In [1]:
```python
import sys                                   # system module
import pandas as pd                          # data package
import matplotlib.pyplot as plt              # graphics module
import datetime as dt                        # date and time module
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as smf
import statsmodels.formula.api as ols
```

READING WORLD BANK DATA

In [2]:
```python
# Reading in Per Capita Energy Consumption Data
energy_url='http://api.worldbank.org/v2/en/indicator/EG.USE.PCAP.KG.OE?dov
energy0=pd.read_excel(energy_url,
                     sheet_name='Data',
                     skiprows=3,
                     index_col=0,
                     usecols=range(270)
                        )
energy0.head()


# Reading in Per Capita GDP  Data
gdp_url='http://api.worldbank.org/v2/en/indicator/NY.GDP.PCAP.CD?download
gdp0=pd.read_excel(gdp_url,
                     sheet_name='Data',
                     skiprows=3,
                     index_col=0,
                     usecols=range(270)
                        )
gdp0.head()

# Reading in % Urban Population In Each Country
urban_url='http://api.worldbank.org/v2/en/indicator/SP.URB.TOTL.IN.ZS?down
urban0=pd.read_excel(urban_url,
                     sheet_name='Data',
                     skiprows=3,
                     index_col=0,
                     usecols=range(270)
                        )
urban0.head()


# Reading in % Contribution Of Services to Total GDP
services_url='http://api.worldbank.org/v2/en/indicator/NV.SRV.TOTL.ZS?down
services0=pd.read_excel(services_url,
                     sheet_name='Data',
                     skiprows=3,
                     index_col=0
```

```
                              index_col=0,
                              usecols=range(270)
                                  )
services0.head()
```

Out[2]:

	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	1966	...
Country Name											
Aruba	ABW	Services, value added (% of GDP)	NV.SRV.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Afghanistan	AFG	Services, value added (% of GDP)	NV.SRV.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Angola	AGO	Services, value added (% of GDP)	NV.SRV.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Albania	ALB	Services, value added (% of GDP)	NV.SRV.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Andorra	AND	Services, value added (% of GDP)	NV.SRV.TOTL.ZS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

5 rows × 61 columns

CLEANING UP DATA

In [3]:
```
# Extracting only Energy Consumption, GDP Per-Capita, Urban Population %,
energy=energy0[['2014']].dropna()
gdp=gdp0[['2014']].dropna()
urban=urban0[['2014']].dropna()
services=services0[['2014']].dropna()

energy=energy.reset_index()
gdp=gdp.reset_index()
urban=urban.reset_index()
services=services.reset_index()

# Renaming the columns
gdp.columns=['Country','GDP']
energy.columns=['Country','Energy']
urban.columns=['Country','Urban']
services.columns=['Country','Services']
```

In [4]:
```
# Merging all data into the same dataframe
data=pd.merge(energy.assign(Country=energy.Country.astype(str)),gdp.assign
data=pd.merge(data.assign(Country=data.Country.astype(str)),urban.assign((
data=pd.merge(data.assign(Country=data.Country.astype(str)),services.assig

data.head()
```

Out[4]:

	Country	Energy	GDP	Urban	Services
0	Albania	808.455840	4578.666728	56.423000	45.782515
1	Arab World	1953.286680	7452.814677	57.557605	40.855709
2	United Arab Emirates	7769.234738	44443.061514	85.375000	38.805326
3	Argentina	2015.187040	12245.256449	91.377000	52.940543
4	Armenia	1018.071240	3994.712355	63.112000	47.413044

REGRESSION ANALYSIS

In [5]:
```
# Regressing Energy Consumption against Per-Capita GDP
p=data[['GDP']]
p=smf.add_constant(p)

t=data[['Energy']]

model=smf.OLS(t,p).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                      Energy   R-squared:
0.440
```

```
0.440
Model:                            OLS   Adj. R-squared:
0.436
Method:                 Least Squares   F-statistic:
121.0
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
3.97e-21
Time:                        20:36:53   Log-Likelihood:
-1417.4

No. Observations:                 156   AIC:
2839.
Df Residuals:                     154   BIC:
2845.
Df Model:                           1
Covariance Type:            nonrobust
======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const        968.0536    223.430      4.333      0.000     526.670
1409.437
GDP            0.0887      0.008     10.998      0.000       0.073
0.105
======================================================================
========
Omnibus:                      131.874   Durbin-Watson:
1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
1621.450
Skew:                           3.049   Prob(JB):
0.00
Kurtosis:                      17.569   Cond. No.
3.59e+04
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 3.59e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

In [6]:

