best features
print("Number of features we should keep: {}".format(rfecv.n_features_))

# We can also see which of those features we should keep:
# Which categories are best
print(rfecv.support_)

# Rank features best (1) to worst
print(rfecv.ranking_)
```

```
Number of features we should keep: 9
[False  True False False False  True False False False False False False
 False False False False False False  True False False False False False
 False False False False False False False False False False  True False
 False False False  True False False False False False False False  True
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False  True  True False False  True False False False False False
 False False False False]
[43  1 52 40 25  1 30 12 72  5 75 24 55 45 39 73 47 36  1 38 74 87 21 17
 13 56 11  4  3 33  9 59 22 29  1 63 34 41 84  1 10 26 28 71 78 42 91  1
 88 92 85 54 80 81 31 86 48  7 20 62 83 50  6 37 60 65 57 76 46 49 44  2
 15 66 16 35 90 82 77 69 64 32 18 51 23 67  1  1  8 53  1 89 68 58 79 61
 27 70 14 19]
```

References

http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#standardization-and-min-max-scaling Albon, Chris. Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning . O'Reilly Media. Kindle Edition.