```
# Regressing Energy Consumption against Per-Capita GDP and Urban Populati

p=data[['GDP','Urban']]
p=smf.add_constant(p)

t=data[['Energy']]

model=smf.OLS(t,p).fit()
print(model.summary())
```

```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                 Energy   R-squared:
0.464
Model:                            OLS   Adj. R-squared:
0.457
Method:                 Least Squares   F-statistic:
66.19
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
1.94e-21
Time:                        20:36:53   Log-Likelihood:
-1414.0
No. Observations:                 156   AIC:
2834.
Df Residuals:                     153   BIC:
2843.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const       -525.6395    611.700     -0.859      0.392   -1734.108
682.829
GDP            0.0717      0.010      6.990      0.000       0.051
0.092
Urban         28.5710     10.923      2.616      0.010       6.992
50.150
==============================================================================
========
Omnibus:                      138.888   Durbin-Watson:
1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
1875.917
Skew:                           3.248   Prob(JB):
0.00
Kurtosis:                      18.697   Cond. No.
1.00e+05
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large,  1e+05. This might indicate that th
ere are
strong multicollinearity or other numerical problems.
```

In [7]:

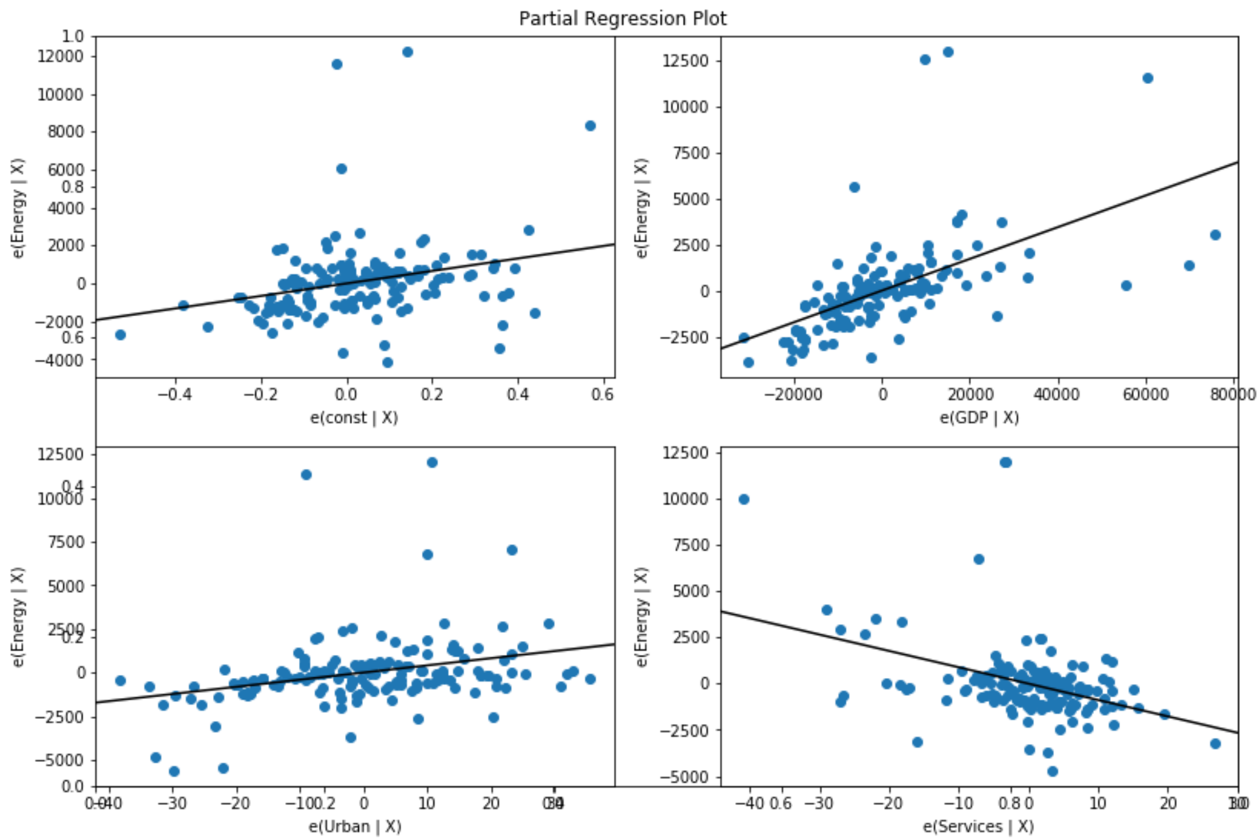
```
# Regressing Energy Consumption against Per-Capita GDP, Urban Pop.%, and
```

```
p=data[['GDP','Urban','Services']]
p=smf.add_constant(p)

t=data[['Energy']]

model=smf.OLS(t,p).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Energy   R-squared:                       0.548
Model:                            OLS   Adj. R-squared:                  0.539
Method:                 Least Squares   F-statistic:                     61.34
Date:                Fri, 21 Dec 2018   Prob (F-statistic):           4.80e-26
Time:                        20:36:53   Log-Likelihood:                -1400.8
No. Observations:                 156   AIC:                             2810.
Df Residuals:                     152   BIC:                             2822.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3284.0911    912.860      3.598      0.000    1480.558    5087.624
GDP            0.0858      0.010      8.741      0.000       0.066       0.105
Urban         41.0568     10.337      3.972      0.000      20.634      61.480
Services     -88.3785     16.656     -5.306      0.000    -121.287     -55.471
==============================================================================
Omnibus:                      135.688   Durbin-Watson:                   1.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2112.148
Skew:                           3.058   Prob(JB):                         0.00
Kurtosis:                      19.957   Cond. No.                     1.62e+05
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.62e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [8]:
```python
fig, ax=plt.subplots(figsize=(12,8))
fig = smf.graphics.plot_partregress_grid(model, fig=fig)
```



Adding more years of data

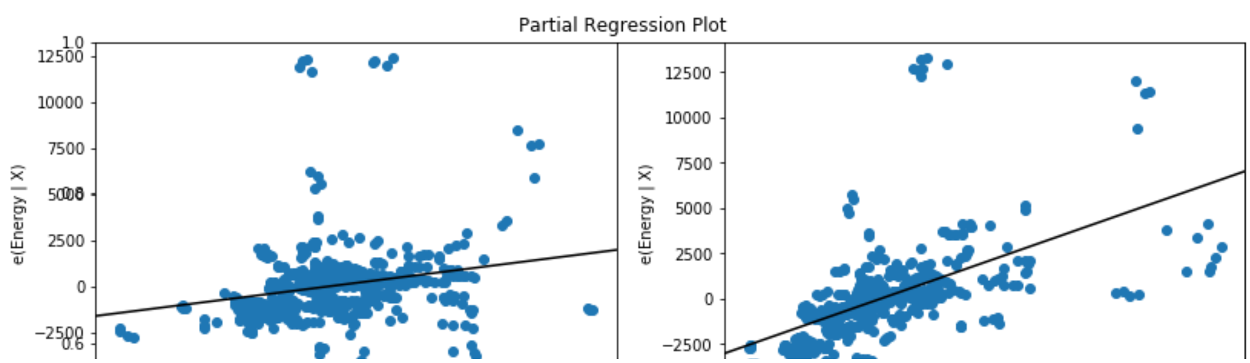
In [9]:
```python
######## 2013 DATA #########
energy_13=energy0[['2013']].dropna()
gdp_13=gdp0[['2013']].dropna()
urban_13=urban0[['2013']].dropna()
services_13=services0[['2013']].dropna()

energy_13=energy_13.reset_index()
gdp_13=gdp_13.reset_index()
urban_13=urban_13.reset_index()
services_13=services_13.reset_index()

gdp_13.columns=['Country','GDP']
energy_13.columns=['Country','Energy']
urban_13.columns=['Country','Urban']
services_13.columns=['Country','Services']
```

```
data_13=pd.merge(energy_13.assign(Country=energy_13.Country.astype(str)),g
data_13=pd.merge(data_13.assign(Country=data_13.Country.astype(str)),urban
data_13=pd.merge(data_13.assign(Country=data_13.Country.astype(str)),servi



######## 2012 DATA #########

energy_12=energy0[['2012']].dropna()
gdp_12=gdp0[['2012']].dropna()
urban_12=urban0[['2012']].dropna()
services_12=services0[['2012']].dropna()

energy_12=energy_12.reset_index()
gdp_12=gdp_12.reset_index()
urban_12=urban_12.reset_index()
services_12=services_12.reset_index()

gdp_12.columns=['Country','GDP']
energy_12.columns=['Country','Energy']
urban_12.columns=['Country','Urban']
services_12.columns=['Country','Services']

data_12=pd.merge(energy_12.assign(Country=energy_12.Country.astype(str)),g
data_12=pd.merge(data_12.assign(Country=data_12.Country.astype(str)),urban
data_12=pd.merge(data_12.assign(Country=data_12.Country.astype(str)),servi


######## 2011 DATA #########

energy_11=energy0[['2011']].dropna()
gdp_11=gdp0[['2011']].dropna()
urban_11=urban0[['2011']].dropna()
services_11=services0[['2011']].dropna()

energy_11=energy_11.reset_index()
gdp_11=gdp_11.reset_index()
urban_11=urban_11.reset_index()
services_11=services_11.reset_index()

gdp_11.columns=['Country','GDP']
energy_11.columns=['Country','Energy']
urban_11.columns=['Country','Urban']
services_11.columns=['Country','Services']

data_11=pd.merge(energy_11.assign(Country=energy_11.Country.astype(str)),g
data_11=pd.merge(data_11.assign(Country=data_11.Country.astype(str)),urban
data_11=pd.merge(data_11.assign(Country=data_11.Country.astype(str)),servi
```

In [10]:
```
data=data.append(data_13)

data=data.append(data_12)

data=data.append(data_11)
data=data.reset_index()
```

In [11]:
```
p=data[['GDP','Urban','Services']]
#p=data[['2014_GDP']]
p=smf.add_constant(p)
p.head()

t=data[['Energy']]
t.head()

model=smf.OLS(t,p).fit()
print(model.summary())


fig, ax=plt.subplots(figsize=(12,8))
fig = smf.graphics.plot_partregress_grid(model, fig=fig)
```

```
==========================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.44e+05. This might indicate that
there are
strong multicollinearity or other numerical problems.
```



In [12]:
```
combined_model_coeff = list(model.tvalues)
```

Splitting into High, middle and low income countries

In [13]:
```
data=data.reset_index()
data.sort_values('GDP',ascending=False).head()
```

Out[13]:

	level_0	index	Country	Energy	GDP	Urban	Services
90	90	90	Luxembourg	6861.106667	119225.380023	89.884	77.599543
569	569	90	Luxembourg	8056.404301	115761.507705	88.906	78.306119
245	245	89	Luxembourg	7312.154005	113625.132900	89.574	78.125206
407	407	90	Luxembourg	7722.190204	106749.013623	89.249	78.256920
264	264	108	Norway	6415.990714	103059.248228	80.286	52.624111

In [14]:
```
data_high=data[data['GDP']>15000]
```

In [15]:
```
data_low=data[data['GDP']<5000]
```

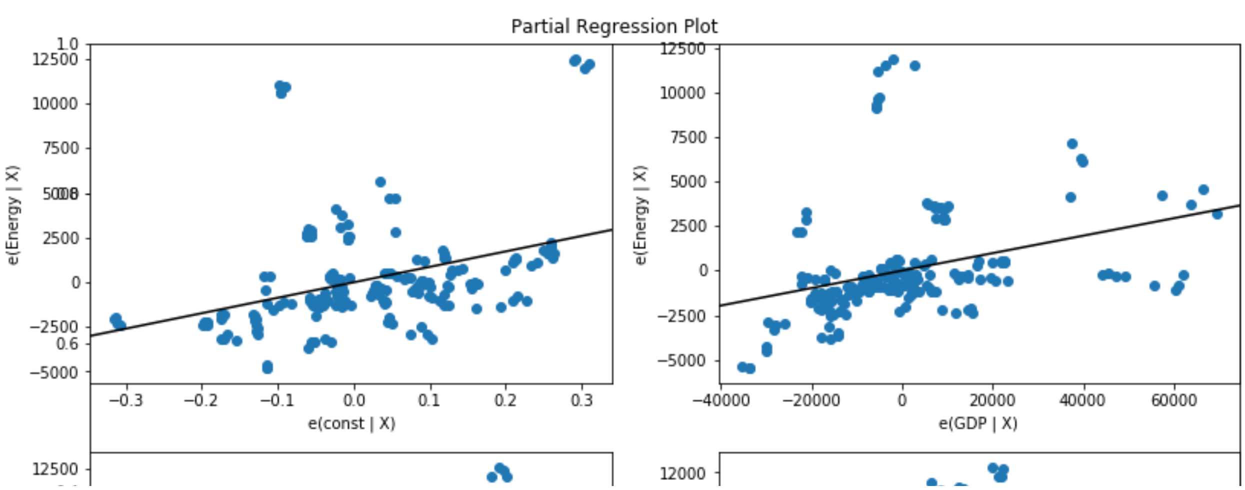
In [16]:
```
data_middle=data[(data['GDP']<=15000) & (data['GDP']>=5000)]
```

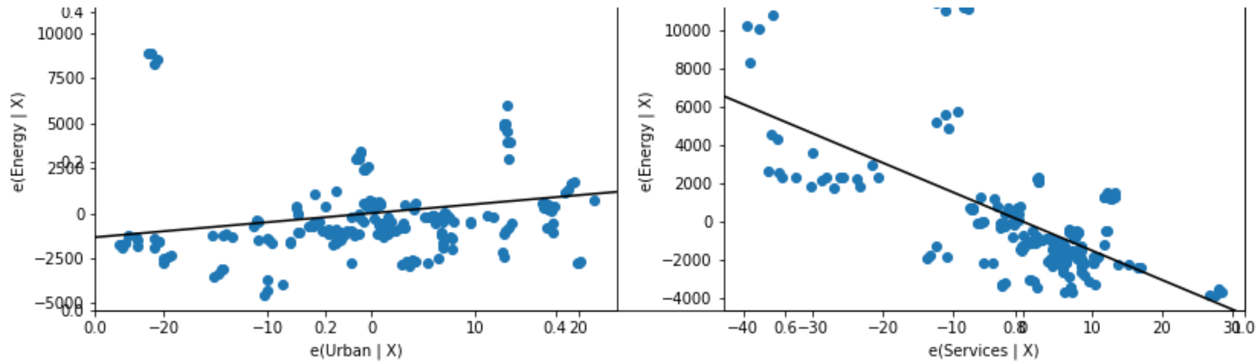
High Income

In [17]:
```
p=data_high[['GDP','Urban','Services']]
#p=data[['2014_GDP']]
p=smf.add_constant(p)
p.head()

t=data_high[['Energy']]
t.head()

model=smf.OLS(t,p).fit()
print(model.summary())


fig, ax=plt.subplots(figsize=(12,8))
fig = smf.graphics.plot_partregress_grid(model, fig=fig)
```

```
                            OLS Regression Results
========================================================================
========
Dep. Variable:                 Energy   R-squared:
0.403
Model:                            OLS   Adj. R-squared:
0.393
Method:                 Least Squares   F-statistic:
44.03
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
8.52e-22
Time:                        20:36:54   Log-Likelihood:
```

```
-1866.3
No. Observations:                 200   AIC:
3741.
Df Residuals:                     196   BIC:
3754.
Df Model:                           3
Covariance Type:            nonrobust
======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const       8647.4280   1557.583      5.552      0.000    5575.653
1.17e+04
GDP            0.0489      0.010      4.974      0.000       0.030
0.068
Urban         50.1254     17.590      2.850      0.005      15.436
84.815
Services    -153.6988     15.317    -10.035      0.000    -183.905
-123.492
======================================================================
========
Omnibus:                      121.820   Durbin-Watson:
2.422
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
644.029
Skew:                           2.449   Prob(JB):                    1
.42e-140
Kurtosis:                      10.301   Cond. No.
3.75e+05
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 3.75e+05. This might indicate that
there are
strong multicollinearity or other numerical problems.
```





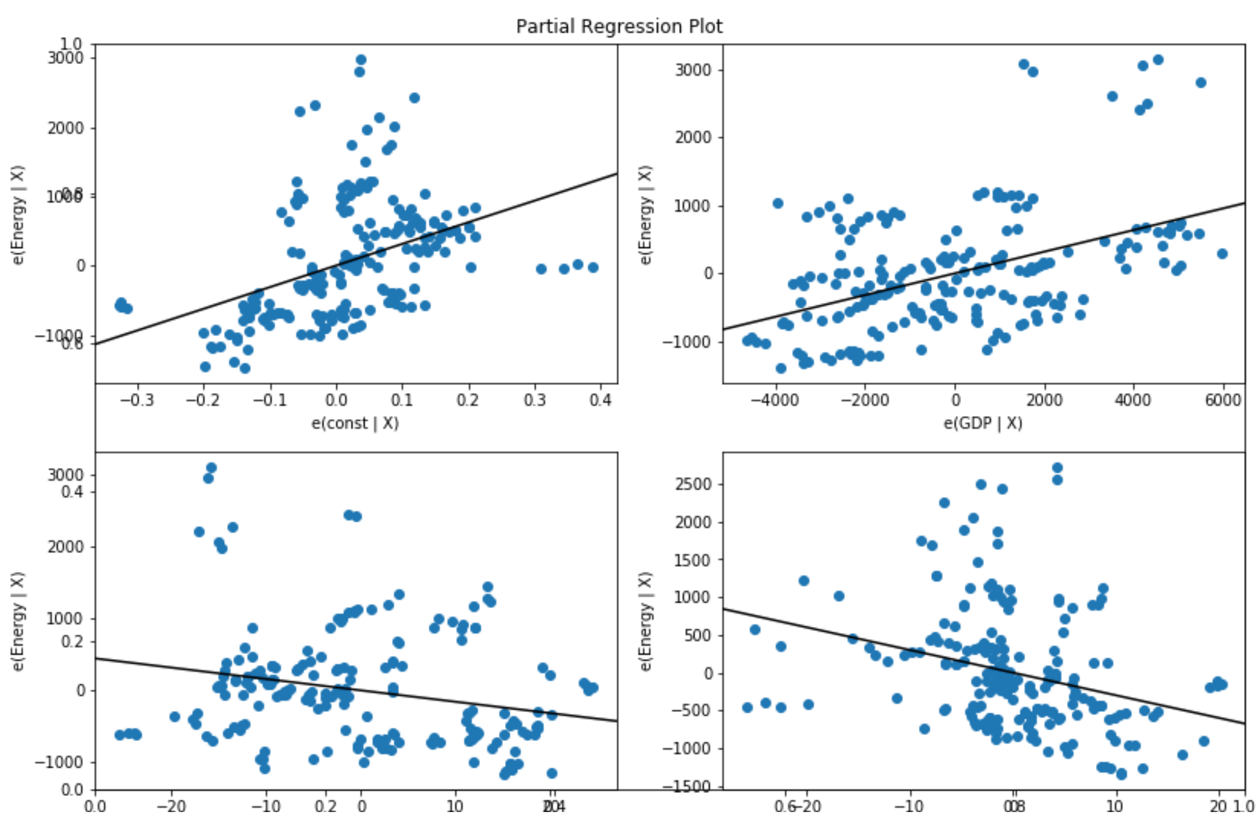
In [18]:
```python
high_income_coeff = list(model.tvalues)
```

Middle Income

In [19]:
```python
p=data_middle[['GDP','Urban','Services']]
#p=data[['2014_GDP']]
p=smf.add_constant(p)
p.head()

t=data_middle[['Energy']]
t.head()

model=smf.OLS(t,p).fit()
print(model.summary())


fig, ax=plt.subplots(figsize=(12,8))
fig = smf.graphics.plot_partregress_grid(model, fig=fig)
```

```
                            OLS Regression Results
======================================================================
========
Dep. Variable:                 Energy   R-squared:
0.244
Model:                            OLS   Adj. R-squared:
0.233
Method:                 Least Squares   F-statistic:
21.52
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
4.02e-12
Time:                        20:36:55   Log-Likelihood:
-1642.0
No. Observations:                 204   AIC:
3292.
Df Residuals:                     200   BIC:
3305.
Df Model:                           3
Covariance Type:            nonrobust
======================================================================
========
```

```
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------
--------
const       3119.7241    452.123      6.900      0.000    2228.184
4011.264
GDP            0.1592      0.021      7.429      0.000       0.117
0.202
Urban        -15.7714      4.594     -3.433      0.001     -24.829
-6.713
Services     -29.9716      6.860     -4.369      0.000     -43.499
-16.444
========================================================================
========
Omnibus:                       46.025   Durbin-Watson:
2.253
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
74.341
Skew:                           1.218   Prob(JB):
7.19e-17
Kurtosis:                       4.676   Cond. No.
8.06e+04
========================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 8.06e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```



e(Urban | X) e(Services | X)

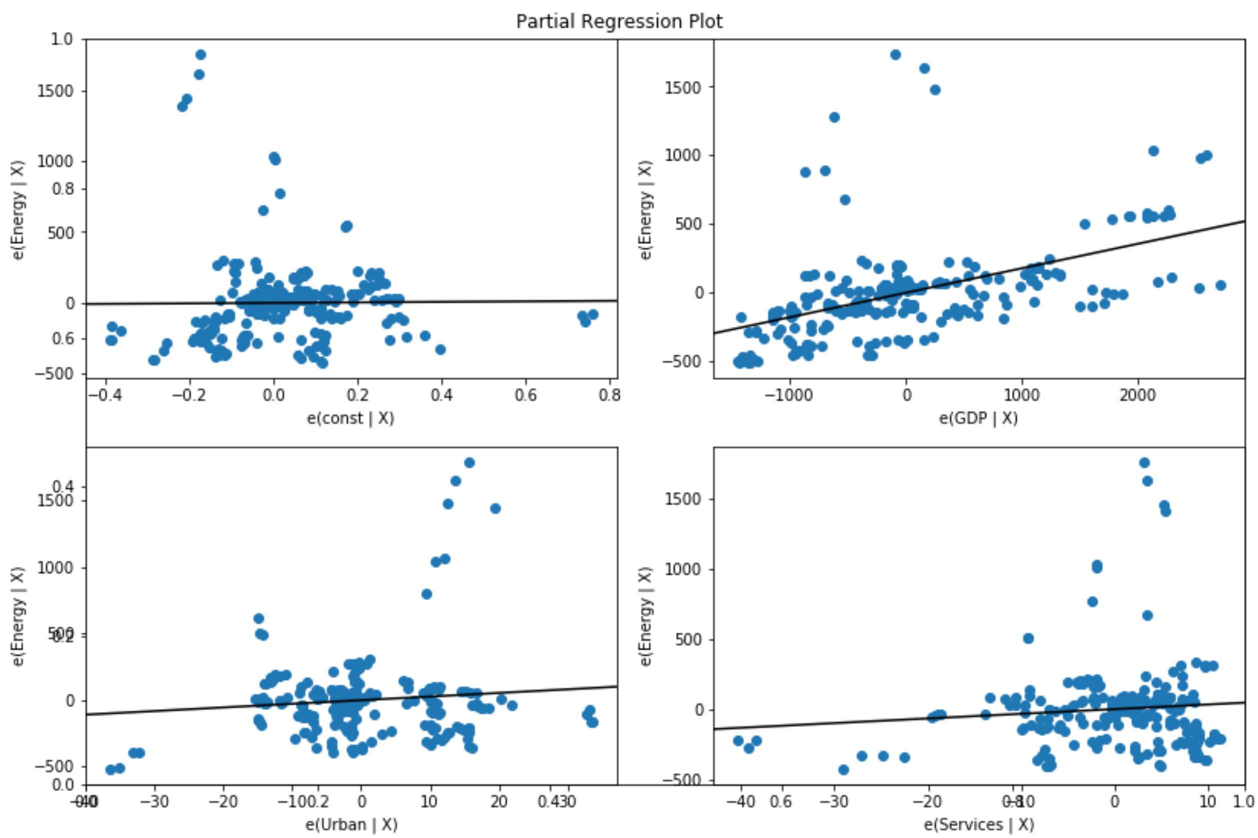
In [20]:
```
middle_income_coeff = list(model.tvalues)
```

Low Income

In [21]:
```
p=data_low[['GDP','Urban','Services']]
#p=data[['2014_GDP']]
p=smf.add_constant(p)
p.head()

t=data_low[['Energy']]
t.head()

model=smf.OLS(t,p).fit()
print(model.summary())


fig, ax=plt.subplots(figsize=(12,8))
fig = smf.graphics.plot_partregress_grid(model, fig=fig)
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                 Energy   R-squared:
0.453
Model:                            OLS   Adj. R-squared:
0.446
Method:                 Least Squares   F-statistic:
64.23
Date:                Fri, 21 Dec 2018   Prob (F-statistic):
2.66e-30
Time:                        20:36:55   Log-Likelihood:
-1681.2
No. Observations:                 237   AIC:
3370.
Df Residuals:                     233   BIC:
3384.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const         17.6760    114.999      0.154      0.878    -208.895
244.247
GDP            0.1783      0.020      8.807      0.000       0.138
0.218
Urban          2.7316      1.733      1.576      0.116      -0.683
```

```
6.146
Services        3.3172      2.276      1.458      0.146      -1.167
7.801
========================================================================
========
Omnibus:                     179.151   Durbin-Watson:
1.508
Prob(Omnibus):                 0.000   Jarque-Bera (JB):
2082.336
Skew:                          2.947   Prob(JB):
0.00
Kurtosis:                     16.272   Cond. No.
1.47e+04
========================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.47e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```



In [31]:
```
low_income_coeff = list(model.tvalues)
```

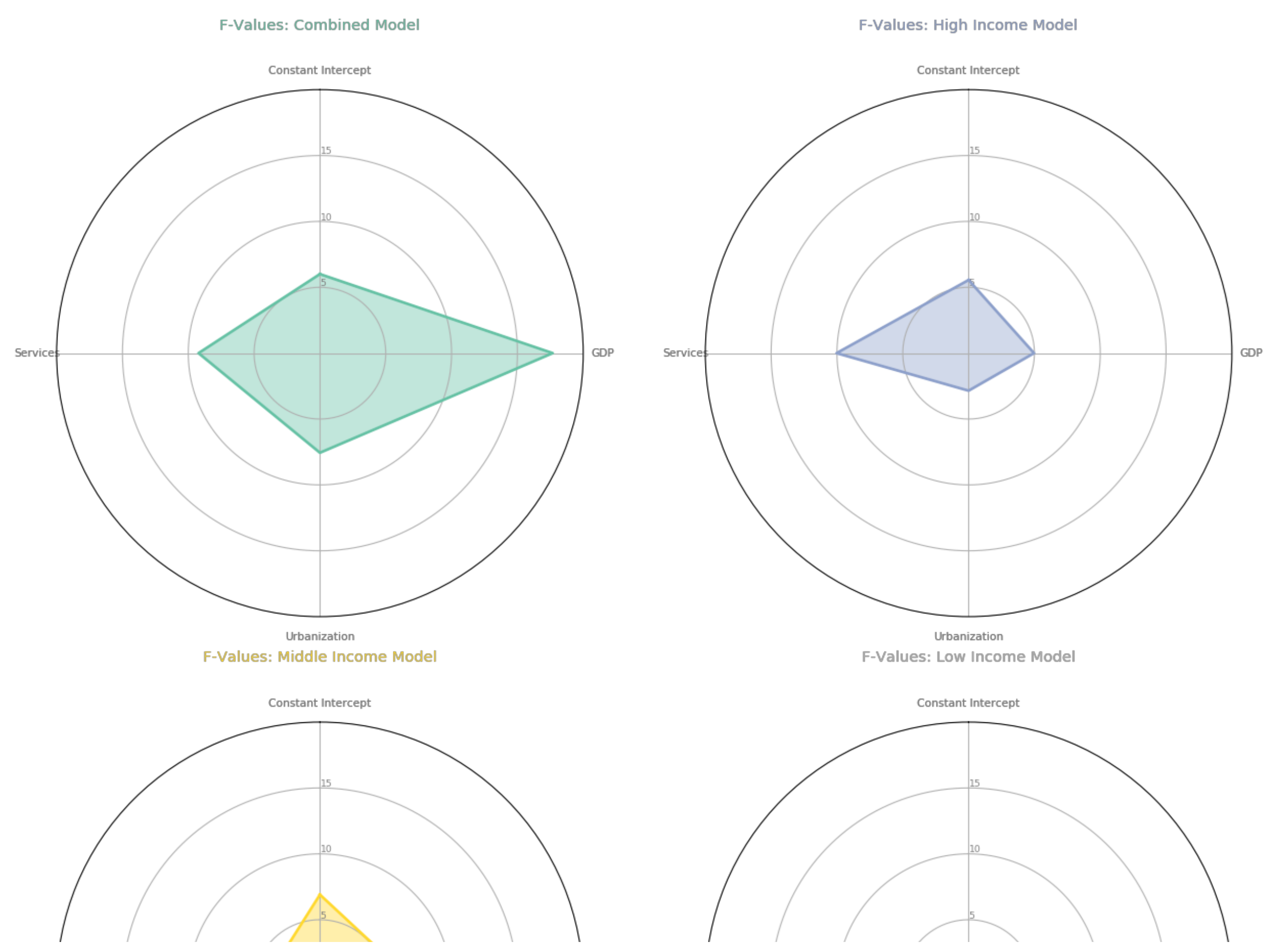
```
In [54]: print([ combined_model_coeff[0], high_income_coeff[0], middle_income_coe
         print([ combined_model_coeff[1], high_income_coeff[1], middle_income_coe
         print([combined_model_coeff[2], high_income_coeff[2], middle_income_coef
         print([combined_model_coeff[3], high_income_coeff[3], middle_income_coef
```

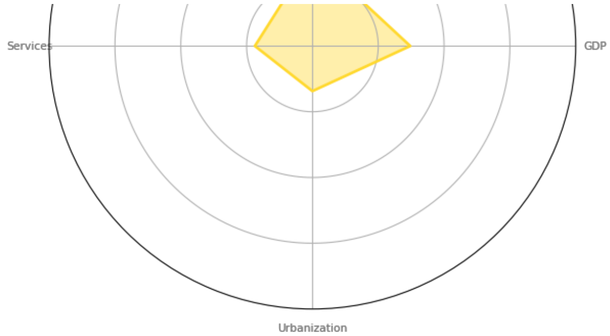
```
[6.0008878893835975, 5.551823237757189, 6.900164122112461, 0.153705170
5996944]
[17.67569163536986, 4.973912758676553, 7.428590177648136, 8.8065161775
2576]
[7.558718271721758, 2.849683339406594, -3.433363355519905, 1.576320422
0961533]
[-9.220842297974716, -10.034746283281255, -4.369026734541702, 1.457615
748946877]
```

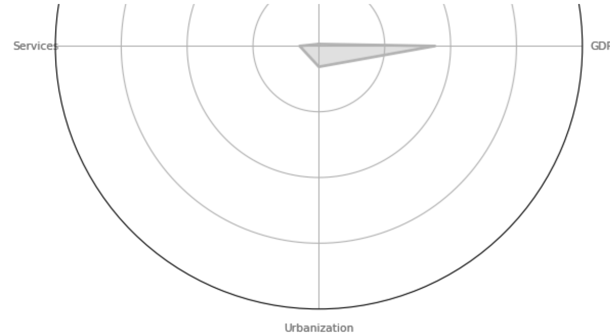
PLOTTING ABSOLUTE F-VALUES FOR EACH COEFFICIENT OF EACH MODEL

```
In [67]: # Libraries
         import matplotlib.pyplot as plt
         import pandas as pd
         from math import pi

         # Set data
         df = pd.DataFrame({
         'F-Values:': ['Combined Model','High Income Model','Middle Income Model',
         'Constant Intercept':[6.00, 5.55, 6.901, 0.15],
         'GDP': [17.68, 4.97, 7.43, 8.81],
         'Urbanization': [7.56, 2.85, 3.43, 1.58],
         'Services': [9.22, 10.03, 4.37, 1.46]
         })

         # ------- PART 1: Define a function that do a plot for one line of the da

         def make_spider( row, title, color):

             # number of variable
             categories=list(df)[1:]
             N = len(categories)

             # What will be the angle of each axis in the plot? (we divide the plo
             angles = [n / float(N) * 2 * pi for n in range(N)]
             angles += angles[:1]

             # Initialise the spider plot
             ax = plt.subplot(2,2,row+1, polar=True, )

             # If you want the first axis to be on top:
             ax.set_theta_offset(pi / 2)
             ax.set_theta_direction(-1)
```

```
    # Draw one axe per variable + add labels labels yet
    plt.xticks(angles[:-1], categories, color='grey', size=8)

    # Draw ylabels
    ax.set_rlabel_position(0)
    plt.yticks([5,10,15], ["5","10","15"], color="grey", size=7)
    plt.ylim(0,20)

    # Ind1
    values=df.loc[row].drop('F-Values:').values.flatten().tolist()
    values += values[:1]
    ax.plot(angles, values, color=color, linewidth=2, linestyle='solid')
    ax.fill(angles, values, color=color, alpha=0.4)

    # Add a title
    plt.title(title, size=11, color=color, y=1.1)

# ------- PART 2: Apply to all individuals
# initialize the figure
my_dpi=96
plt.figure(figsize=(1500/my_dpi, 1500/my_dpi), dpi=my_dpi)
# Create a color palette:
my_palette = plt.cm.get_cmap("Set2", len(df.index))

# Loop to plot
for row in range(0, len(df.index)):
    make_spider( row=row, title='F-Values: '+df['F-Values:'][row], color=
```







In [ ]: