Project 1: Life Expectancy Dataset (WHO and UN)

By:

Rhea Sethi: 106087299

Snehil Shandilya: 306077534

Steven Spear: 606085052

Dhriti Sahoo: 505928964

Motivation for the study:

Motivation for the study is what factors can influence life expectancy. The dataset is taken from the World Health Organization (WHO) and United Nations website for the year 2012 and takes into account 183 countries. The variables included in the dataset include social and economic indicators as well as the effects of diseases and their respective rates of vaccinations. The results of the study would help us understand the major factors influencing life expectancy for different countries and can help determine specific policy actions.

Variable Description

1. life_expectancy: Life Expectancy in years
2. adult_mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
3. infant_deaths: Number of Infant Deaths per 1000 population
4. alcohol: Alcohol, recorded per capita (15+) consumption (in liters of pure alcohol)
5. percentage_expenditure: Expenditure on health as a percentage of Gross Domestic Product per capita(%)
6. hepatitis_b: Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
7. bmi: Average Body Mass Index of entire population
8. under_five_deaths: Number of under-five deaths per 1000 population
9. Total_expenditure: General government expenditure on health as a percentage of total government expenditure (%)
10. Diphtheria: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
11. HIV/AIDS: Deaths per 1000 live births HIV/AIDS (0-4 years)
12. GDP: Gross Domestic Product per capita (in USD)
13. Schooling: Number of years of Schooling(years)

In [123]:
```
#import all the neccessary packages
import wooldridge as woo
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.formula.api as smf
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
import statsmodels.api as sms
```

In [2]:
```
#import csv file
# Assign data to variable
df = pd.read_excel("/Users/snehilshandilya/Desktop/430_Data_Cleaned.xlsx", index_col = "Country")
df
```

Out[2]:

Country	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	polio	total_
Afghanistan	59.5	272	69	0.01	78.184215	67.0	17.6	93	67	
Albania	76.9	86	0	5.14	412.443356	99.0	55.8	1	99	
Algeria	75.1	113	21	0.66	555.926083	95.0	56.1	24	95	
Angola	56.0	358	72	8.24	256.122524	75.0	21.5	110	75	
Antigua and Barbuda	75.9	134	0	8.18	2156.229842	98.0	45.7	0	97	
...	...	...	...	...	...	...	...	...	...	
Venezuela (Bolivarian Republic of)	73.7	161	9	6.70	0.000000	81.0	6.4	10	73	
Viet Nam	75.6	13	29	4.12	0.000000	97.0	15.3	36	97	
Yemen	64.7	236	36	0.04	0.000000	67.0	38.7	46	68	
Zambia	59.2	349	29	2.59	196.915250	78.0	21.7	43	7	
Zimbabwe	56.6	429	26	6.09	92.602336	97.0	3.3	39	95	

183 rows × 14 columns

Inspecting the data

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 183 entries, Afghanistan to Zimbabwe
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   life_expectancy         183 non-null    float64
 1   adult_mortality         183 non-null    int64  
 2   infant_deaths           183 non-null    int64  
 3   alcohol                 182 non-null    float64
 4   percentage_expenditure  183 non-null    float64
 5   hepatitis_b             170 non-null    float64
 6   bmi                     181 non-null    float64
 7   under_five_deaths       183 non-null    int64  
 8   polio                   183 non-null    int64  
 9   total_expenditure       181 non-null    float64
 10  diphtheria              183 non-null    int64  
 11  hiv_aids                183 non-null    float64
 12  gdp                     154 non-null    float64
 13  schooling               173 non-null    float64
dtypes: float64(9), int64(5)
memory usage: 21.4+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	p
count	183.000000	183.000000	183.000000	182.000000	183.000000	170.000000	181.000000	183.000000	183.000
mean	70.916940	148.792350	26.338798	4.138187	1011.471726	82.635294	40.099448	35.562842	83.939
std	8.562151	108.617944	96.811105	4.272119	2273.787938	24.861199	20.775655	127.826312	24.120
min	49.700000	2.000000	0.000000	0.010000	0.000000	2.000000	2.100000	0.000000	3.000
25%	64.500000	66.500000	0.000000	0.010000	32.597328	81.000000	21.900000	0.000000	83.000
50%	73.200000	138.000000	3.000000	2.890000	196.915250	94.000000	45.900000	3.000000	94.000
75%	76.500000	215.500000	19.500000	7.785000	847.322649	97.000000	58.500000	23.500000	98.000
max	88.000000	513.000000	1100.000000	16.350000	18379.329740	99.000000	76.200000	1400.000000	99.000

In [5]:
```
df.isnull().any()
```

Out[5]:
```
life_expectancy           False
adult_mortality           False
infant_deaths             False
alcohol                    True
percentage_expenditure    False
hepatitis_b                True
bmi                        True
under_five_deaths         False
polio                     False
total_expenditure          True
diphtheria                False
hiv_aids                  False
gdp                        True
schooling                  True
dtype: bool
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
life_expectancy            0
adult_mortality            0
infant_deaths              0
alcohol                    1
percentage_expenditure     0
hepatitis_b               13
bmi                        2
under_five_deaths          0
polio                      0
total_expenditure          2
diphtheria                 0
hiv_aids                   0
gdp                       29
schooling                 10
dtype: int64
```

Since we have NAs in our dataset (some of which come up as the value "0"), we replace them using the median before conducting our analysis

In [7]:
```
df = df.fillna(df.median())
df
```

Out[7]:

Country	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	polio	total_
Afghanistan	59.5	272	69	0.01	78.184215	67.0	17.6	93	67	
Albania	76.9	86	0	5.14	412.443356	99.0	55.8	1	99	
Algeria	75.1	113	21	0.66	555.926083	95.0	56.1	24	95	
Angola	56.0	358	72	8.24	256.122524	75.0	21.5	110	75	
Antigua and Barbuda	75.9	134	0	8.18	2156.229842	98.0	45.7	0	97	
...	...	...	...	...	...	...	...	...	...	
Venezuela (Bolivarian Republic of)	73.7	161	9	6.70	0.000000	81.0	6.4	10	73	
Viet Nam	75.6	13	29	4.12	0.000000	97.0	15.3	36	97	
Yemen	64.7	236	36	0.04	0.000000	67.0	38.7	46	68	
Zambia	59.2	349	29	2.59	196.915250	78.0	21.7	43	7	
Zimbabwe	56.6	429	26	6.09	92.602336	97.0	3.3	39	95	

183 rows × 14 columns

In [8]:
```
df.isnull().any()
```

Out[8]:
```
life_expectancy           False
adult_mortality           False
infant_deaths             False
alcohol                   False
percentage_expenditure    False
hepatitis_b               False
bmi                       False
under_five_deaths         False
polio                     False
total_expenditure         False
diphtheria                False
hiv_aids                  False
gdp                       False
schooling                 False
dtype: bool
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
life_expectancy           0
adult_mortality           0
infant_deaths             0
alcohol                   0
percentage_expenditure    0
hepatitis_b               0
bmi                       0
under_five_deaths         0
polio                     0
total_expenditure         0
diphtheria                0
hiv_aids                  0
gdp                       0
schooling                 0
dtype: int64
```

In [10]:
```
df = df.replace(0,df.median())
df
```

Out[10]:

Country	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	polio	total_
Afghanistan	59.5	272	69	0.01	78.184215	67.0	17.6	93	67	
Albania	76.9	86	3	5.14	412.443356	99.0	55.8	1	99	
Algeria	75.1	113	21	0.66	555.926083	95.0	56.1	24	95	
Angola	56.0	358	72	8.24	256.122524	75.0	21.5	110	75	
Antigua and Barbuda	75.9	134	3	8.18	2156.229842	98.0	45.7	3	97	
...	...	...	...	...	...	...	...	...	...	
Venezuela (Bolivarian Republic of)	73.7	161	9	6.70	196.915250	81.0	6.4	10	73	
Viet Nam	75.6	13	29	4.12	196.915250	97.0	15.3	36	97	
Yemen	64.7	236	36	0.04	196.915250	67.0	38.7	46	68	
Zambia	59.2	349	29	2.59	196.915250	78.0	21.7	43	7	
Zimbabwe	56.6	429	26	6.09	92.602336	97.0	3.3	39	95	

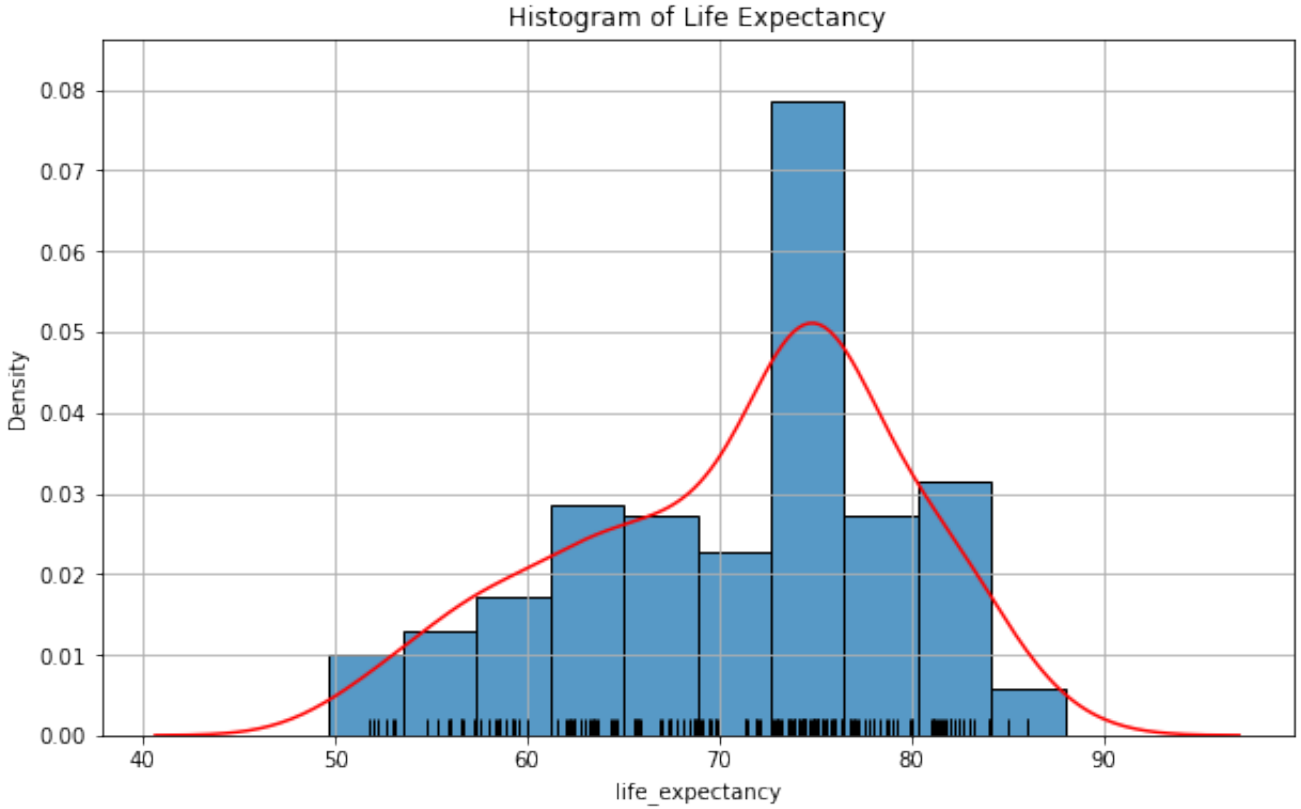
183 rows × 14 columns

Q1. Part a and b

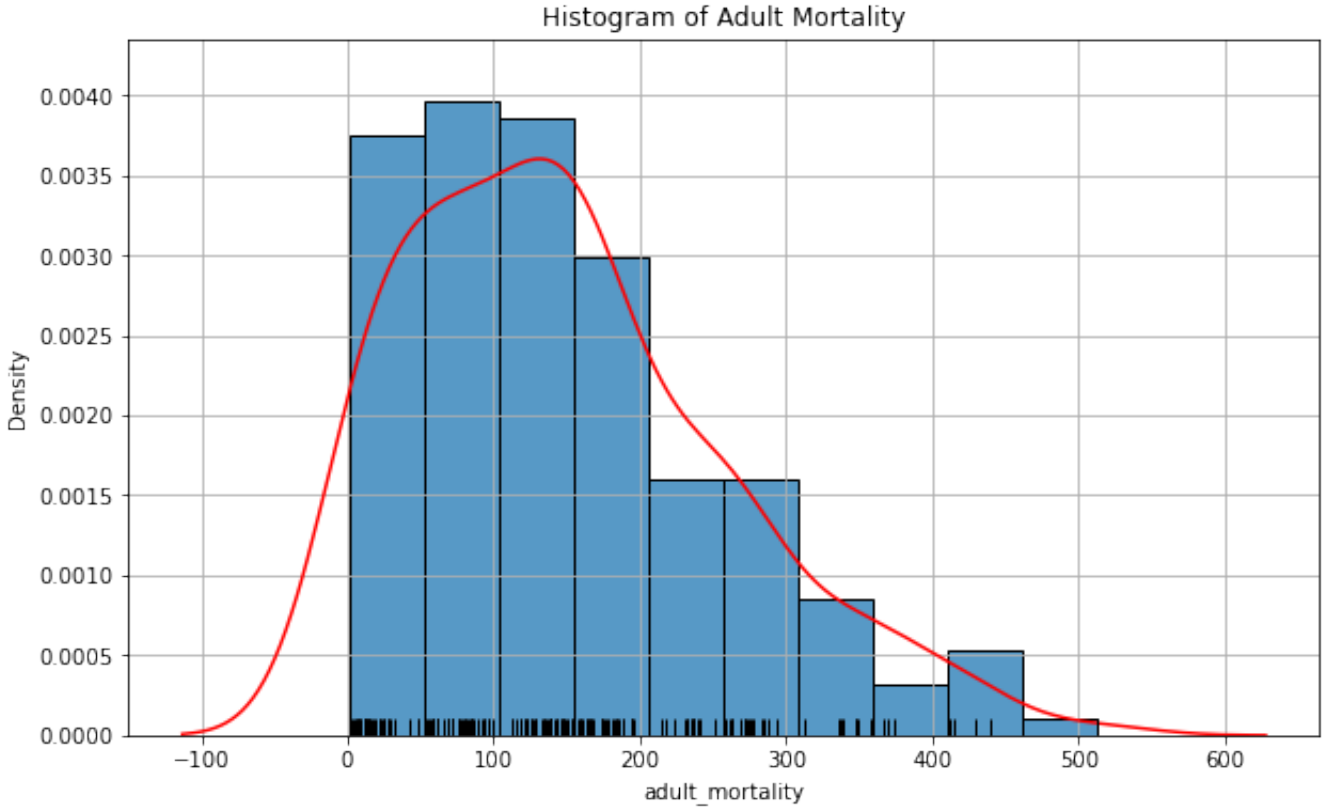
Descriptive Analysis

Analysis of the histograms, quantile plots, scatter plots etc. along with density curves suggests that the majority of the variables look skewed and therefore not normally distributed.

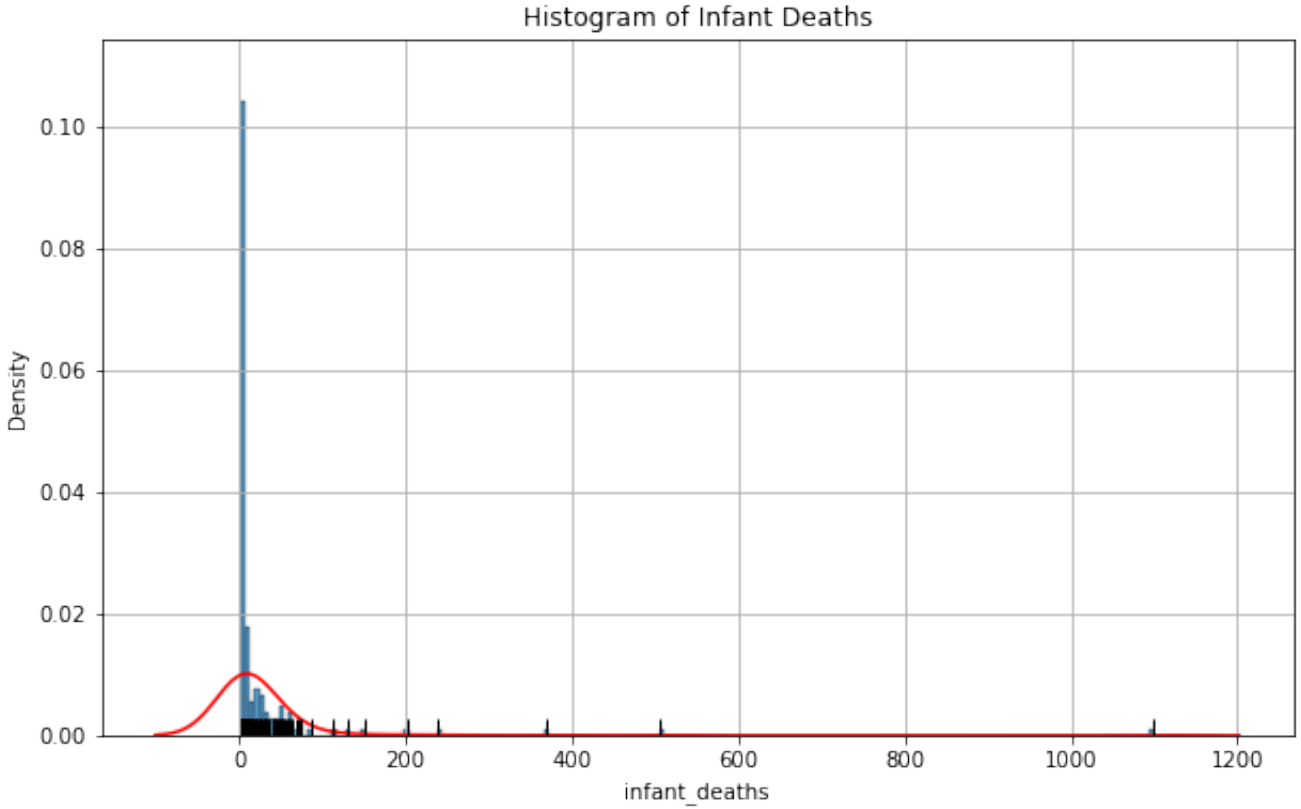
In [11]:
```
#histogram/denisty plot - life_expectancy
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy")
sns.histplot(df.life_expectancy, stat = "density")
sns.kdeplot(df.life_expectancy, color = "red")
sns.rugplot(df.life_expectancy, color = "black")

plt.grid()
```



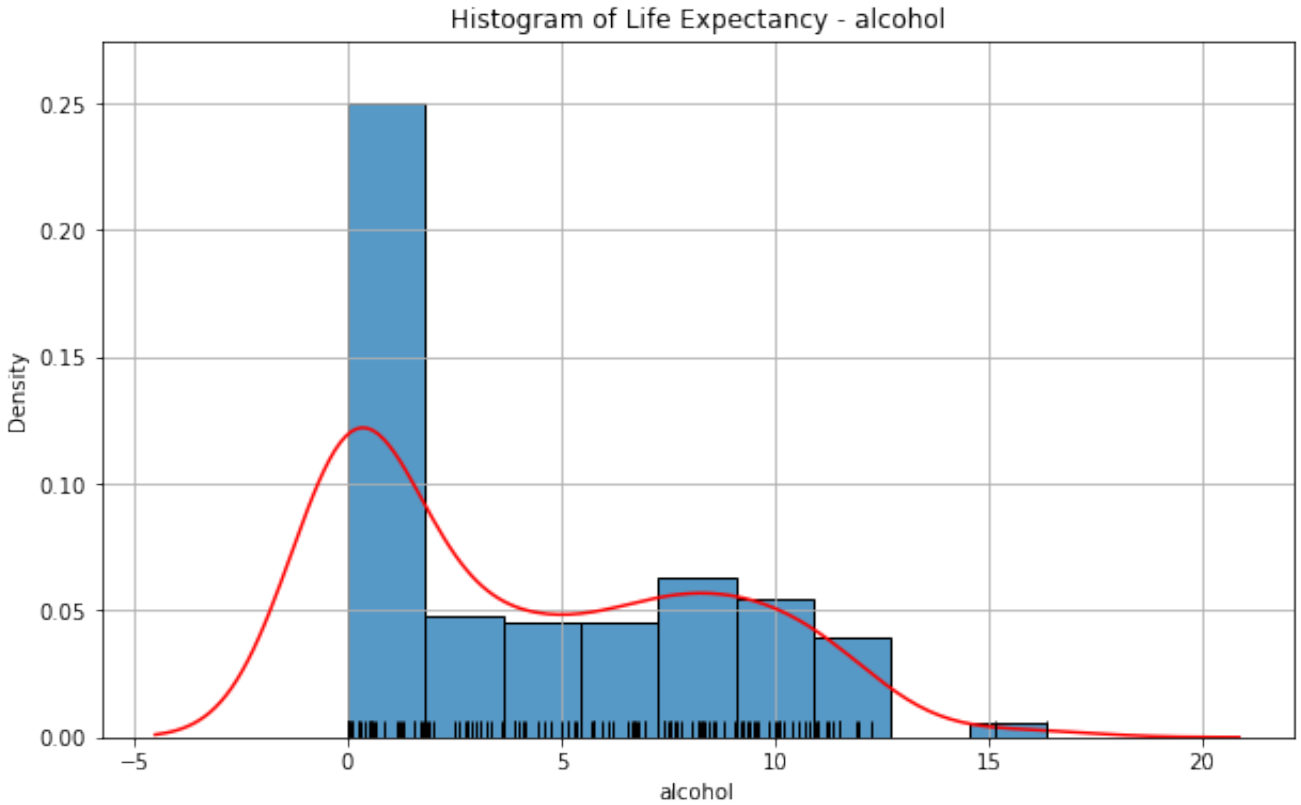
In [12]:
```
#histogram/denisty plot - adult_mortality
plt.figure(figsize=(10,6))
plt.title("Histogram of Adult Mortality")
sns.histplot(df.adult_mortality, stat = "density")
sns.kdeplot(df.adult_mortality, color = "red")
sns.rugplot(df.adult_mortality, color = "black")

plt.grid()
```



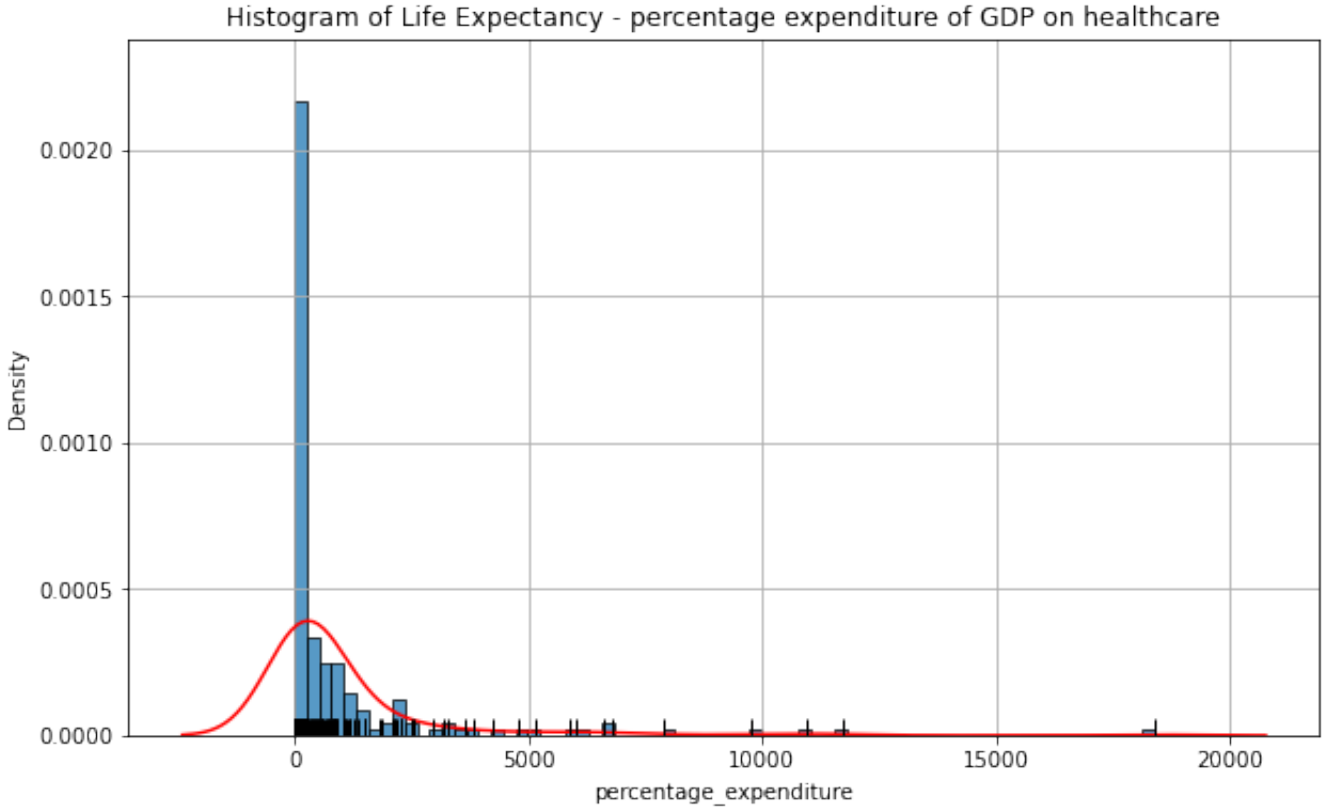
In [13]:
```
#histogram/denisty plot - infant_deaths
plt.figure(figsize=(10,6))
plt.title("Histogram of Infant Deaths")
sns.histplot(df.infant_deaths, stat = "density")
sns.kdeplot(df.infant_deaths, color = "red")
sns.rugplot(df.infant_deaths, color = "black")

plt.grid()
```



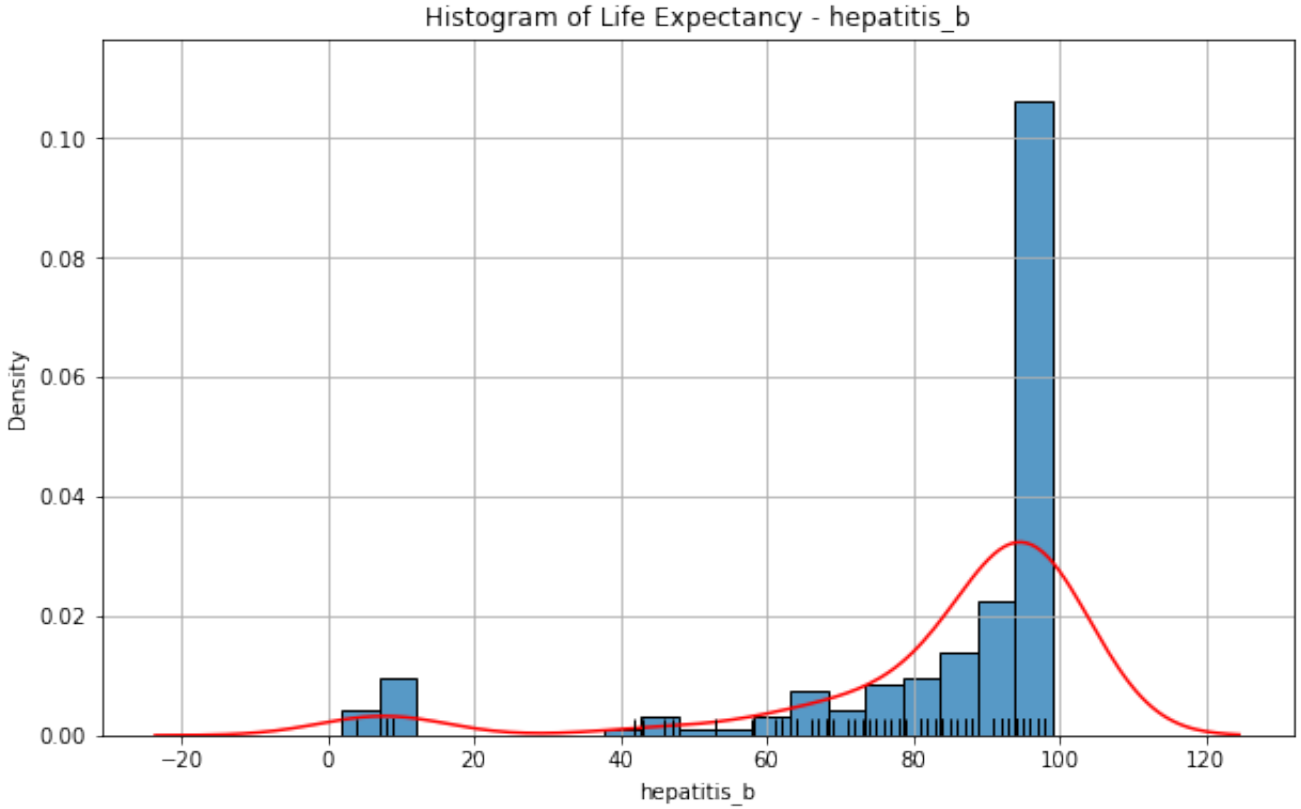
In [14]:
```
#histogram/denisty plot - alcohol
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - alcohol")
sns.histplot(df.alcohol, stat = "density")
sns.kdeplot(df.alcohol, color = "red")
sns.rugplot(df.alcohol, color = "black")

plt.grid()
```



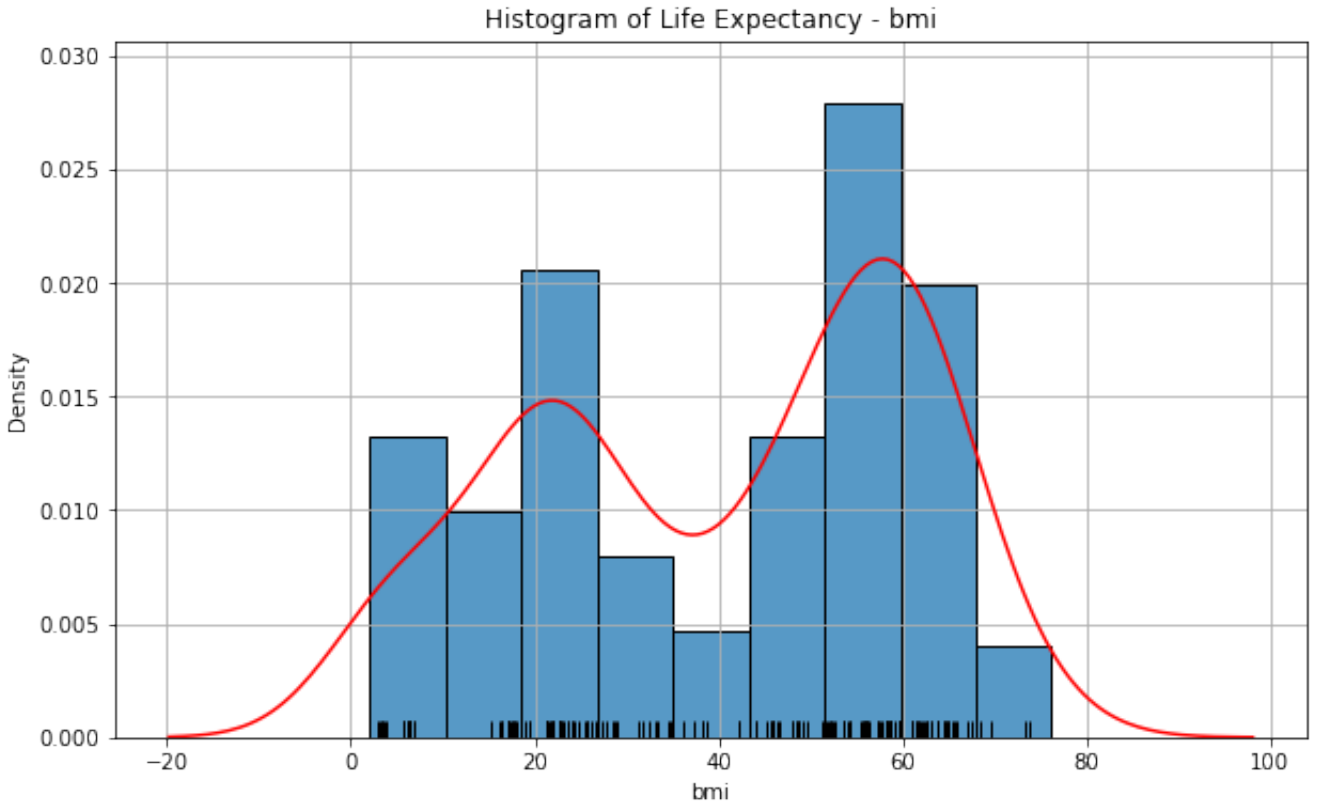
In [15]:
```
#histogram/denisty plot - percentage_expenditure
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - percentage expenditure of GDP on healthcare")
sns.histplot(df.percentage_expenditure, stat = "density")
sns.kdeplot(df.percentage_expenditure, color = "red")
sns.rugplot(df.percentage_expenditure, color = "black")

plt.grid()
```



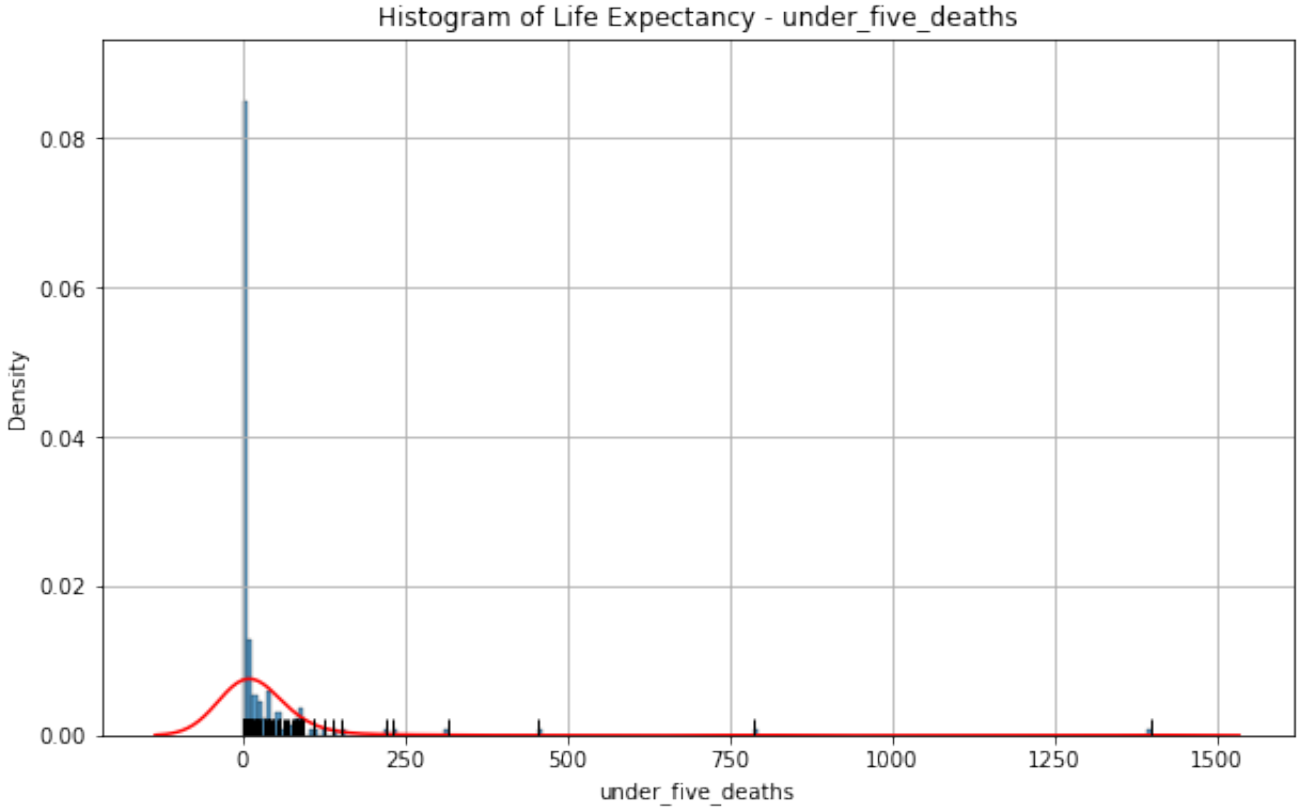
In [16]:
```
#histogram/denisty plot - hepatitis_b
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - hepatitis_b")
sns.histplot(df.hepatitis_b, stat = "density")
sns.kdeplot(df.hepatitis_b, color = "red")
sns.rugplot(df.hepatitis_b, color = "black")

plt.grid()
```



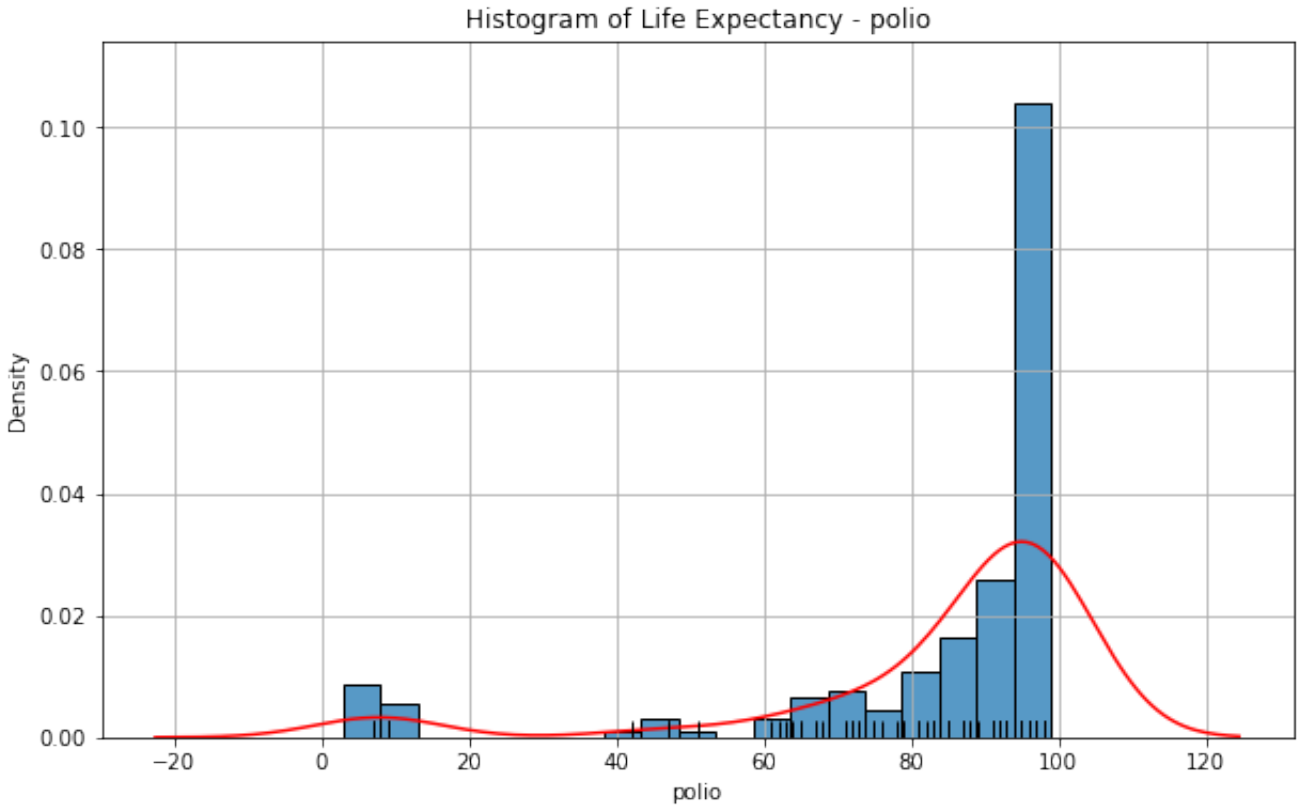
In [17]:
```
#histogram/denisty plot - bmi
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - bmi")
sns.histplot(df.bmi, stat = "density")
sns.kdeplot(df.bmi, color = "red")
sns.rugplot(df.bmi, color = "black")

plt.grid()
```



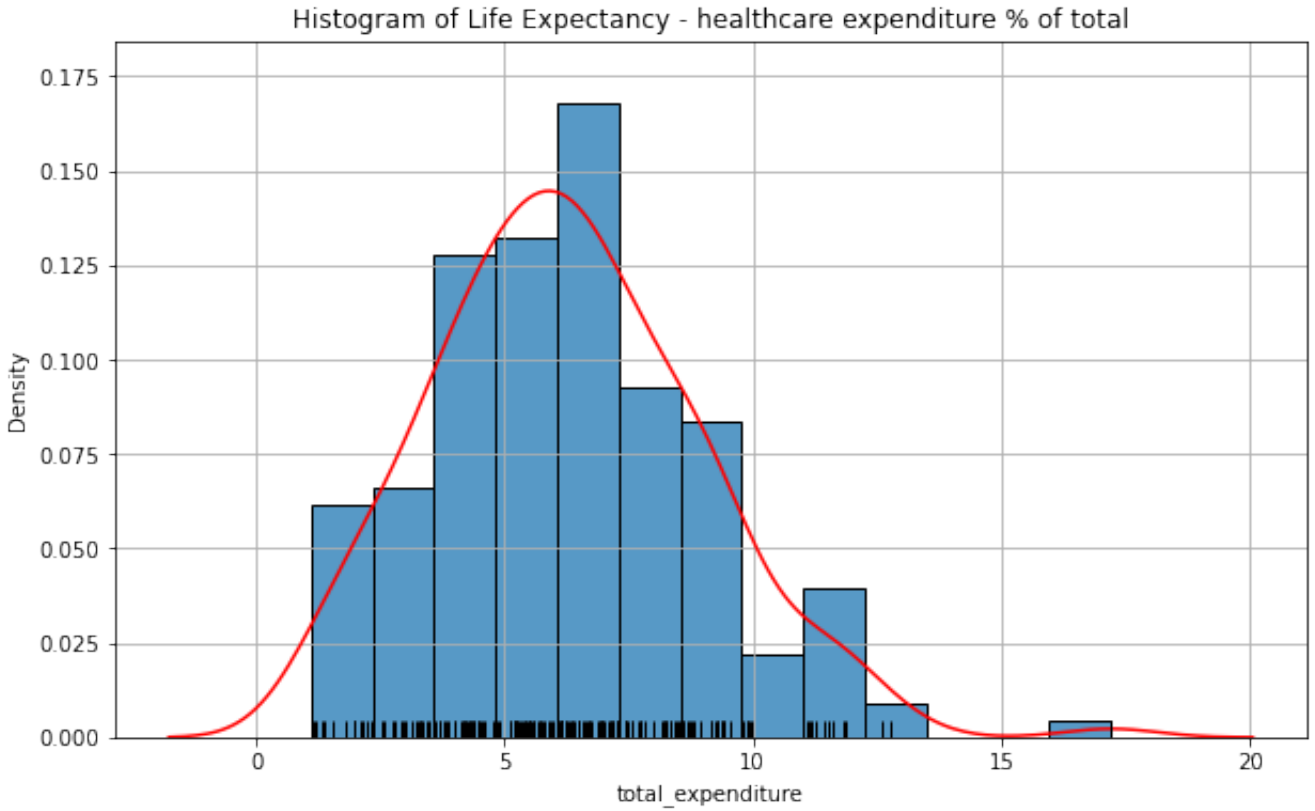
In [18]:
```
#histogram/denisty plot - under_five_deaths
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - under_five_deaths")
sns.histplot(df.under_five_deaths, stat = "density")
sns.kdeplot(df.under_five_deaths, color = "red")
sns.rugplot(df.under_five_deaths, color = "black")

plt.grid()
```



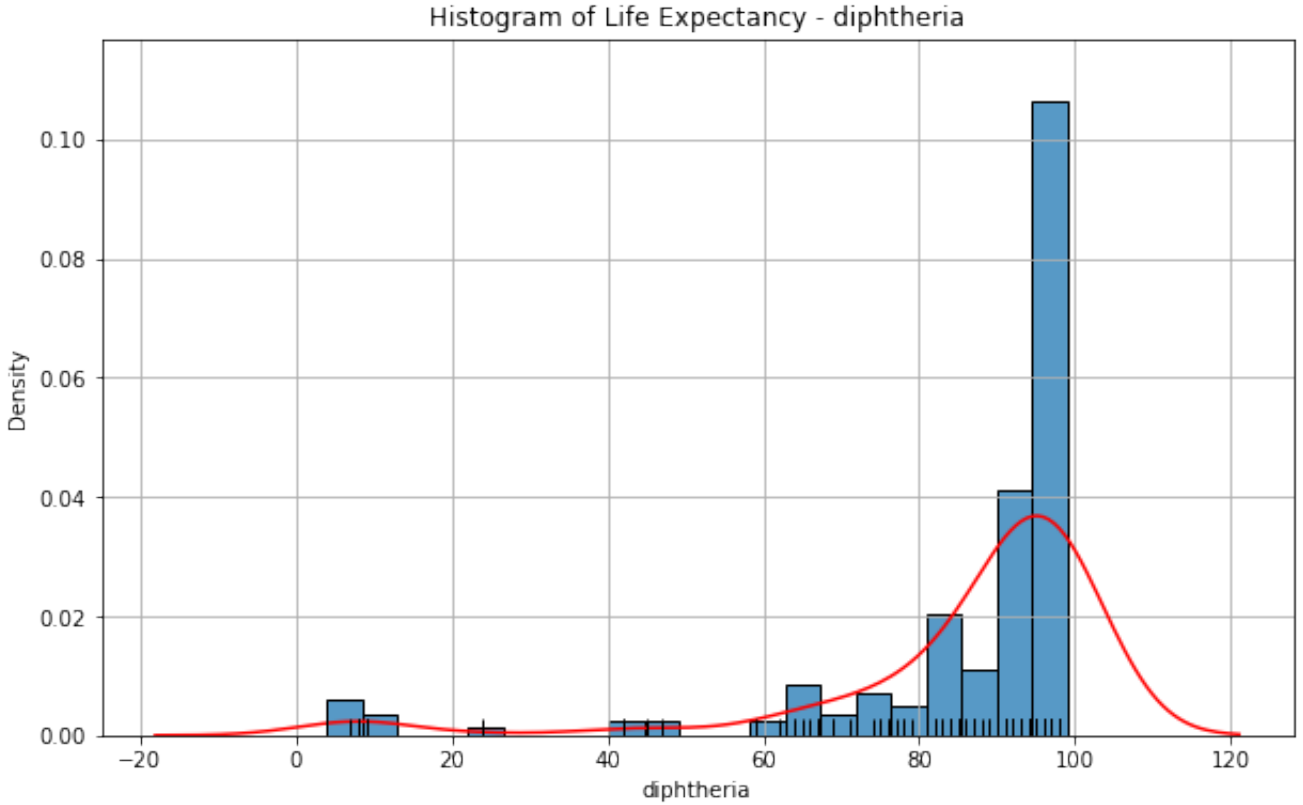
In [19]:
```
#histogram/denisty plot - polio
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - polio")
sns.histplot(df.polio, stat = "density")
sns.kdeplot(df.polio, color = "red")
sns.rugplot(df.polio, color = "black")

plt.grid()
```



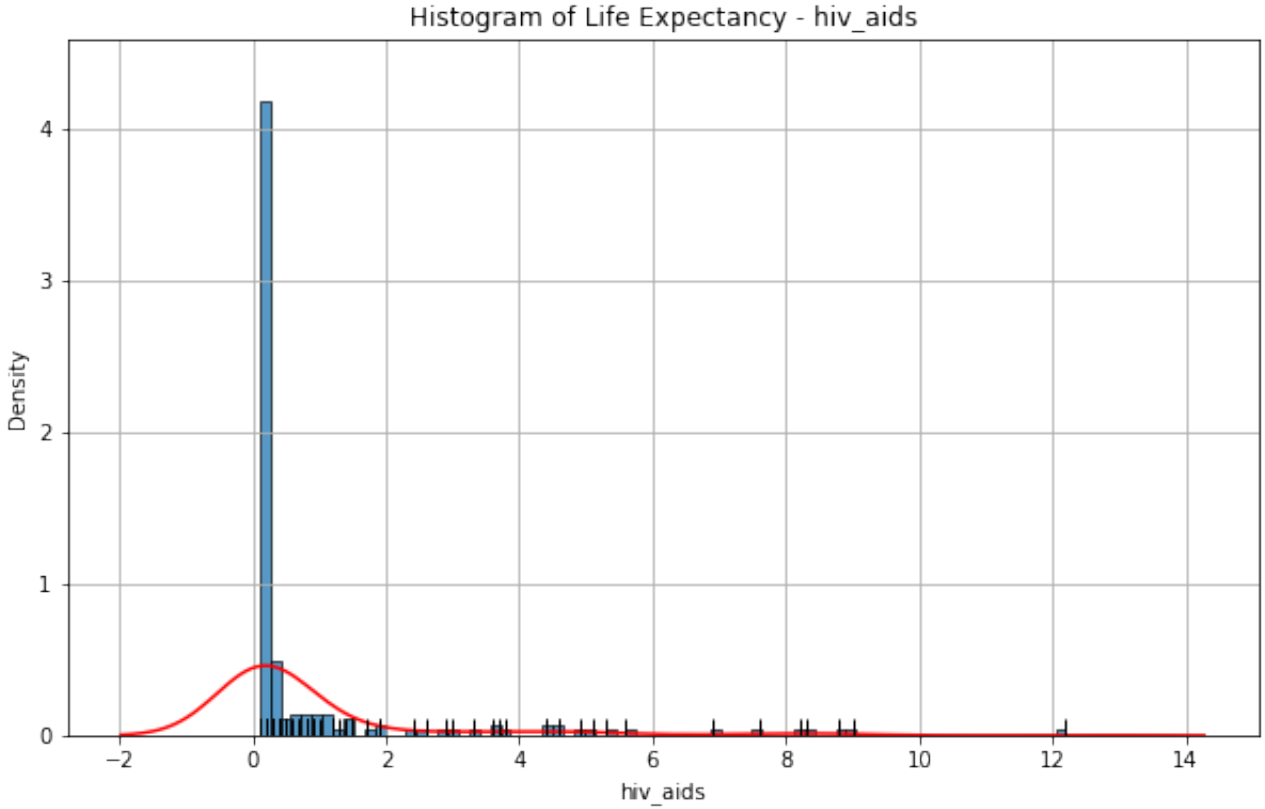
In [20]:
```
#histogram/denisty plot - total_expenditure
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - healthcare expenditure % of total")
sns.histplot(df.total_expenditure, stat = "density")
sns.kdeplot(df.total_expenditure, color = "red")
sns.rugplot(df.total_expenditure, color = "black")

plt.grid()
```



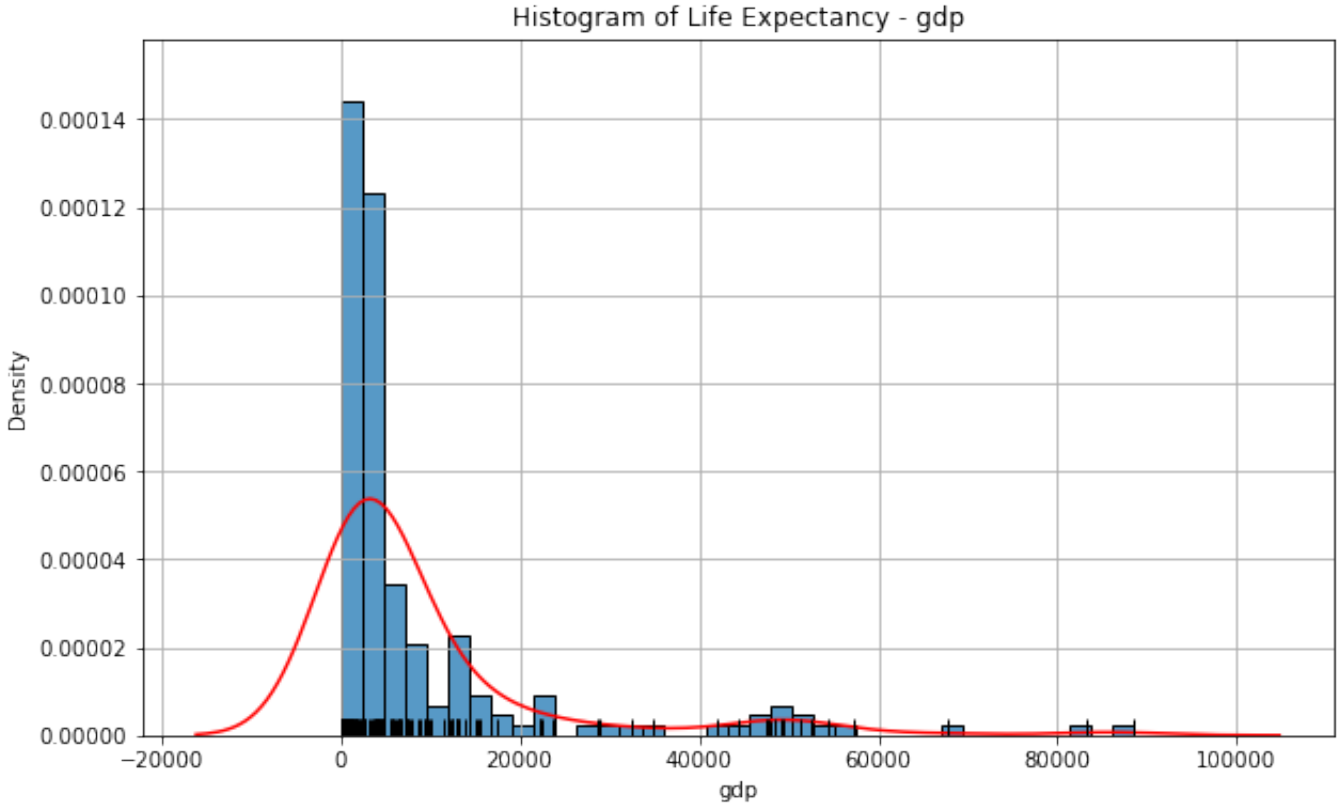
In [21]:
```
#histogram/denisty plot - diphtheria
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - diphtheria")
sns.histplot(df.diphtheria, stat = "density")
sns.kdeplot(df.diphtheria, color = "red")
sns.rugplot(df.diphtheria, color = "black")

plt.grid()
```



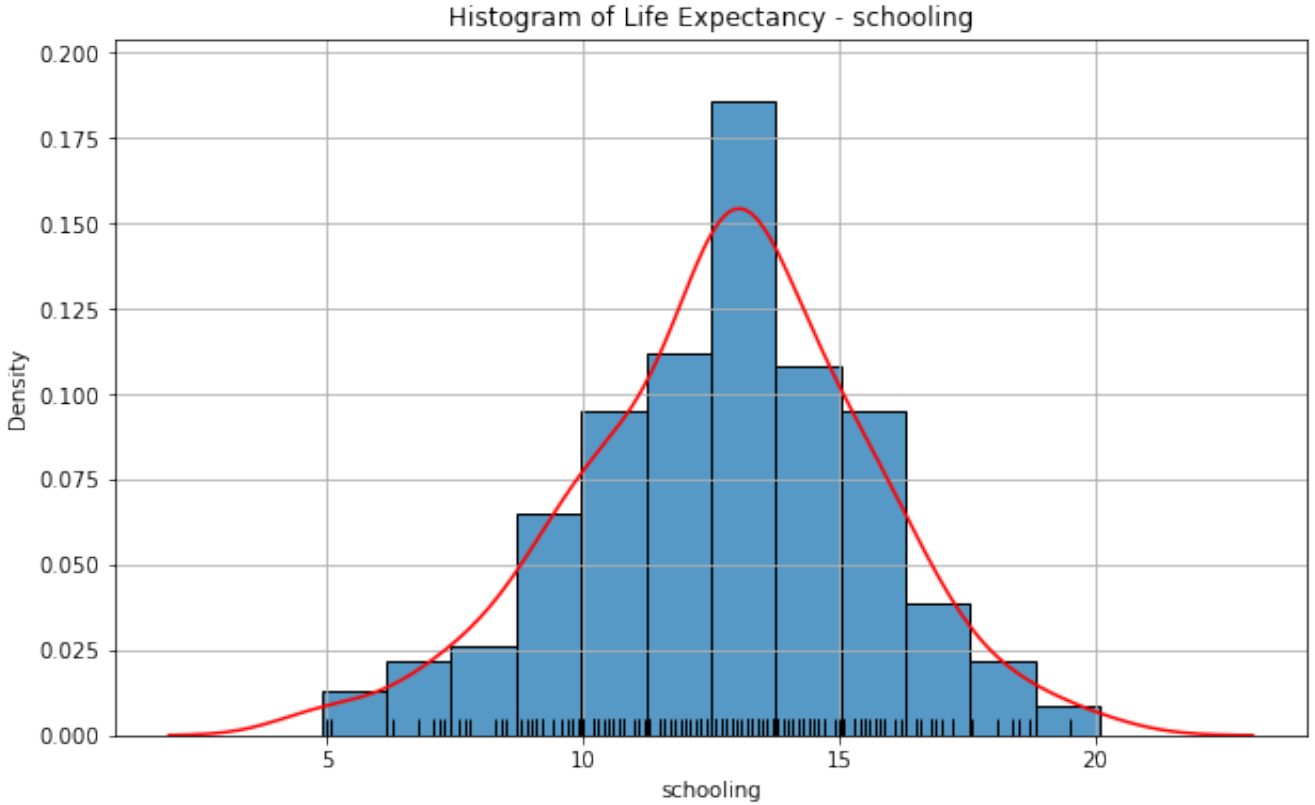
In [22]:
```
#histogram/denisty plot - hiv_aids
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - hiv_aids")
sns.histplot(df.hiv_aids, stat = "density")
sns.kdeplot(df.hiv_aids, color = "red")
sns.rugplot(df.hiv_aids, color = "black")

plt.grid()
```



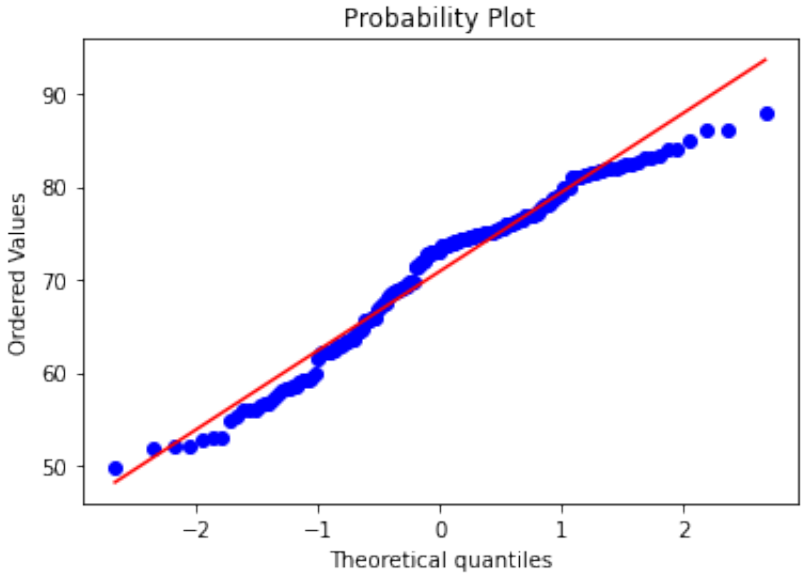
In [23]:
```
#histogram/denisty plot - gdp
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - gdp")
sns.histplot(df.gdp, stat = "density")
sns.kdeplot(df.gdp, color = "red")
sns.rugplot(df.gdp, color = "black")

plt.grid()
```



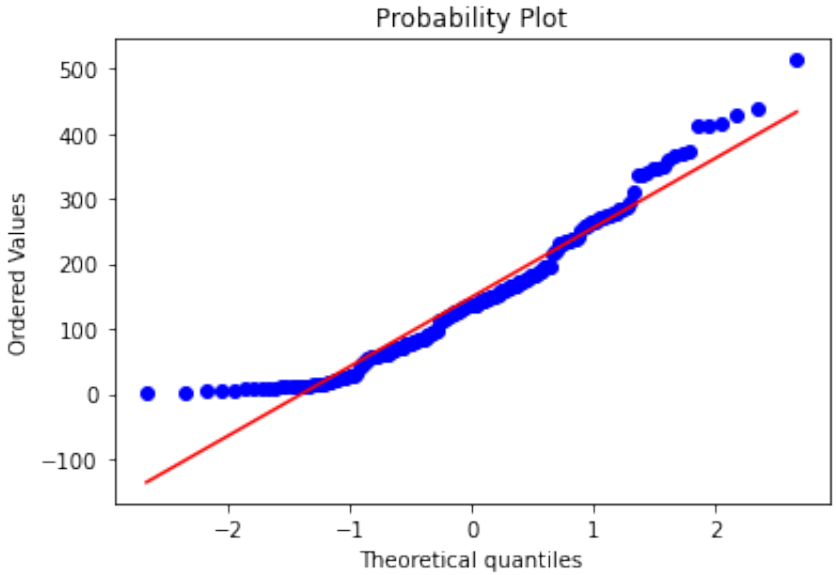
In [24]:
```
#histogram/denisty plot - schooling
plt.figure(figsize=(10,6))
plt.title("Histogram of Life Expectancy - schooling")
sns.histplot(df.schooling, stat = "density")
sns.kdeplot(df.schooling, color = "red")
sns.rugplot(df.schooling, color = "black")

plt.grid()
```



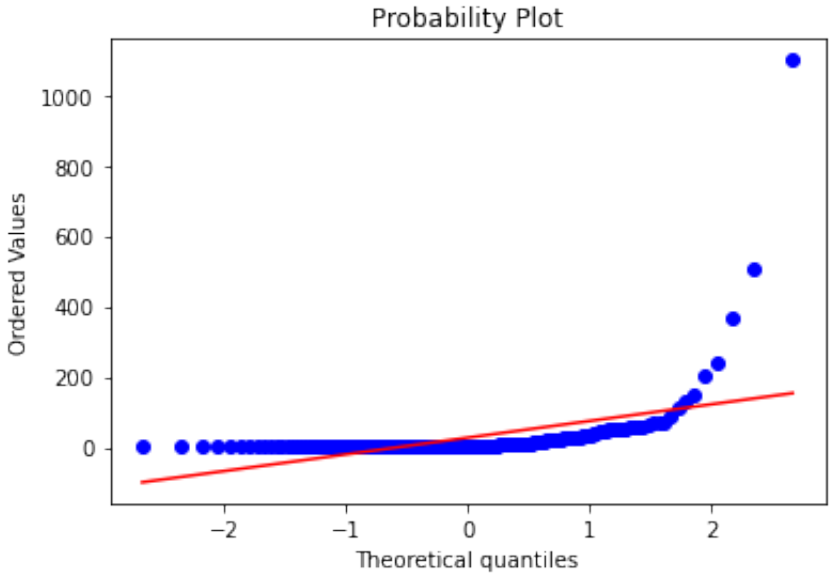
In [25]:
```
#qq plot - life_expectancy

stats.probplot(df.life_expectancy, dist="norm", plot=plt)
plt.show()
```



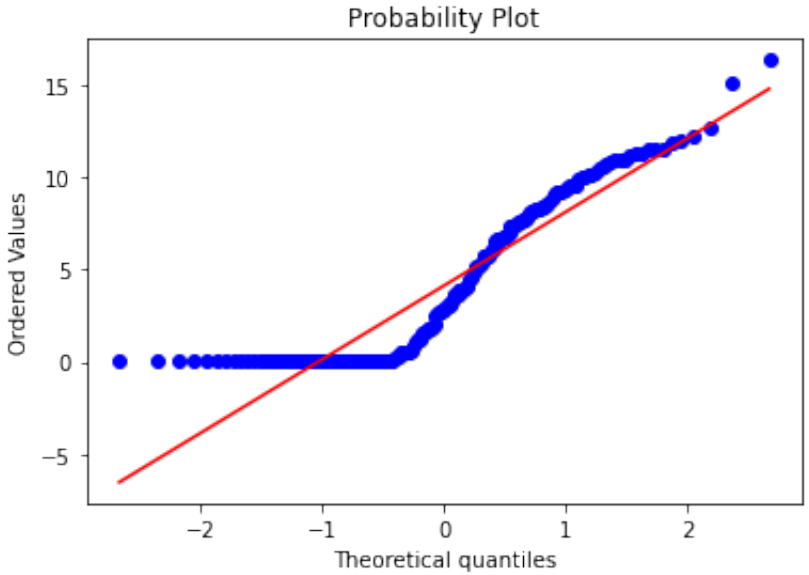
In [26]:
```
#qq plot - adult_mortality

stats.probplot(df.adult_mortality, dist="norm", plot=plt)
plt.show()
```



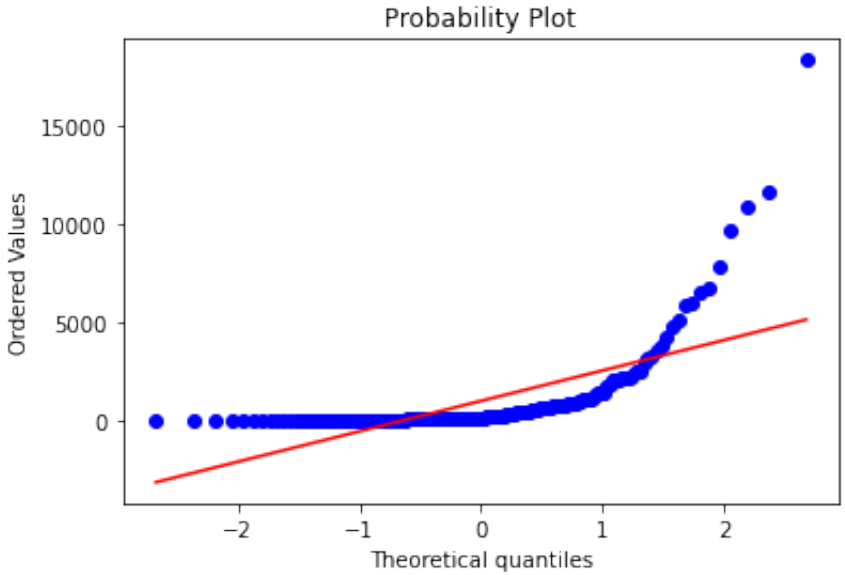
In [27]:
```
#qq plot - infant_deaths

stats.probplot(df.infant_deaths, dist="norm", plot=plt)
plt.show()
```



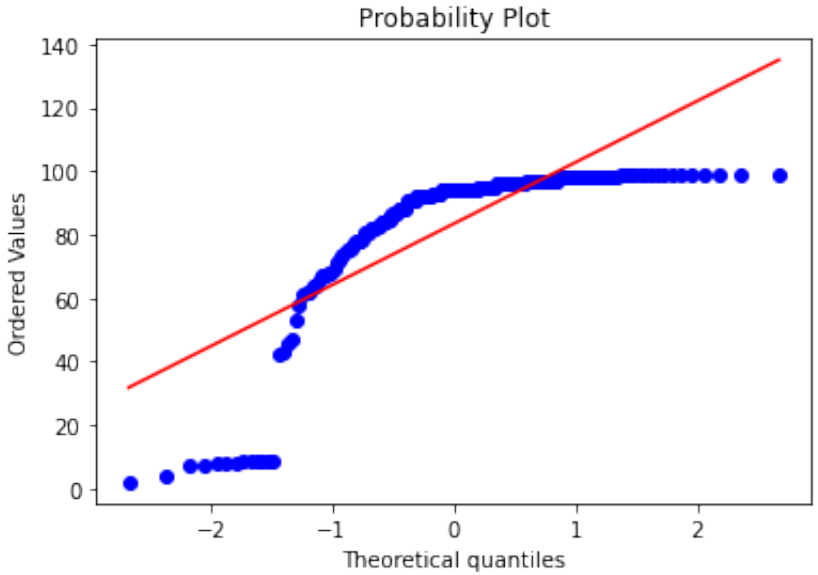
In [28]:
```
#qq plot - alcohol

stats.probplot(df.alcohol, dist="norm", plot=plt)
plt.show()
```



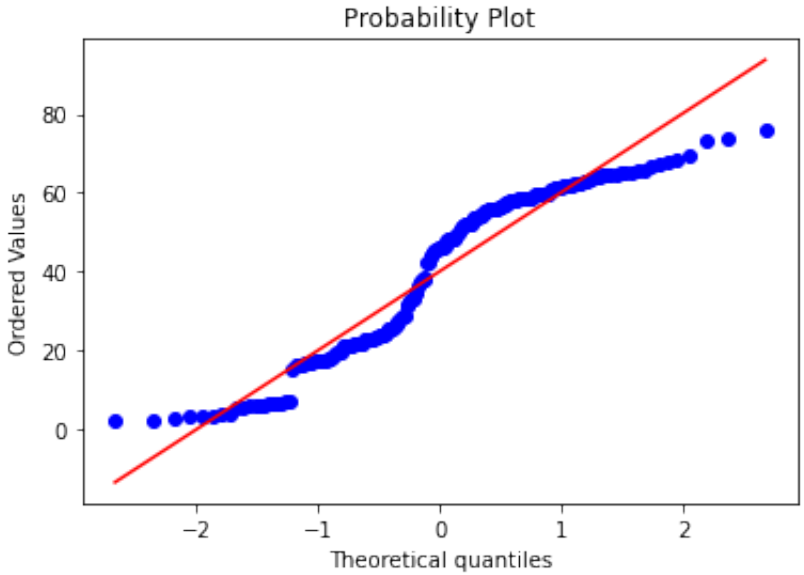
In [29]:
```
#qq plot - percentage_expenditure

stats.probplot(df.percentage_expenditure, dist="norm", plot=plt)
plt.show()
```



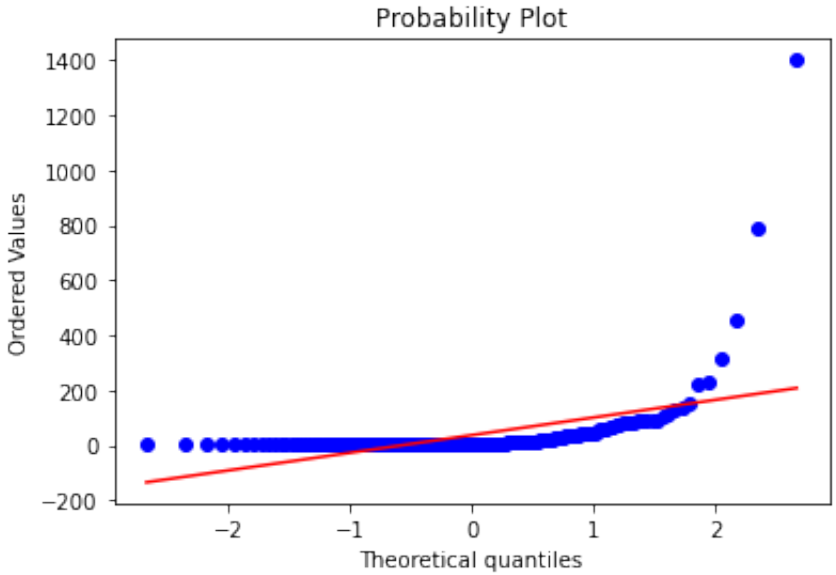
In [30]:
```
#qq plot - hepatitis_b

stats.probplot(df.hepatitis_b, dist="norm", plot=plt)
plt.show()
```



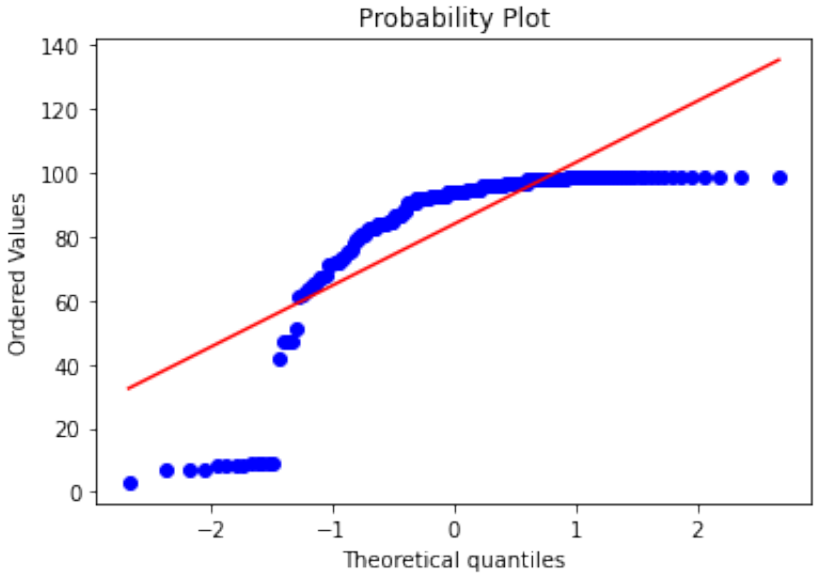
In [31]:
```
#qq plot - bmi

stats.probplot(df.bmi, dist="norm", plot=plt)
plt.show()
```



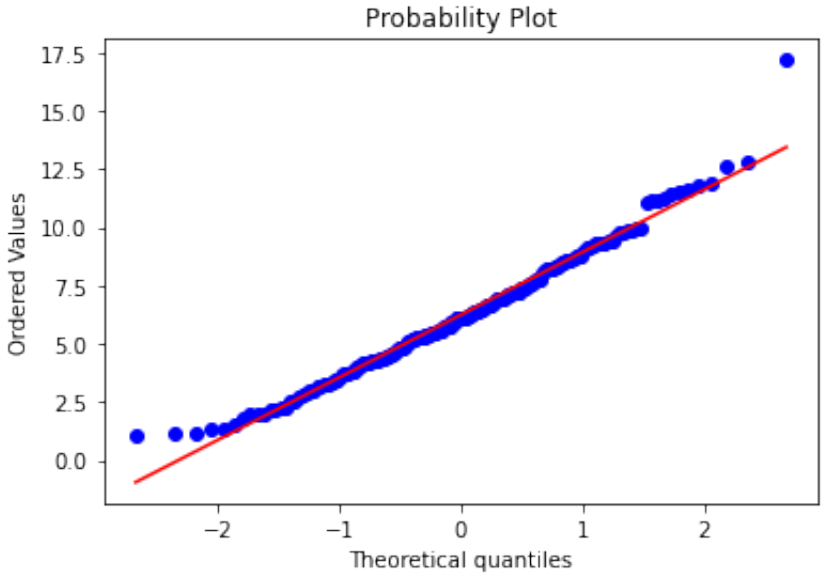
In [32]:
```
#qq plot - under_five_deaths

stats.probplot(df.under_five_deaths, dist="norm", plot=plt)
plt.show()
```



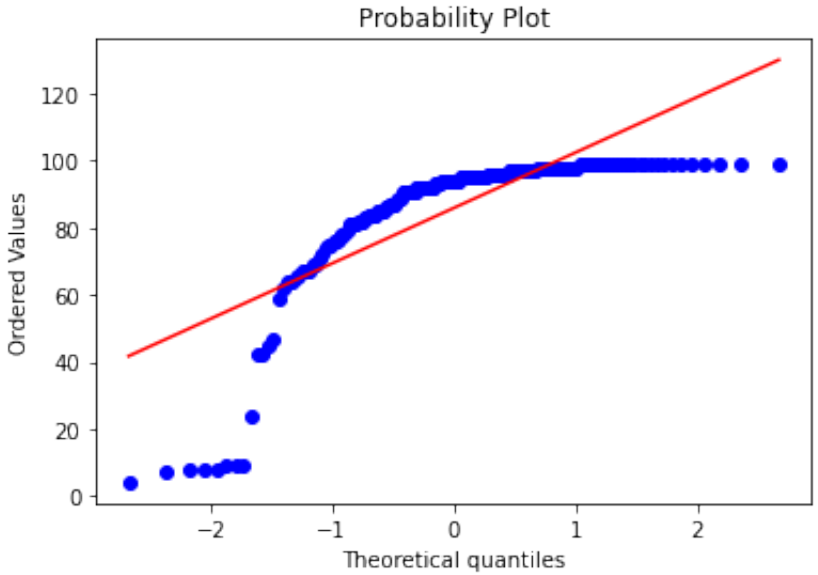
In [33]:
```
#qq plot -polio

stats.probplot(df.polio, dist="norm", plot=plt)
plt.show()
```



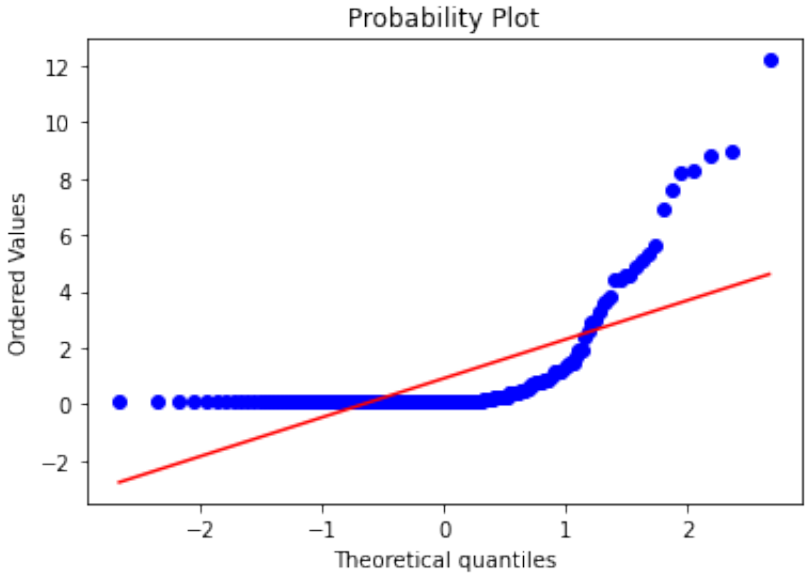
In [34]:
```
#qq plot - total_expenditure

stats.probplot(df.total_expenditure, dist="norm", plot=plt)
plt.show()
```



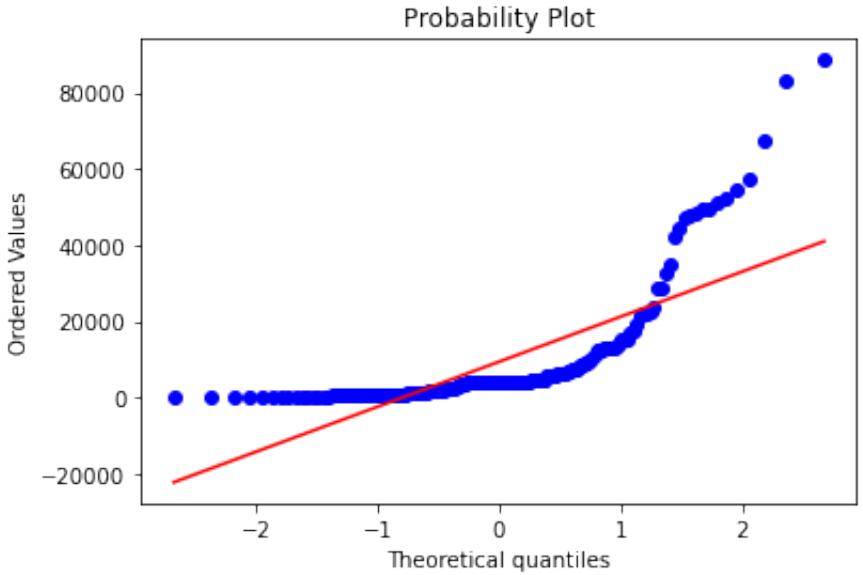
In [35]:
```
#qq plot - diphtheria

stats.probplot(df.diphtheria, dist="norm", plot=plt)
plt.show()
```



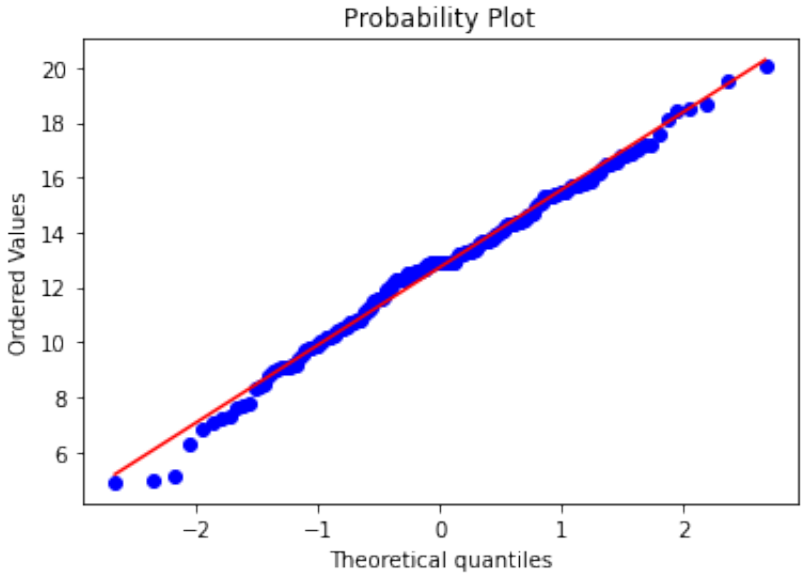
In [36]:
```
#qq plot - hiv_aids

stats.probplot(df.hiv_aids, dist="norm", plot=plt)
plt.show()
```



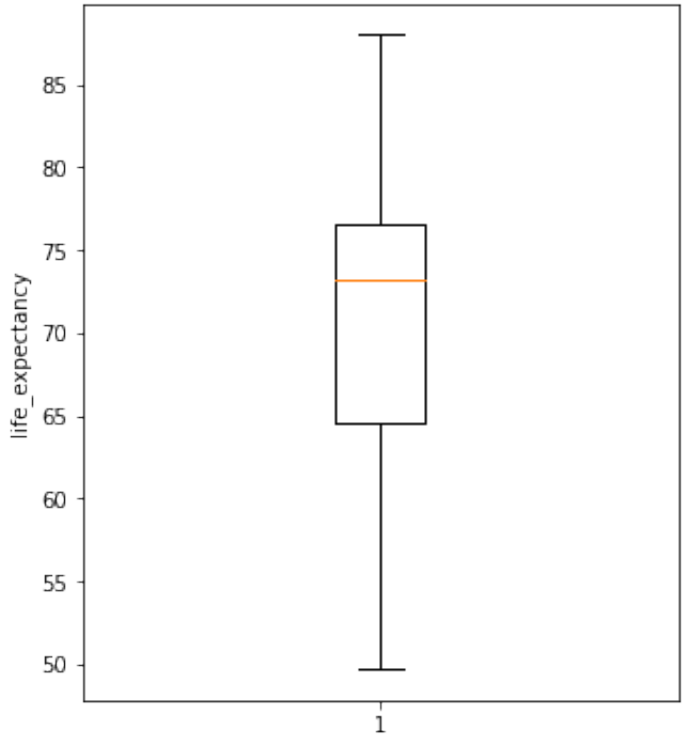
In [37]:
```
#qq plot - gdp

stats.probplot(df.gdp, dist="norm", plot=plt)
plt.show()
```



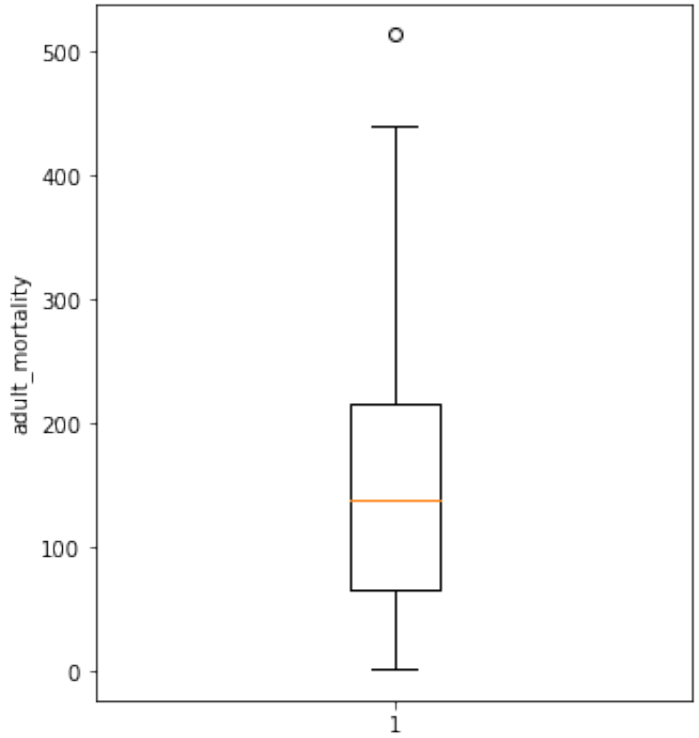
In [38]:
```
#qq plot - schooling

stats.probplot(df.schooling, dist="norm", plot=plt)
plt.show()
```



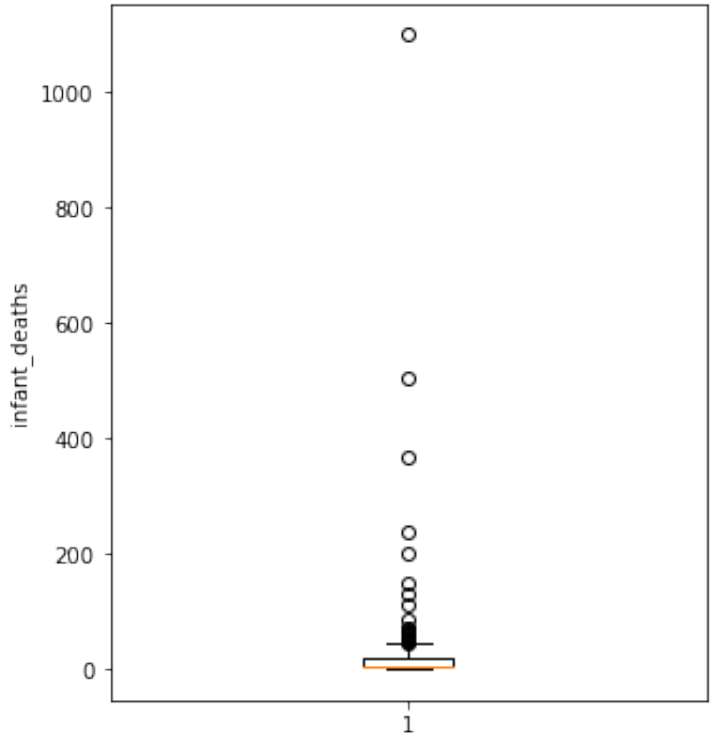
In [39]:
```
# boxplot - life_expectancy
plt.figure(figsize=(5,6))
plt.boxplot(df.life_expectancy)
plt.ylabel("life_expectancy")
plt.show()
```



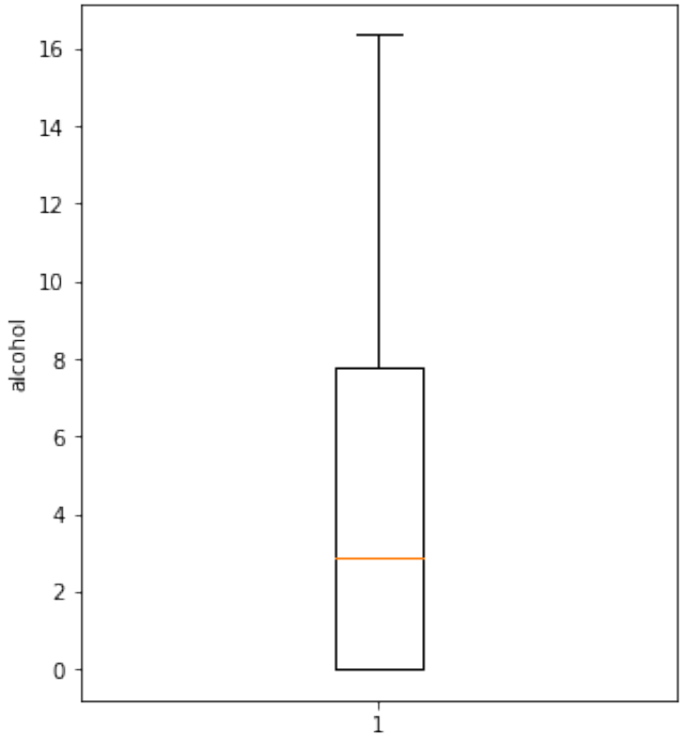
In [40]:
```
# boxplot - adult_mortality
plt.figure(figsize=(5,6))
plt.boxplot(df.adult_mortality)
plt.ylabel("adult_mortality")
plt.show()
```



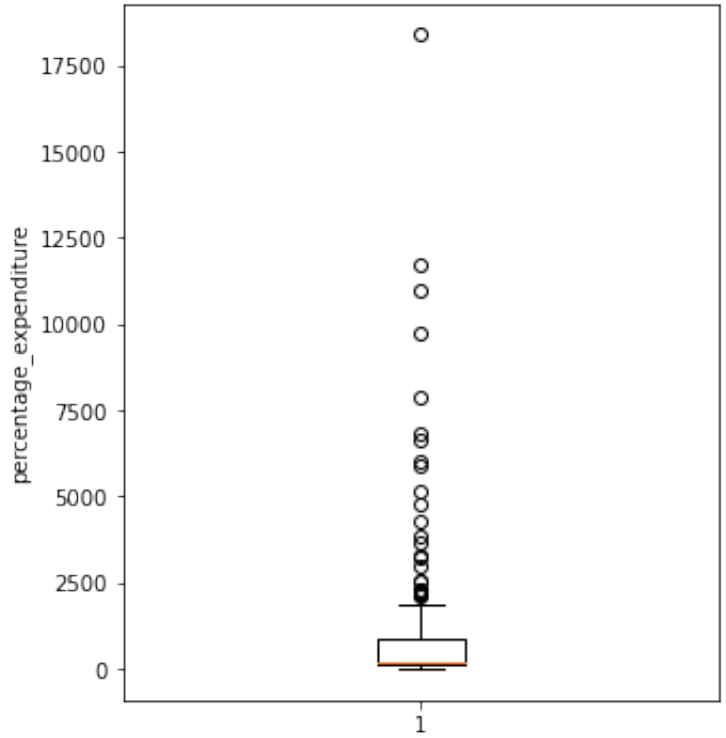
In [41]:
```
# boxplot - infant_deaths
plt.figure(figsize=(5,6))
plt.boxplot(df.infant_deaths)
plt.ylabel("infant_deaths")
plt.show()
```



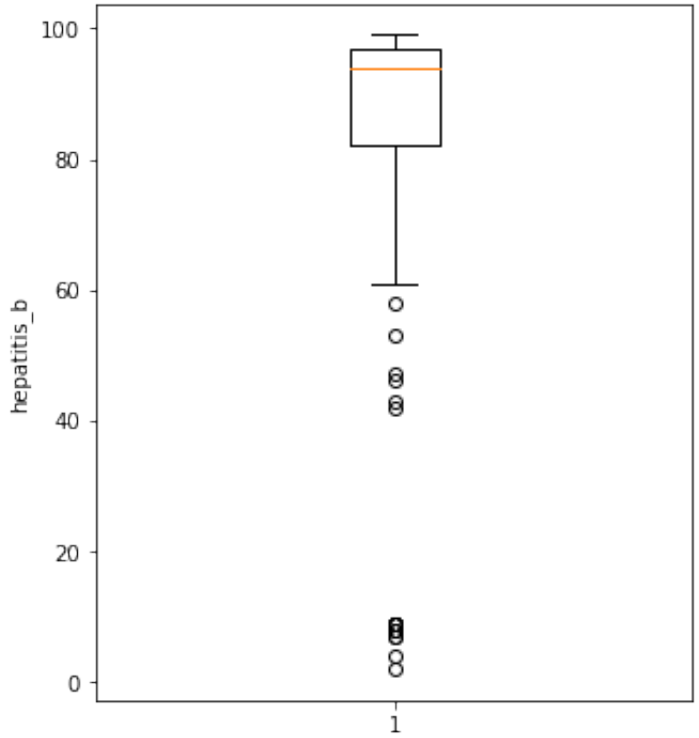
In [42]:
```
# boxplot - alcohol
plt.figure(figsize=(5,6))
plt.boxplot(df.alcohol)
plt.ylabel("alcohol")
plt.show()
```



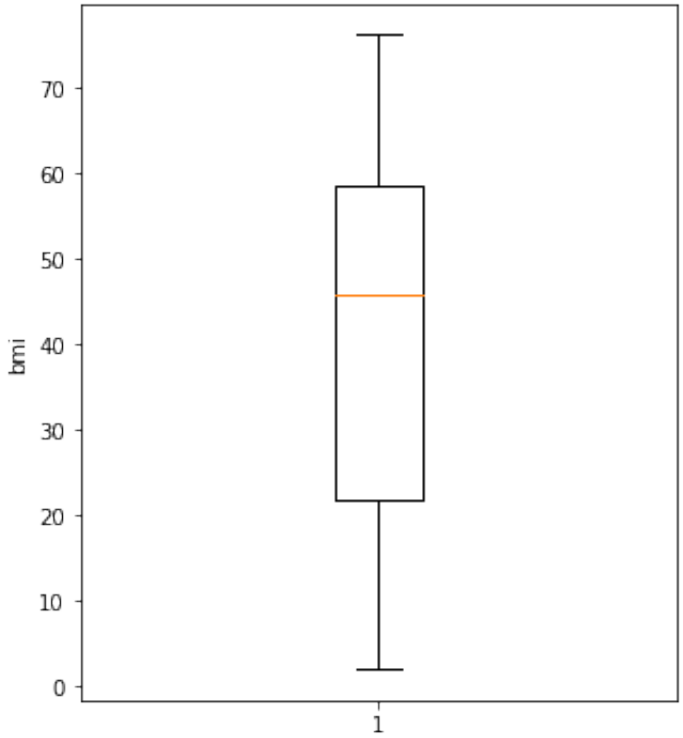
In [43]:
```
# boxplot - percentage_expenditure
plt.figure(figsize=(5,6))
plt.boxplot(df.percentage_expenditure)
plt.ylabel("percentage_expenditure")
plt.show()
```



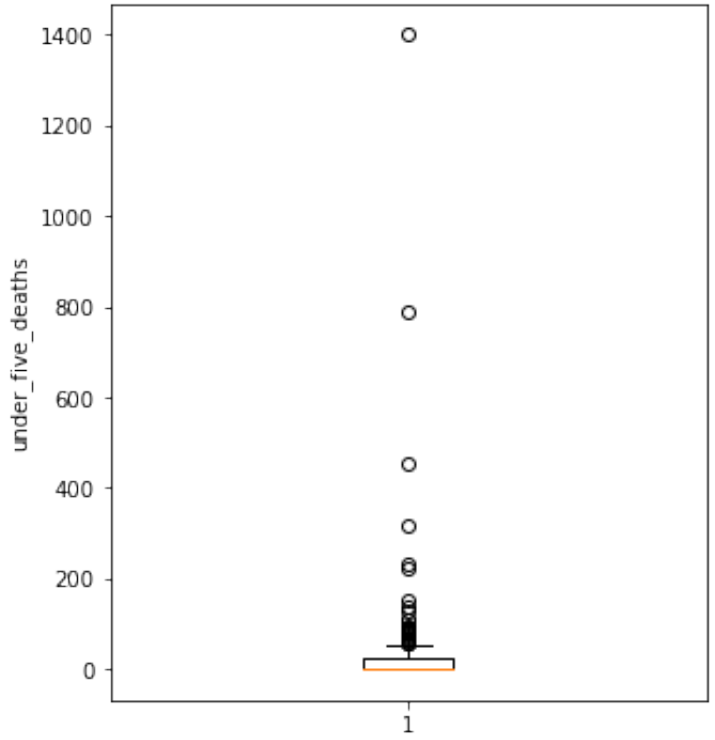
In [44]:
```
# boxplot - hepatitis_b
plt.figure(figsize=(5,6))
plt.boxplot(df.hepatitis_b)
plt.ylabel("hepatitis_b")
plt.show()
```



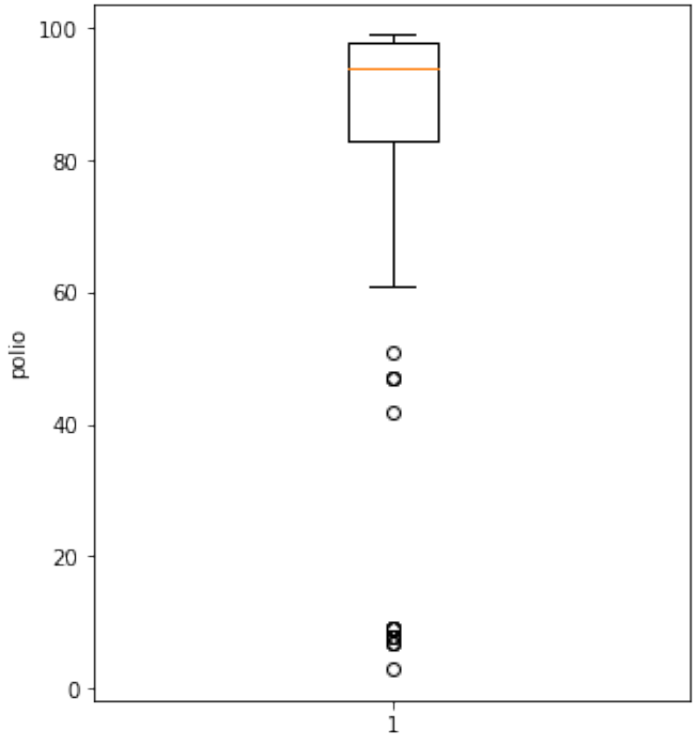
In [45]:
```
# boxplot - bmi
plt.figure(figsize=(5,6))
plt.boxplot(df.bmi)
plt.ylabel("bmi")
plt.show()
```



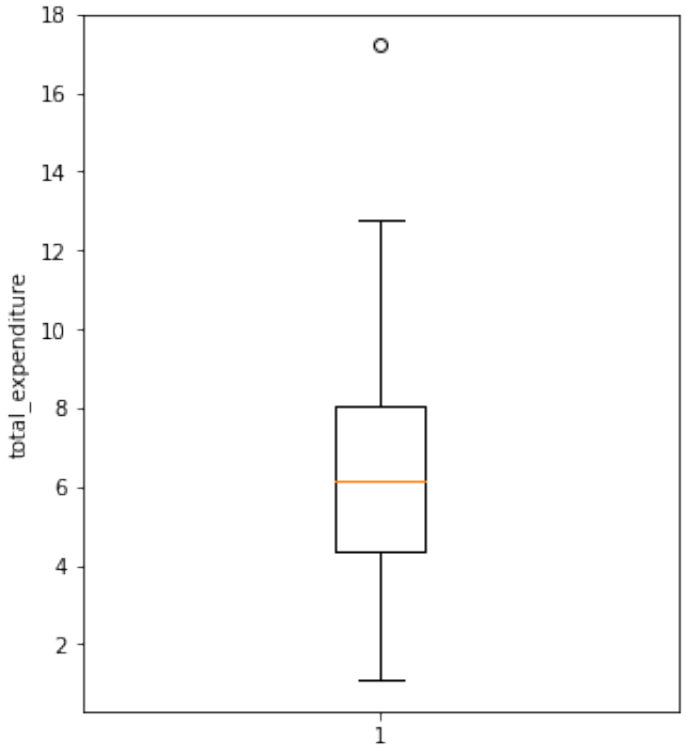
In [46]:
```
# boxplot - under_five_deaths
plt.figure(figsize=(5,6))
plt.boxplot(df.under_five_deaths)
plt.ylabel("under_five_deaths")
plt.show()
```



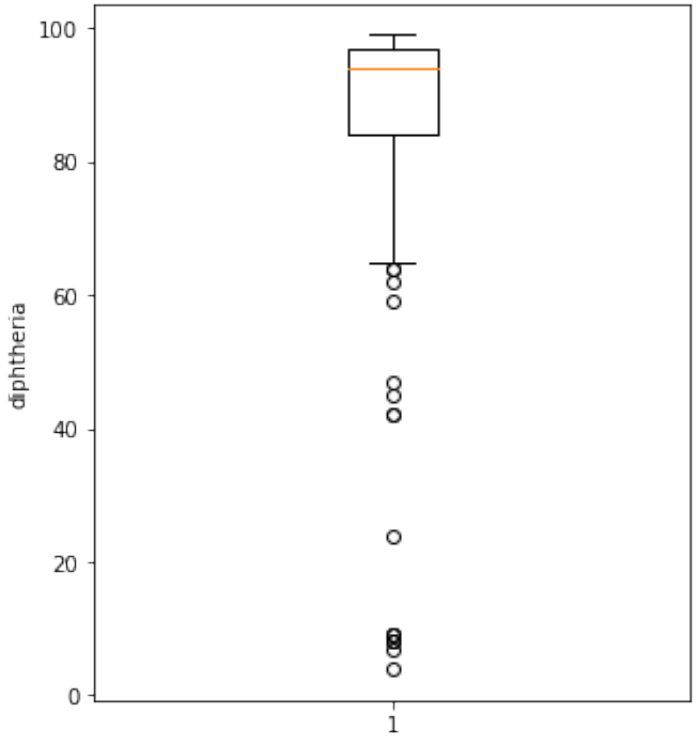
In [47]:
```
# boxplot - polio
plt.figure(figsize=(5,6))
plt.boxplot(df.polio)
plt.ylabel("polio")
plt.show()
```



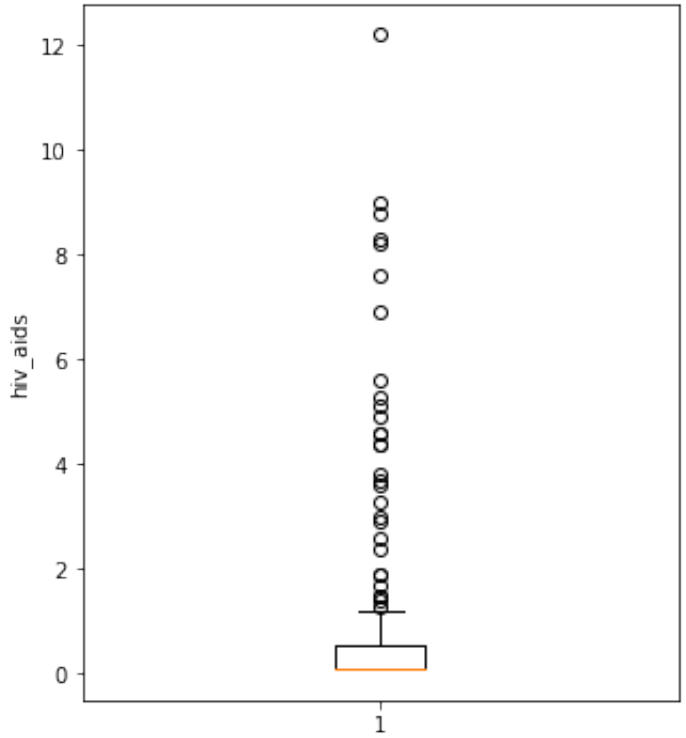
In [48]:
```
# boxplot - total_expenditure
plt.figure(figsize=(5,6))
plt.boxplot(df.total_expenditure)
plt.ylabel("total_expenditure")
plt.show()
```



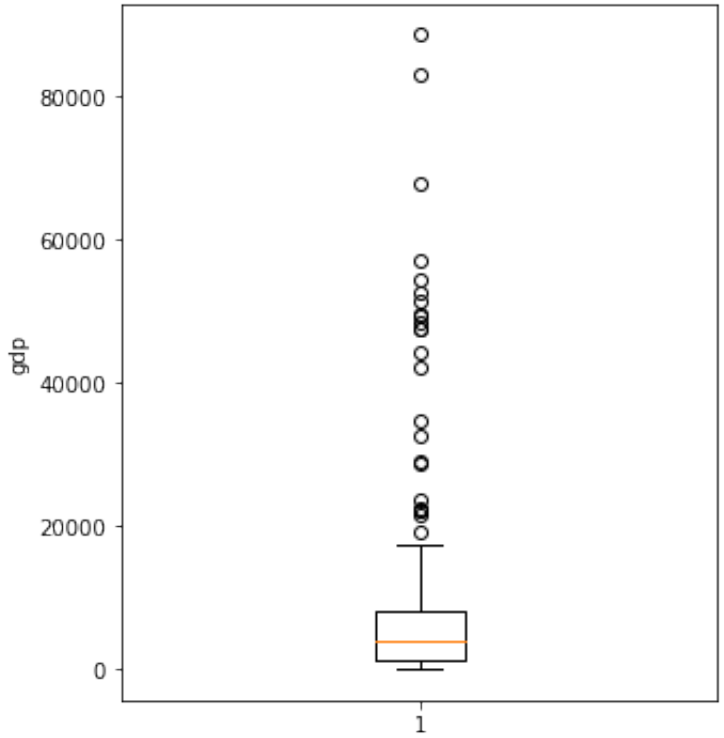
In [49]:
```
# boxplot - diphtheria
plt.figure(figsize=(5,6))
plt.boxplot(df.diphtheria)
plt.ylabel("diphtheria")
plt.show()
```



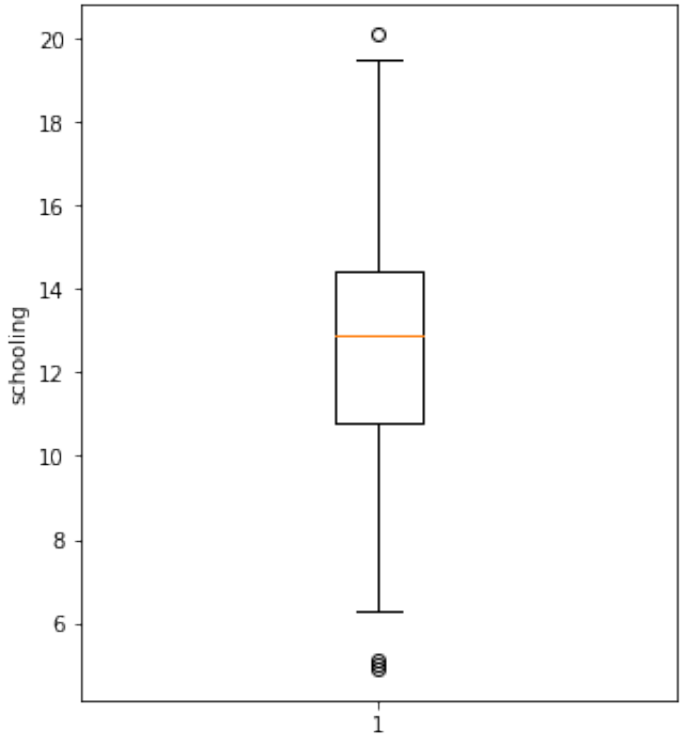
In [50]:
```
# boxplot - hiv_aids
plt.figure(figsize=(5,6))
plt.boxplot(df.hiv_aids)
plt.ylabel("hiv_aids")
plt.show()
```



In [51]:
```
# boxplot - gdp
plt.figure(figsize=(5,6))
plt.boxplot(df.gdp)
plt.ylabel("gdp")
plt.show()
```



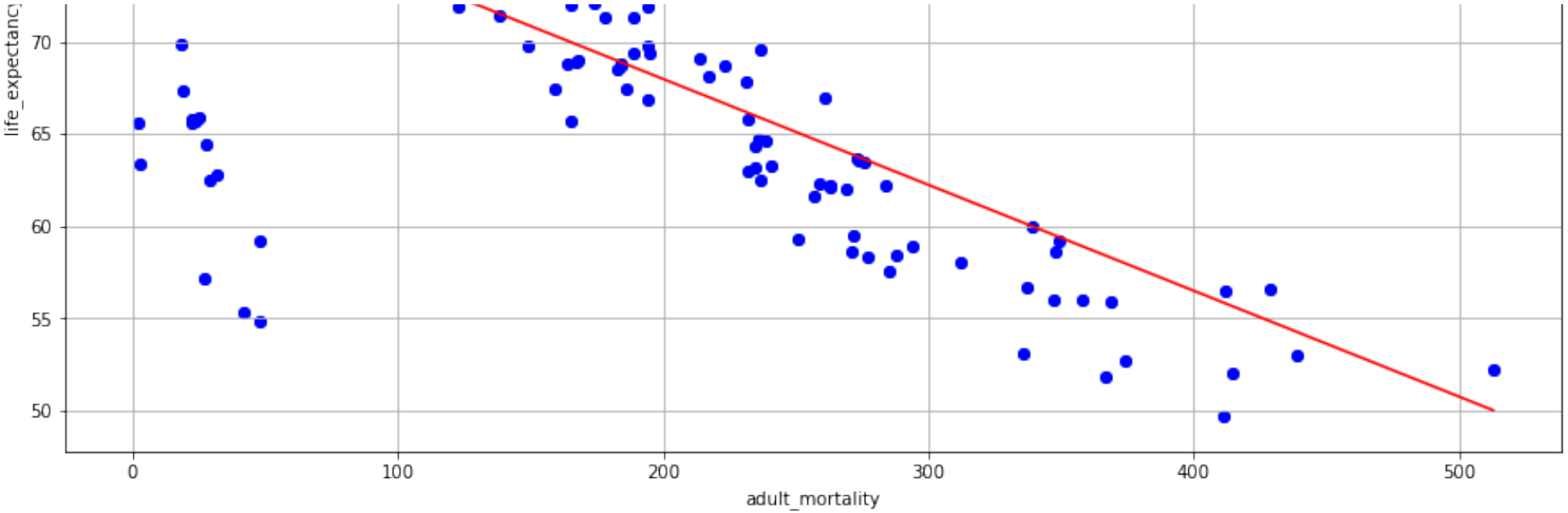
In [52]:
```
# boxplot - schooling
plt.figure(figsize=(5,6))
plt.boxplot(df.schooling)
plt.ylabel("schooling")
plt.show()
```



In [53]:

```
#scatterplot - life_expectancy and adult_mortality

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["adult_mortality"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["adult_mortality"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.adult_mortality.max(), 100)

# combining the two plots
plt.scatter(df["adult_mortality"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("adult_mortality")
plt.grid()
```

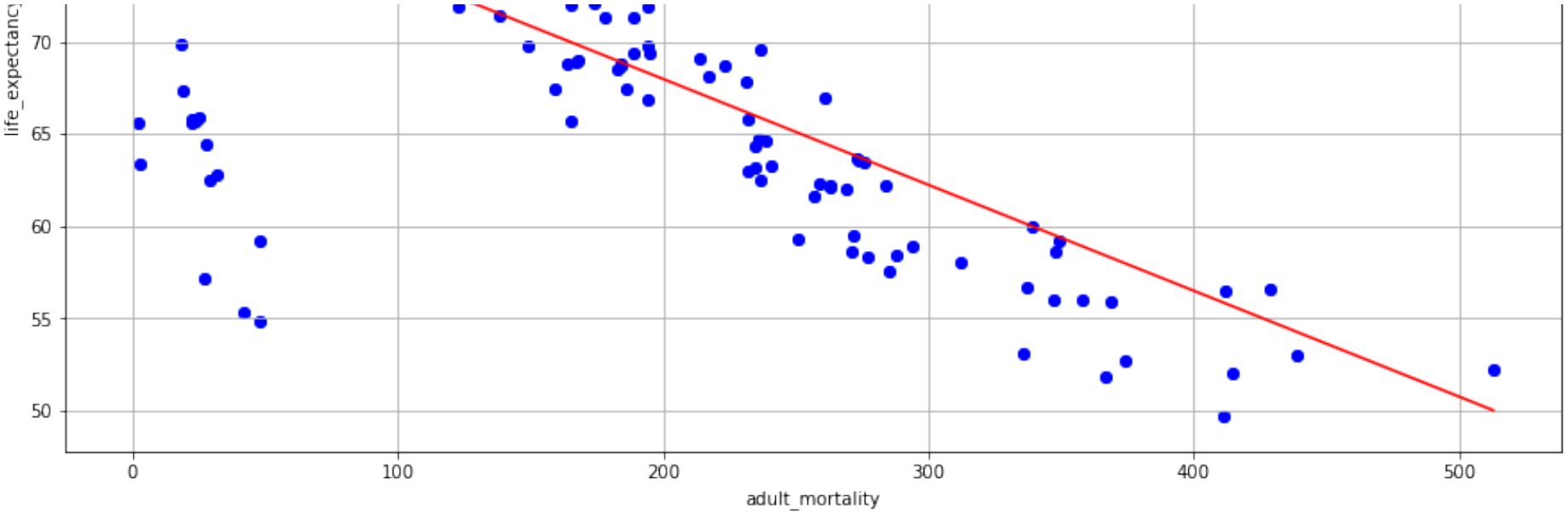
```
The slope of the regression line is: -0.05735733813441245 The Intercept is: 79.45127300580279
```


life_expectanc
70
65
60
55
50
0
100
200
300
400
500
adult_mortality

In [54]:

```
#scatterplot - life_expectancy and adult_mortality

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["adult_mortality"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["adult_mortality"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.adult_mortality.max(), 100)

# combining the two plots
plt.scatter(df["adult_mortality"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("adult_mortality")
plt.grid()
```

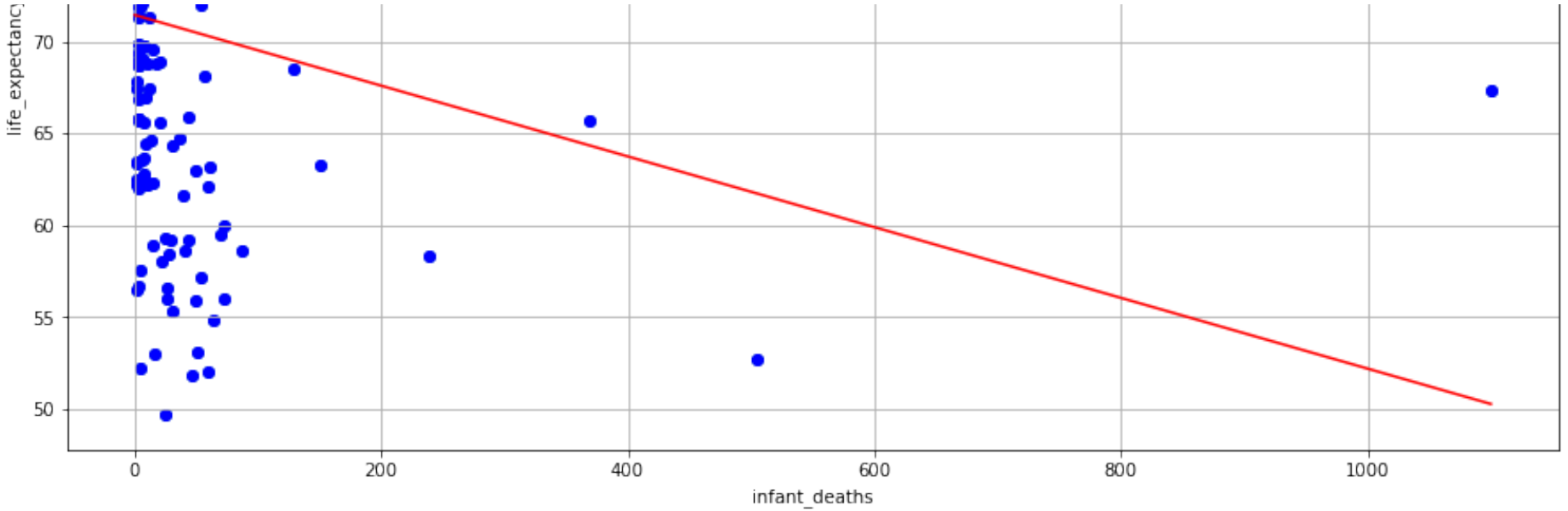
```
The slope of the regression line is: -0.05735733813441245 The Intercept is: 79.45127300580279
```


life_expectanc
70
65
60
55
50
0
100
200
300
400
500
adult_mortality

In [55]:

```
#scatterplot - life_expectancy and infant_deaths

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["infant_deaths"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["infant_deaths"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.infant_deaths.max(), 100)

# combining the two plots
plt.scatter(df["infant_deaths"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("infant_deaths")
plt.grid()
```

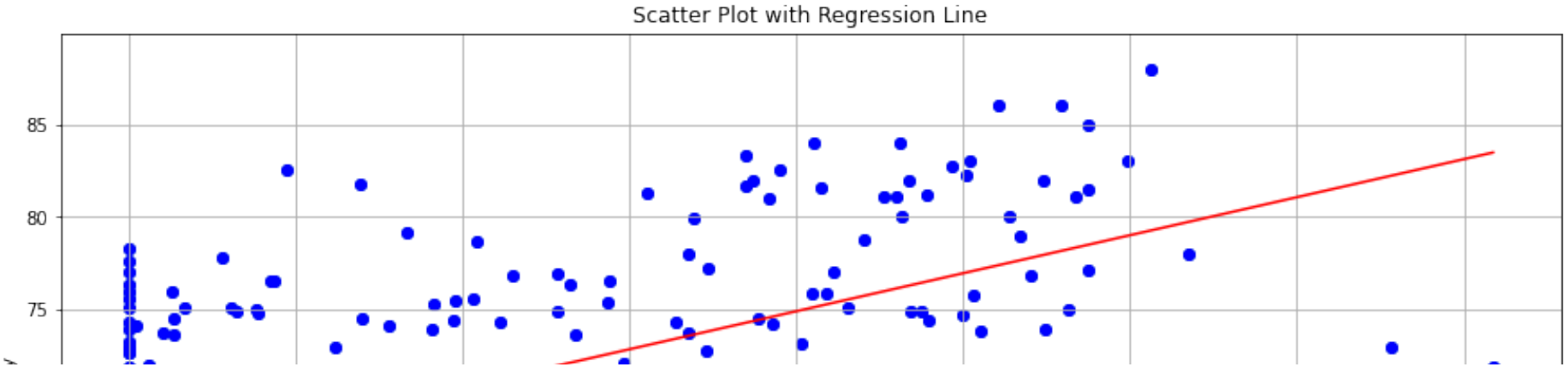
```
The slope of the regression line is: -0.01923701082112625 The Intercept is: 71.44127983593324
```

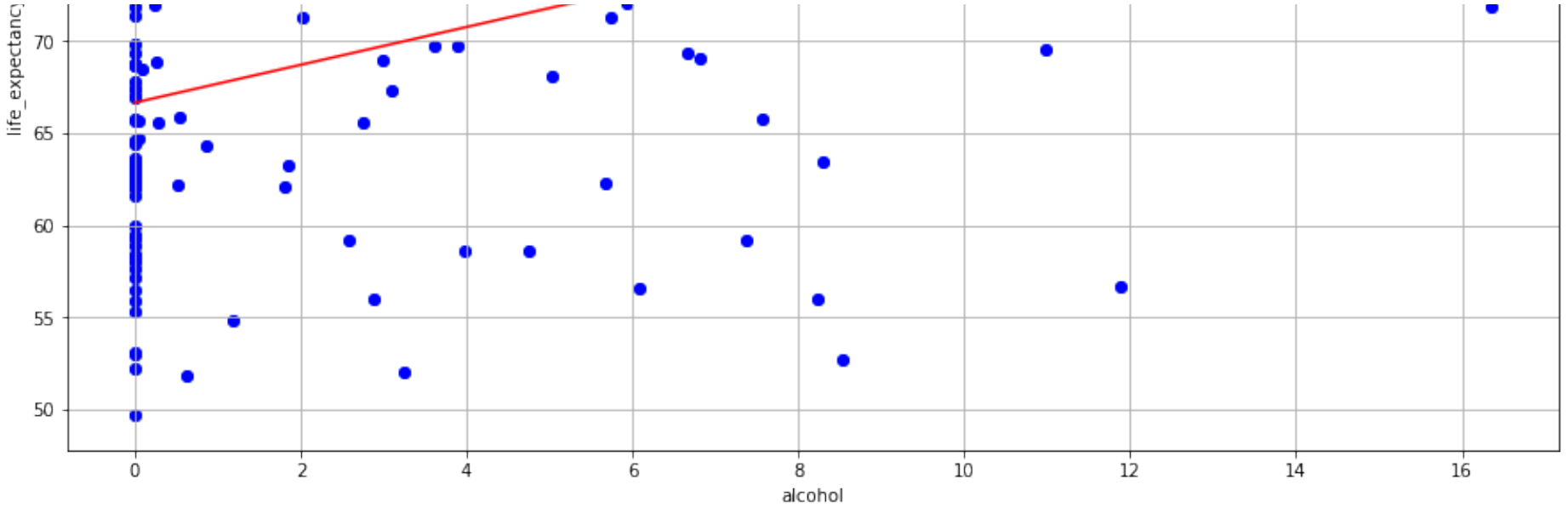

life_expectanc
70
65
60
55
50
0
200
400
600
800
1000
infant_deaths

In [56]:

```
#scatterplot - life_expectancy and alcohol

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["alcohol"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["alcohol"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.alcohol.max(), 100)

# combining the two plots
plt.scatter(df["alcohol"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("alcohol")
plt.grid()
```

```
The slope of the regression line is: 1.0276734476767146 The Intercept is: 66.67124462633056
```

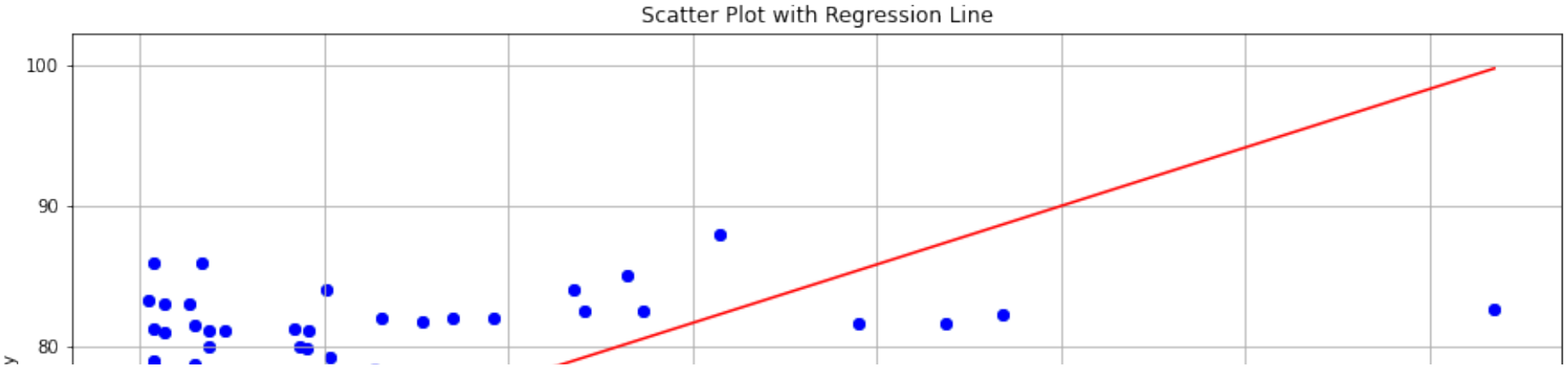


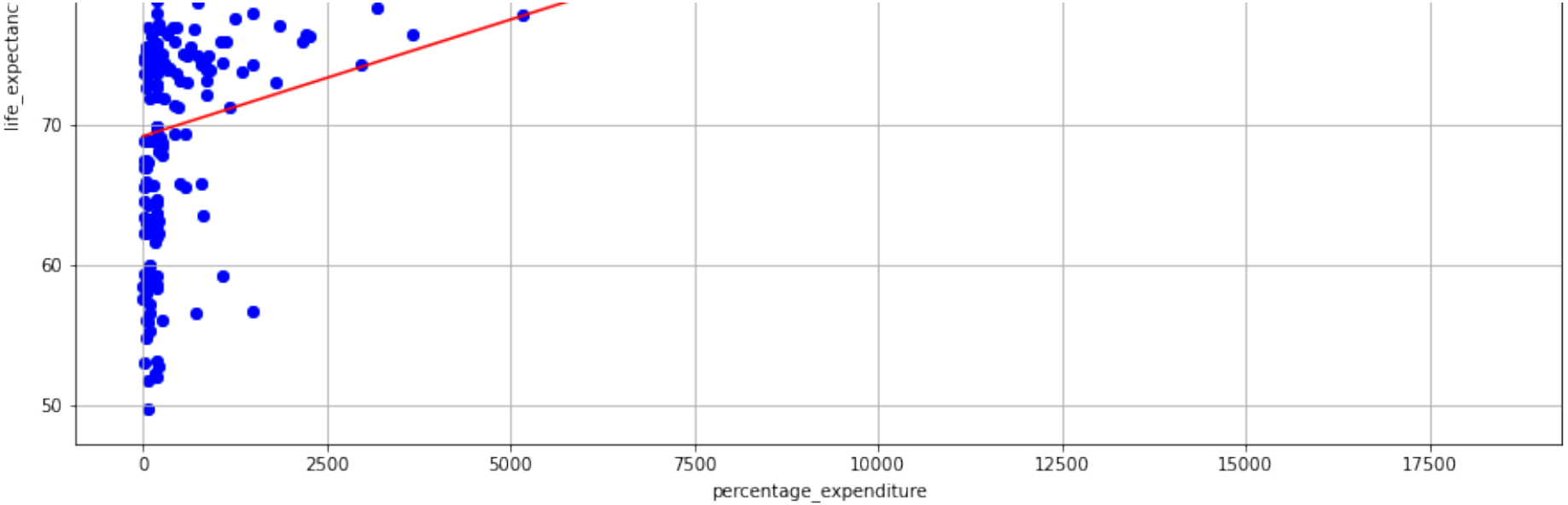


In [57]:

```
#scatterplot - life_expectancy and percentage_expenditure

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["percentage_expenditure"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["percentage_expenditure"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.percentage_expenditure.max(), 100)

# combining the two plots
plt.scatter(df["percentage_expenditure"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("percentage_expenditure")
plt.grid()
```

```
The slope of the regression line is: 0.0016651273896627907 The Intercept is: 69.18075006772337
```

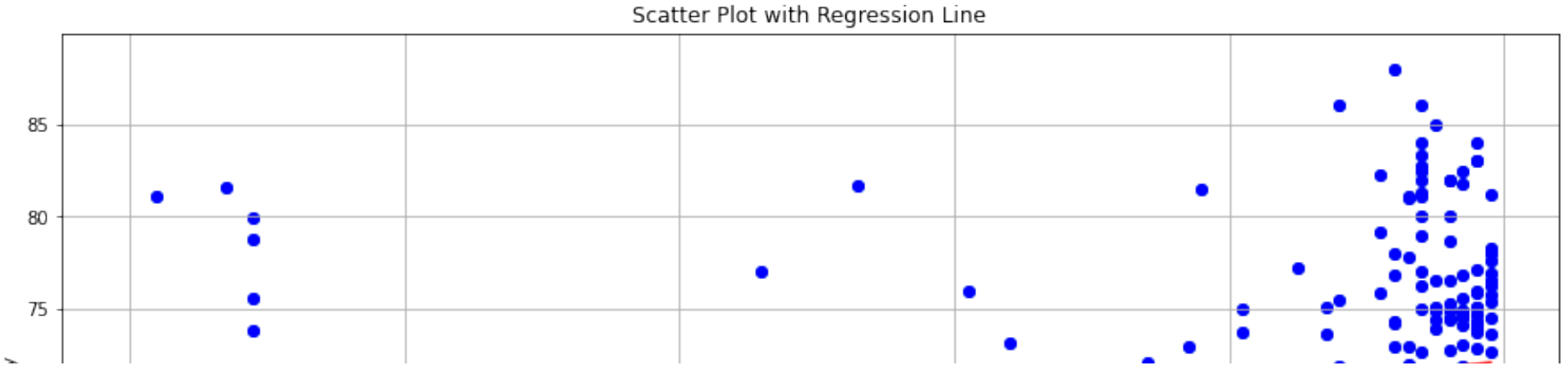


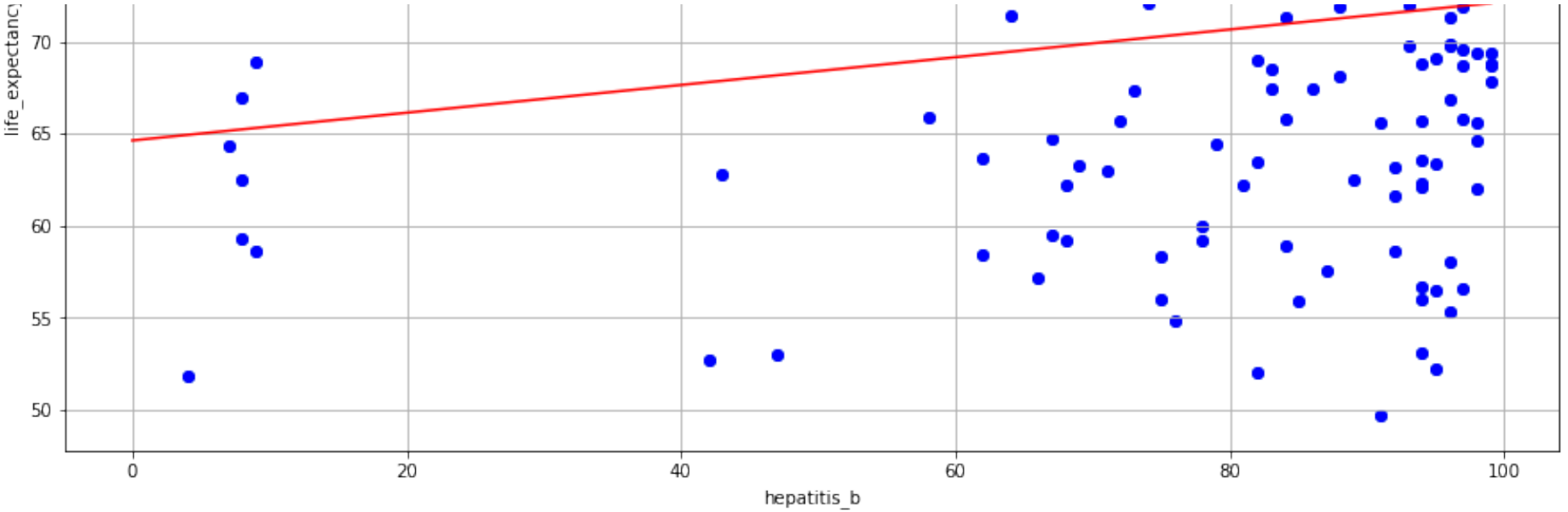

life_expectanc
70
60
50
0
2500
5000
7500
10000
12500
15000
17500
percentage_expenditure

In [58]:

```
#scatterplot - life_expectancy and hepatitis_b

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["hepatitis_b"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["hepatitis_b"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.hepatitis_b.max(), 100)

# combining the two plots
plt.scatter(df["hepatitis_b"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("hepatitis_b")
plt.grid()
```

```
The slope of the regression line is: 0.07532534787790952 The Intercept is: 64.63159528909465
```

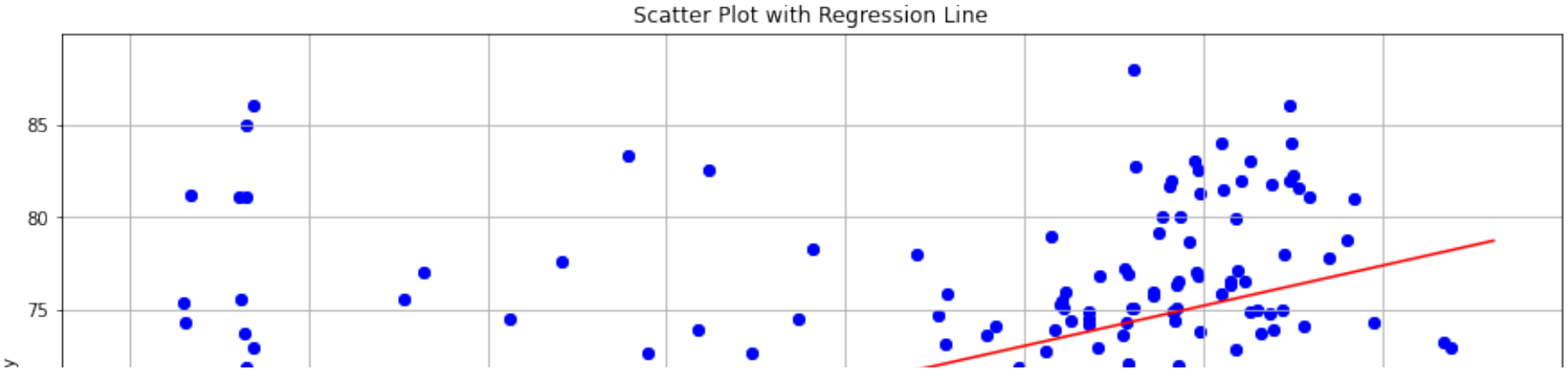


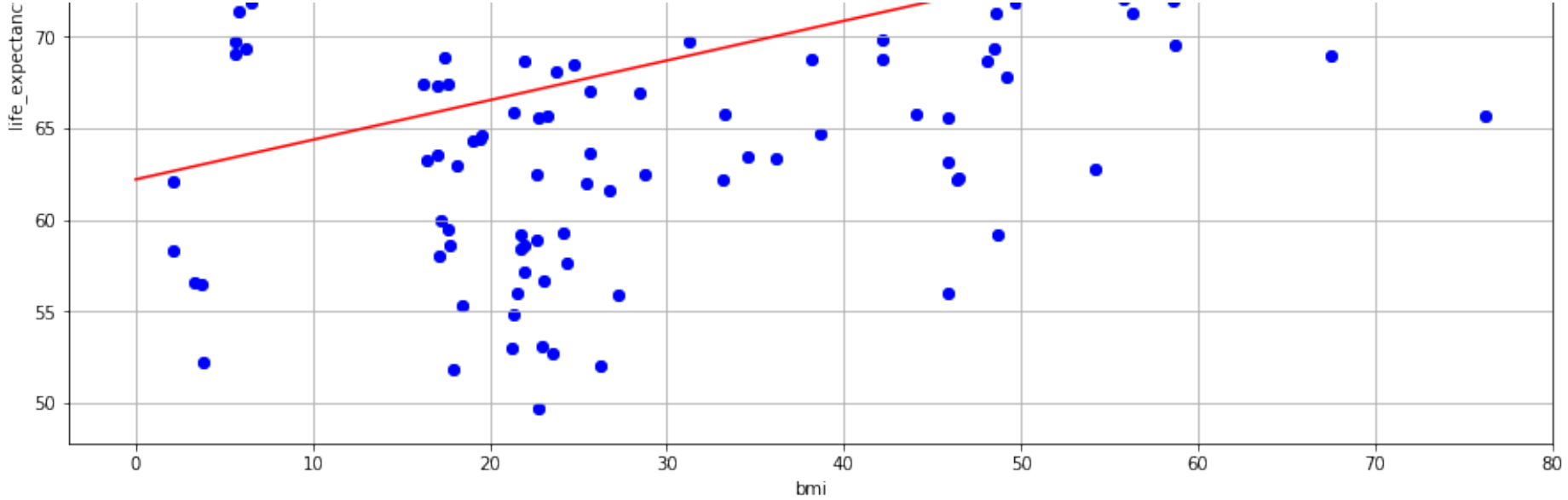


In [59]:

```
#scatterplot - life_expectancy and bmi

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["bmi"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["bmi"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.bmi.max(), 100)

# combining the two plots
plt.scatter(df["bmi"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("bmi")
plt.grid()
```

```
The slope of the regression line is: 0.21666498119007094 The Intercept is: 62.21505858606124
```

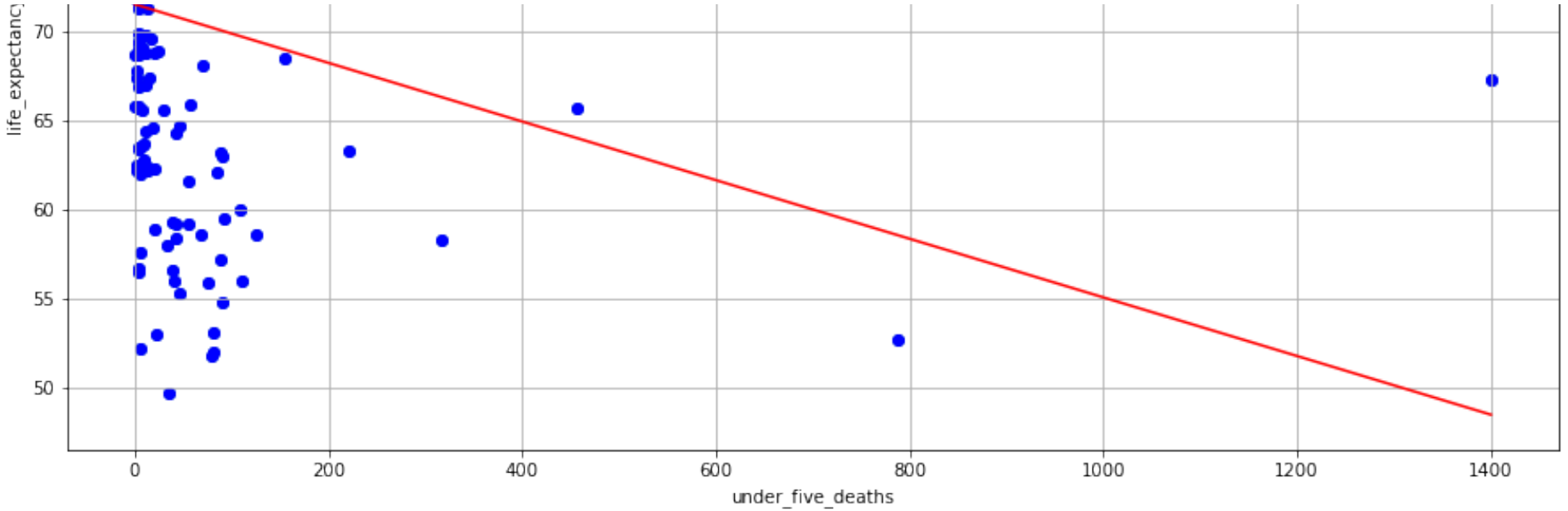




In [60]:

```
#scatterplot - life_expectancy and under_five_deaths

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["under_five_deaths"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["under_five_deaths"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.under_five_deaths.max(), 100)

# combining the two plots
plt.scatter(df["under_five_deaths"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("under_five_deaths")
plt.grid()
```

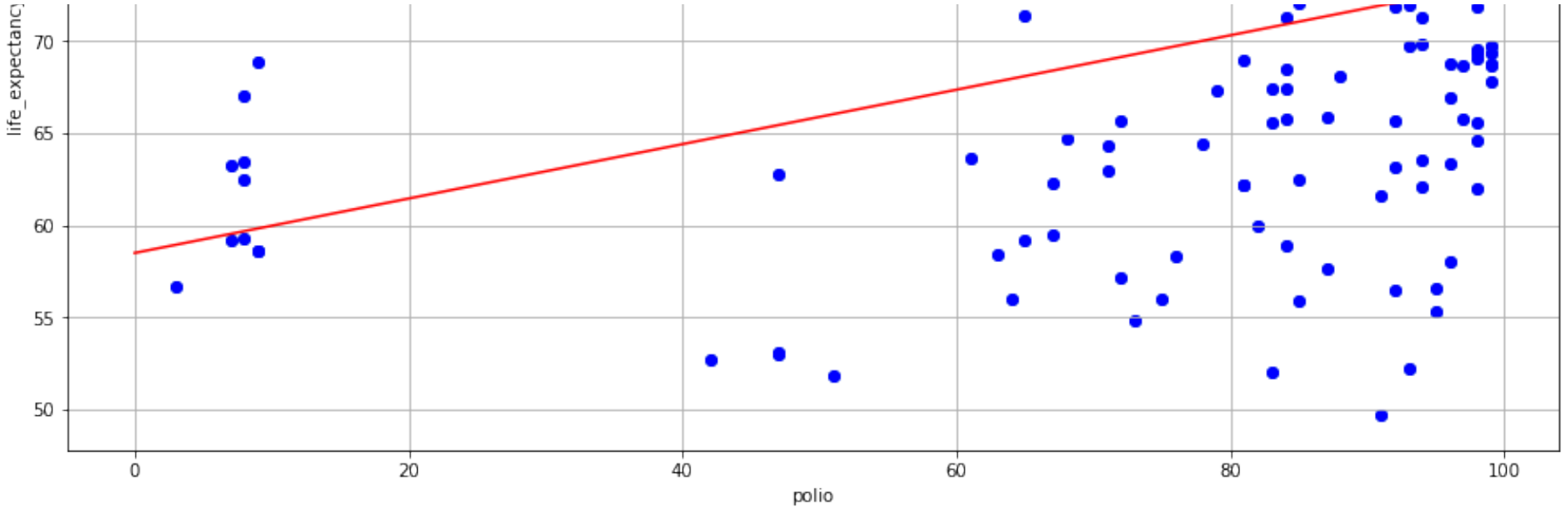
```
The slope of the regression line is: -0.016454313053045205 The Intercept is: 71.51558916014854
```


life_expectanc
70
65
60
55
50
0
200
400
600
800
1000
1200
1400
under_five_deaths

In [61]:

```
#scatterplot - life_expectancy and polio

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["polio"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["polio"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.polio.max(), 100)

# combining the two plots
plt.scatter(df["polio"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("polio")
plt.grid()
```

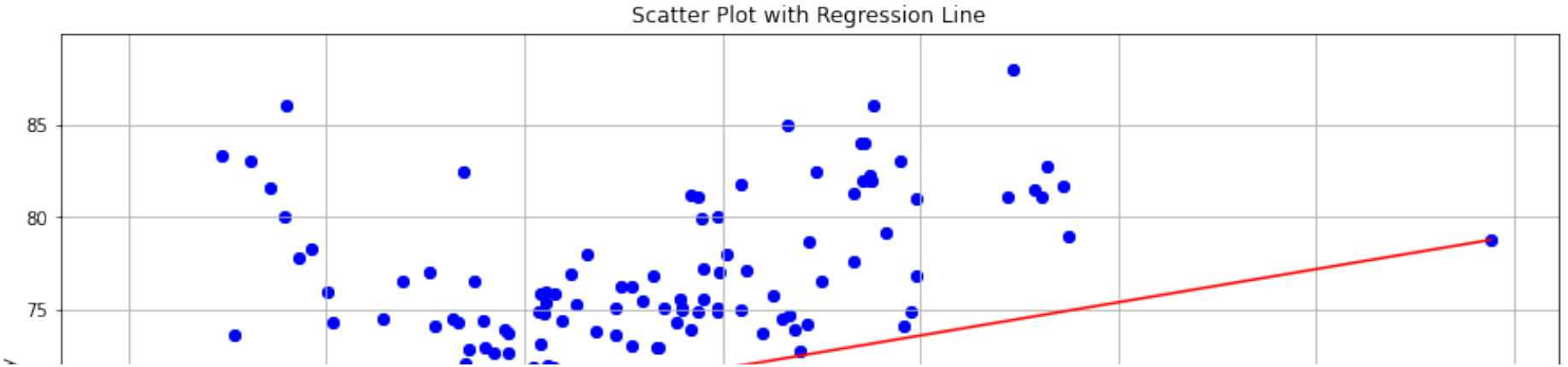
```
The slope of the regression line is: 0.14791035856573734 The Intercept is: 58.50136055776893
```

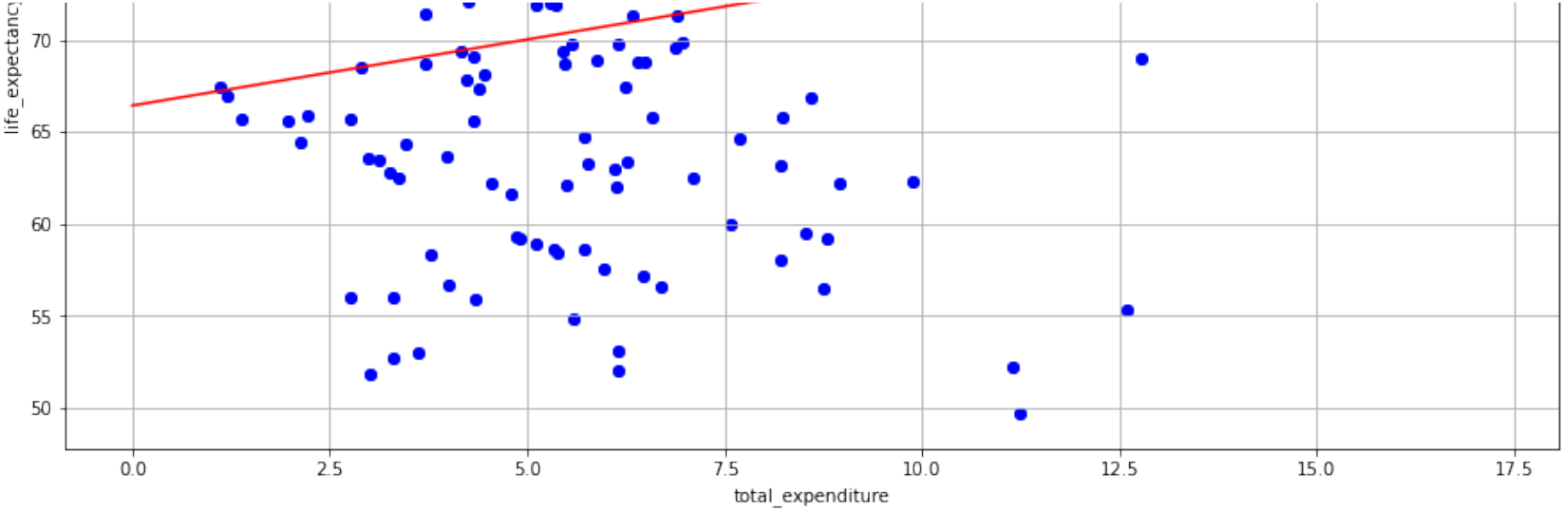


In [62]:

```
#scatterplot - life_expectancy and total_expenditure

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["total_expenditure"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["total_expenditure"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.total_expenditure.max(), 100)

# combining the two plots
plt.scatter(df["total_expenditure"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("total_expenditure")
plt.grid()
```

```
The slope of the regression line is: 0.7174808343041238 The Intercept is: 66.43264546970661
```

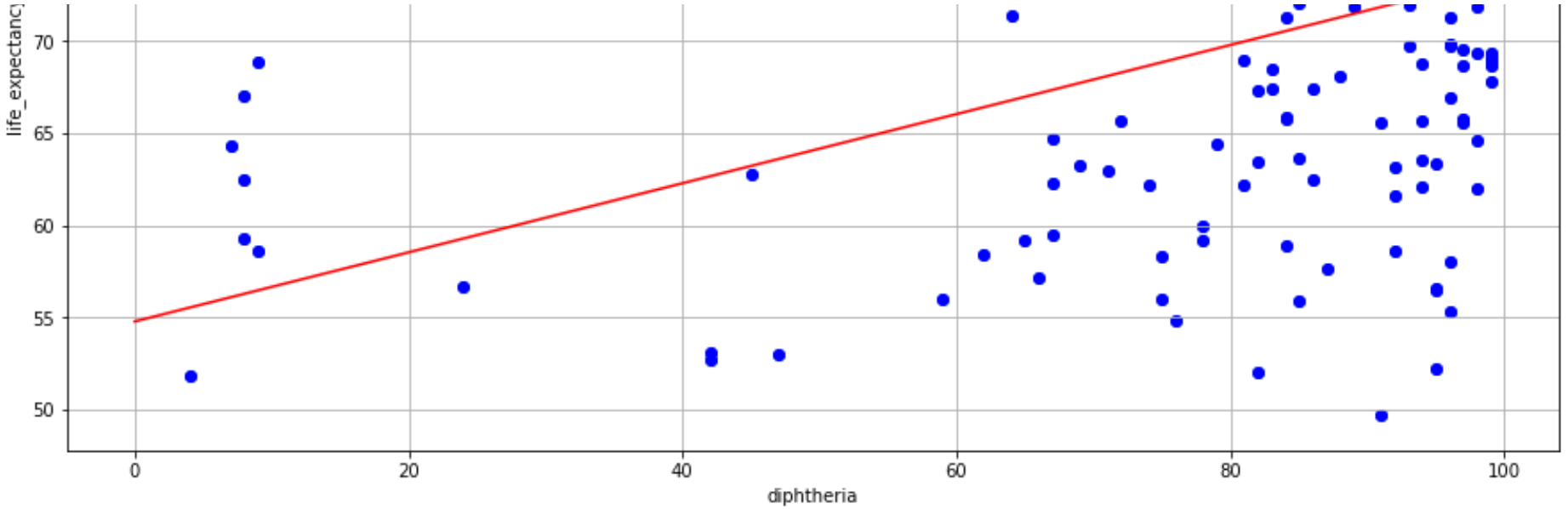



life_expectanc
70
65
60
55
50
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
total_expenditure

In [63]:

```
#scatterplot - life_expectancy and diphtheria

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["diphtheria"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["diphtheria"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.diphtheria.max(), 100)

# combining the two plots
plt.scatter(df["diphtheria"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("diphtheria")
plt.grid()
```

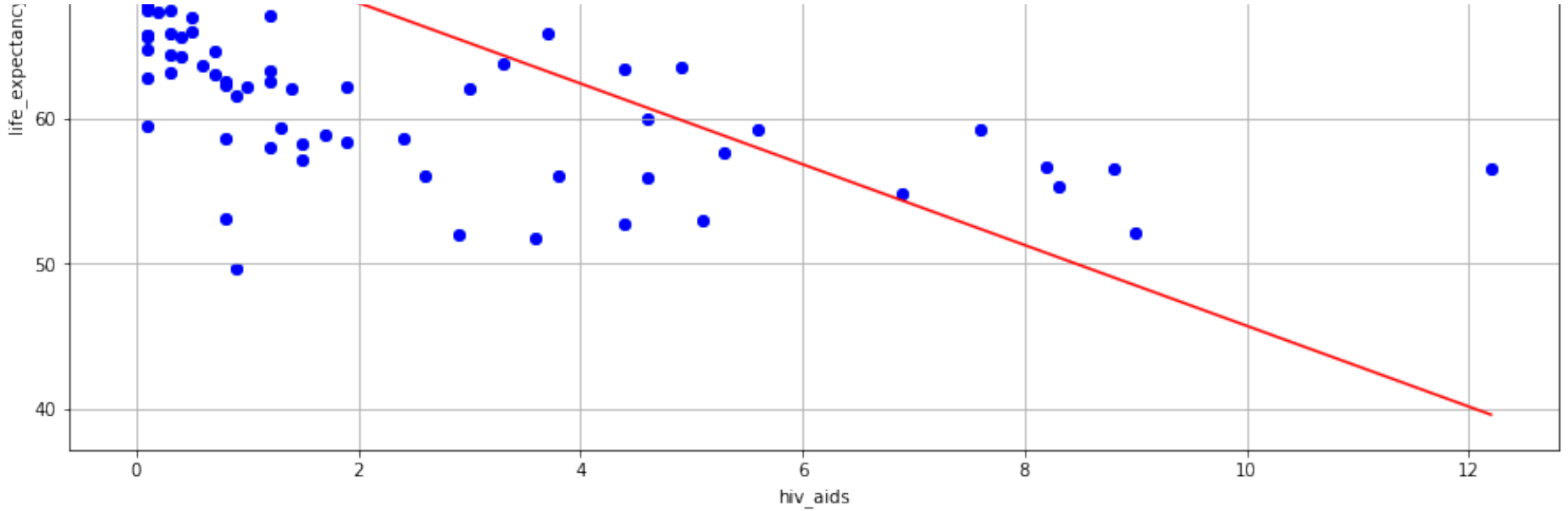
```
The slope of the regression line is: 0.18779556285876453 The Intercept is: 54.783966974302515
```



In [64]:

```
#scatterplot - life_expectancy and hiv_aids

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["hiv_aids"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["hiv_aids"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.hiv_aids.max(), 100)

# combining the two plots
plt.scatter(df["hiv_aids"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("hiv_aids")
plt.grid()
```

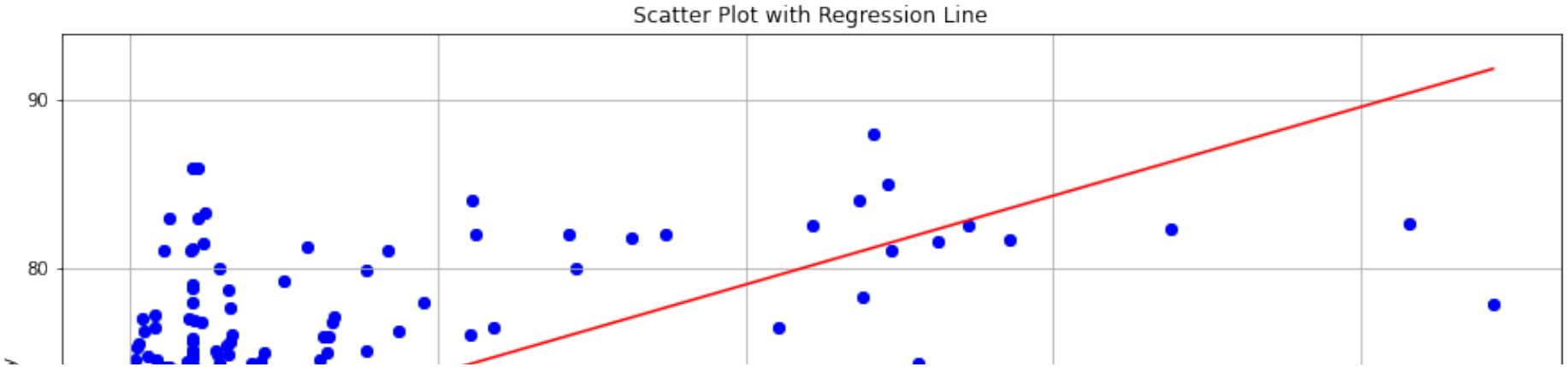
```
The slope of the regression line is: -2.773660319523558 The Intercept is: 73.48750158574424
```

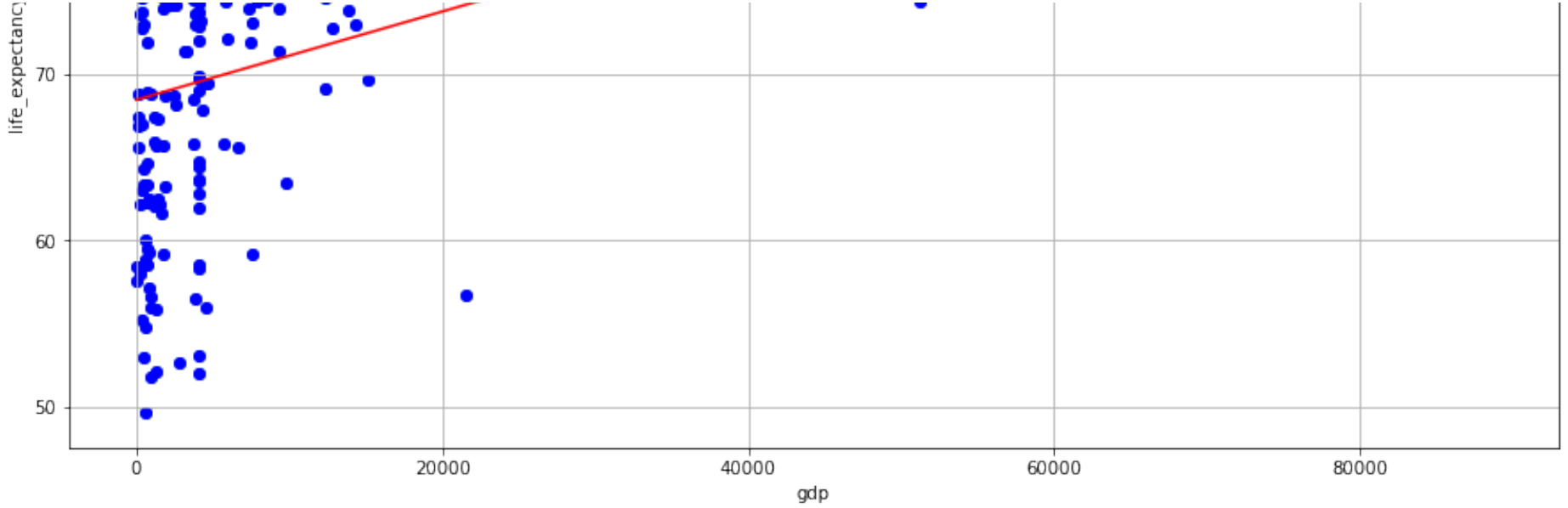


In [65]:

```
#scatterplot - life_expectancy and gdp

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["gdp"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["gdp"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.gdp.max(), 100)

# combining the two plots
plt.scatter(df["gdp"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("gdp")
plt.grid()
```

```
The slope of the regression line is: 0.00026438628491594454 The Intercept is: 68.43734109482033
```

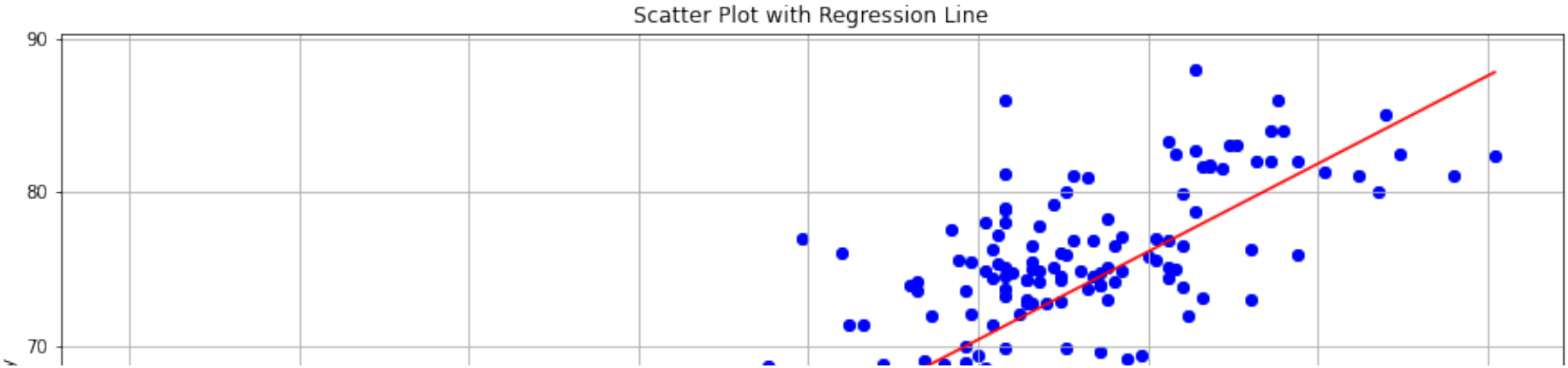


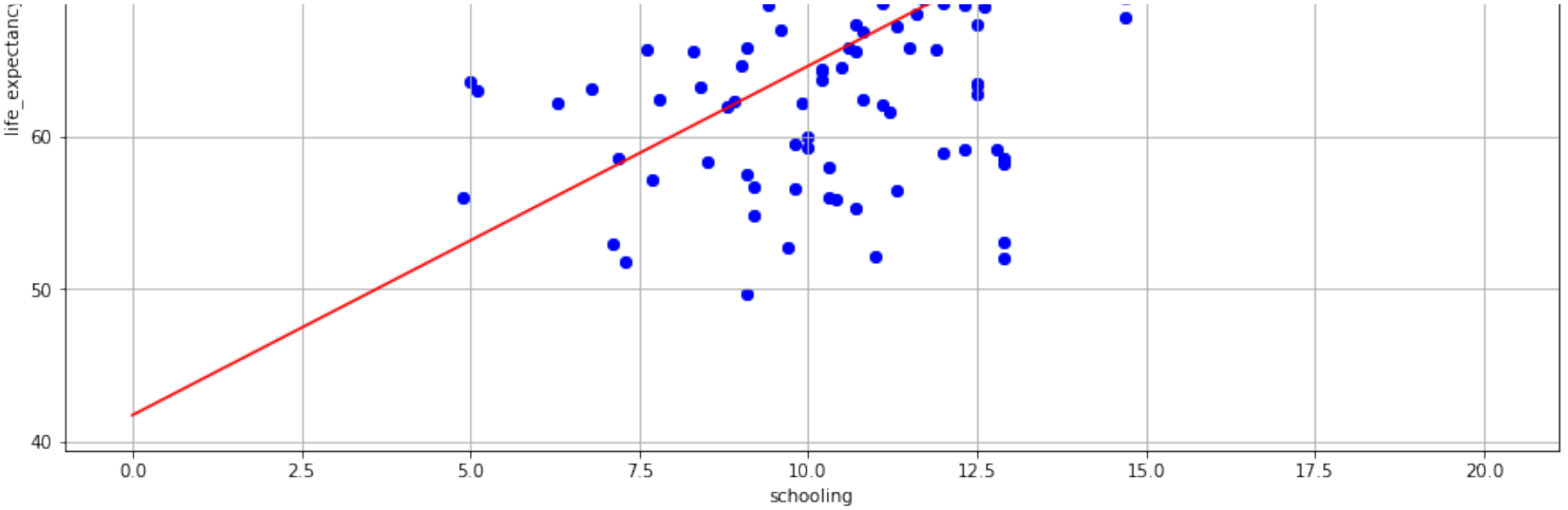


In [66]:

```
#scatterplot - life_expectancy and schooling

plt.figure(figsize = (15, 8))

# Create scatterplot betwene two variables
plt.scatter(df["schooling"], df["life_expectancy"])

# Create regression line
m,b = np.polyfit(df["schooling"], df["life_expectancy"], deg = 1)
print("The slope of the regression line is:", m, "The Intercept is:", b)

# Create a series of equaly spaced values
x_range = np.linspace(0, df.schooling.max(), 100)

# combining the two plots
plt.scatter(df["schooling"], df["life_expectancy"],color = "blue")
plt.plot(x_range, m*x_range+b, color = "red")
plt.title("Scatter Plot with Regression Line")
plt.ylabel("life_expectancy")
plt.xlabel("schooling")
plt.grid()
```

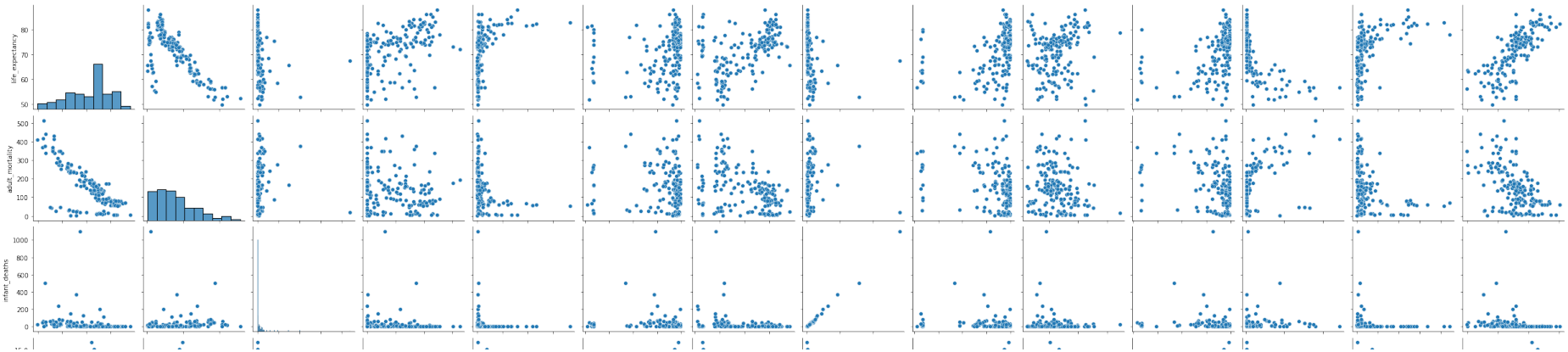
```
The slope of the regression line is: 2.2909497972526003 The Intercept is: 41.74426058722552
```

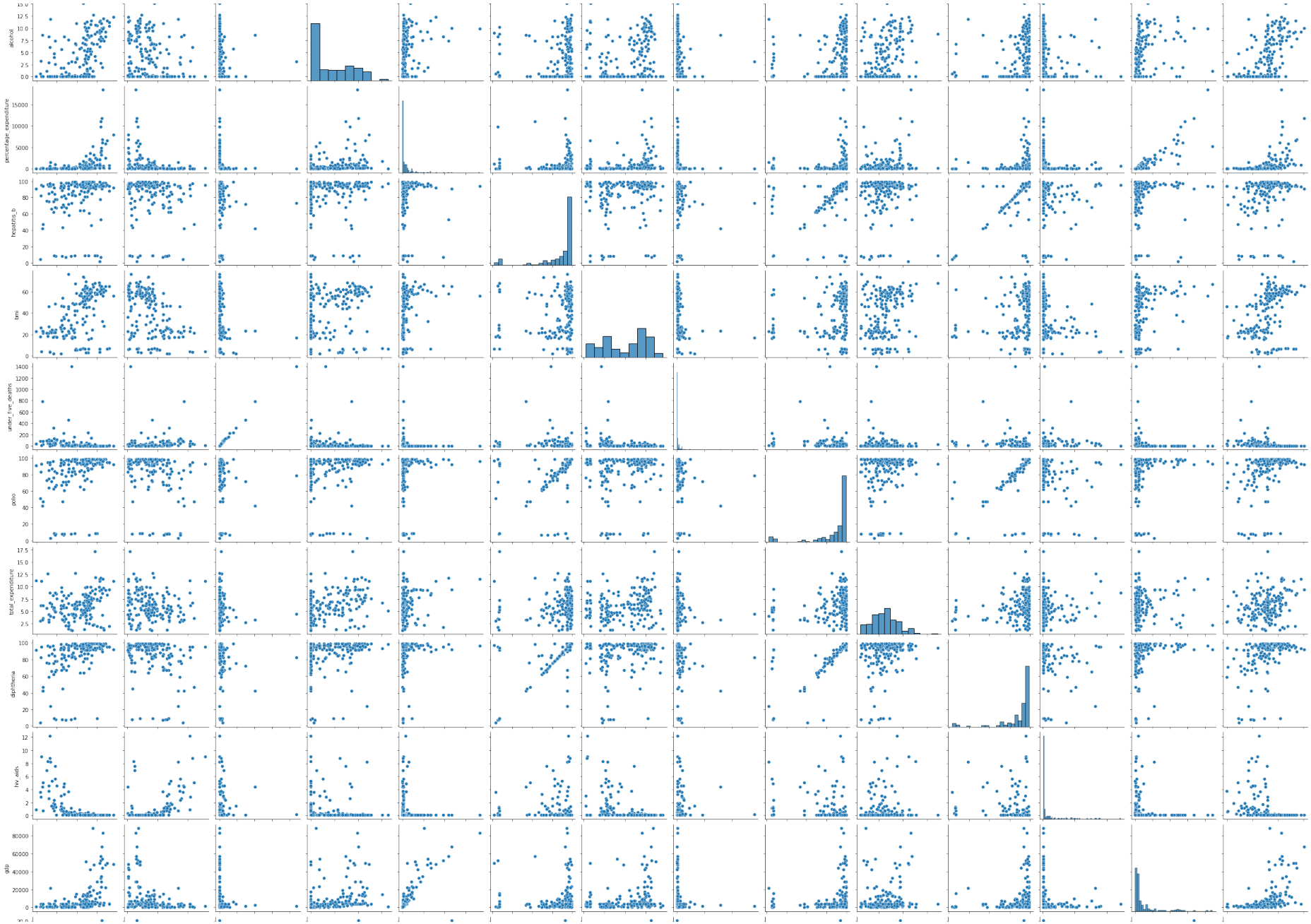


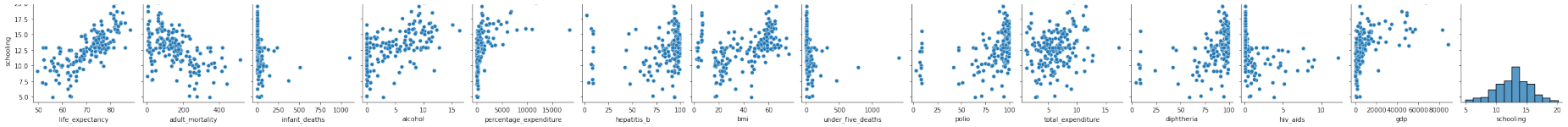


In [67]:
```
# Bivariate Characterizations-
# To understand the relationships between our variables
# a) Scatterplot Matrix
sns.pairplot(df)
```

Out[67]: <seaborn.axisgrid.PairGrid at 0x7fad99109940>

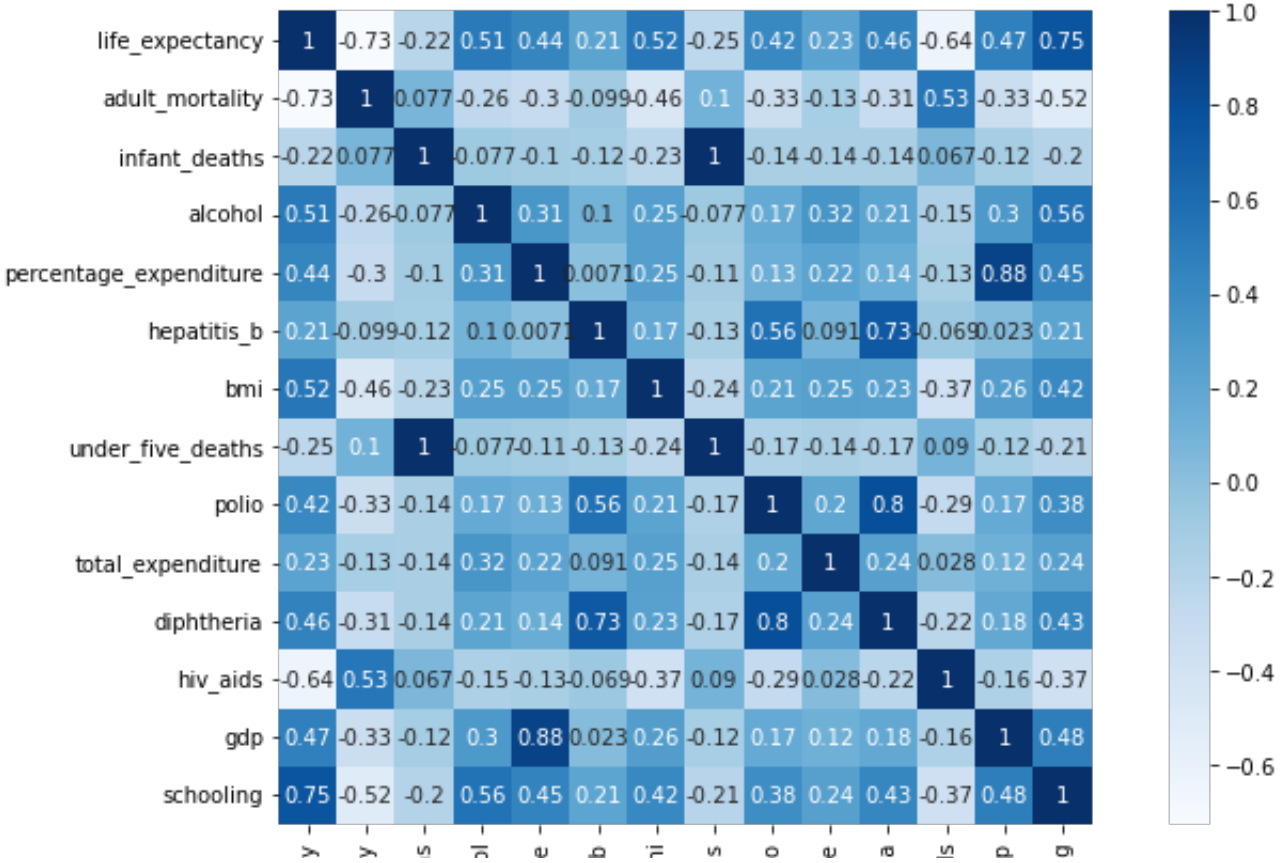


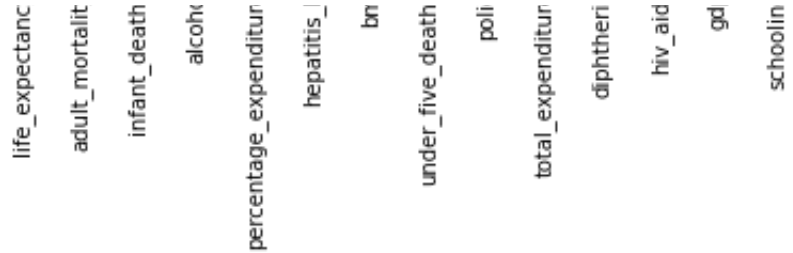

alcohol
percentage_expenditure
hepatitis_b
bmi
under_five_deaths
polio
total_expenditure
diphtheria
hiv_aids
gdp



In [68]:
```
#b) Correlation Matrix
plt.figure(figsize=(13,7))
c= df.corr()
sns.heatmap(c,cmap="Blues",annot=True,square = True)
```

Out[68]: <AxesSubplot:>





Part c:

Test for Linearity and Transformations

Using the Harvey-Collier test, we check the linearity of the model. If the p-value is higher than 5% i.e. 0.05, we reject the null. The null, in this case, being that the model is linear.

In [69]:
```
ols_mod1 = smf.ols(formula = 'life_expectancy ~ adult_mortality', data =df)
```

In [70]:
```
ols_fit1 = ols_mod1.fit()
```

In [71]:
```
pip install simple-colors
```

```
Requirement already satisfied: simple-colors in ./opt/anaconda3/lib/python3.9/site-packages (0.1.5)
Note: you may need to restart the kernel to use updated packages.
```

In [72]:
```
#1 Adult Mortality
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit1)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -1.137275986286063), ('p-value', 0.2569430099397232)]
```

In [73]:
```
ols_mod2 = smf.ols(formula = 'life_expectancy ~ infant_deaths', data =df)
```

In [74]:
```
ols_fit2 = ols_mod2.fit()
```

In [75]:
```
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit2)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.4854025615627821), ('p-value', 0.6279843924542972)]
```

In [76]:
```
ols_mod3 = smf.ols(formula = 'life_expectancy ~ alcohol', data =df)
```

In [77]:
```
ols_fit3 = ols_mod3.fit()
```

In [78]:
```
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit3)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', 0.29810153004243345), ('p-value', 0.765971407435829)]
```

In [79]:
```
ols_mod4 = smf.ols(formula = 'life_expectancy ~ polio', data =df)
```

In [80]:
```
ols_fit4 = ols_mod4.fit()
```

In [81]:
```
#4 Polio
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit4)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.3082171226299988), ('p-value', 0.758275501226657)]
```

In [82]:
```
ols_mod5 = smf.ols(formula = 'life_expectancy ~ percentage_expenditure', data =df)
```

In [83]:
```
ols_fit5 = ols_mod5.fit()
```

In [84]:
```
#5 Percentage expenditure
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit5)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.2942608865431716), ('p-value', 0.7688995125274594)]
```

In [85]:
```
ols_mod6 = smf.ols(formula = 'life_expectancy ~ hepatitis_b', data =df)
```

In [86]:
```
ols_fit6 = ols_mod6.fit()
```

In [87]:
```
#6 Hepatitis B
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit6)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.3793853998655909), ('p-value', 0.7048513784103143)]
```

In [88]:
```
ols_mod7 = smf.ols(formula = 'life_expectancy ~ bmi', data =df)
```

In [89]:
```
ols_fit7 = ols_mod7.fit()
```

In [90]:
```
#7 BMI
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit7)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.319949613172706), ('p-value', 0.7493796704935565)]
```

In [91]:
```
ols_mod8 = smf.ols(formula = 'life_expectancy ~ under_five_deaths', data =df)
```

In [92]:
```
ols_fit8 = ols_mod8.fit()
```

In [93]:
```
#8 Under Five Deaths
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit8)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.4048472900341504), ('p-value', 0.6860731521308883)]
```

In [94]:
```
ols_mod9 = smf.ols(formula = 'life_expectancy ~ total_expenditure', data =df)
```

In [95]:
```
ols_fit9 = ols_mod9.fit()
```

In [96]:
```
#9 Total Expenditure ***
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit9)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.7525881700870919), ('p-value', 0.45268599682890787)]
```

In [97]:
```
ols_mod10 = smf.ols(formula = 'life_expectancy ~ diphtheria', data =df)
```

In [98]:
```
ols_fit10 = ols_mod10.fit()
```

In [99]:
```
#10 Diphtheria
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit10)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.6065164881300859), ('p-value', 0.5449402829771046)]
```

In [100]:
```
ols_mod11 = smf.ols(formula = 'life_expectancy ~ hiv_aids', data =df)
```

In [101]:
```
ols_fit11 = ols_mod11.fit()
```

In [102]:
```
ols_mod12 = smf.ols(formula = 'life_expectancy ~ gdp', data =df)
```

In [103]:
```
ols_fit12 = ols_mod12.fit()
```

In [104]:
```
#11 GDP
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit12)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

```
Linearity Test Results:
[('t-stat', -0.3959221742751515), ('p-value', 0.6926339539847022)]
```

In [105]:
```
ols_mod13 = smf.ols(formula = 'life_expectancy ~ schooling', data =df)
```

In [106]:
```
ols_fit13 = ols_mod13.fit()
```

In [107]:
```
#12 Schooling ***
import statsmodels.stats.api as sms
from simple_colors import *
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit13)
print(blue("Linearity Test Results:", ['bold']))
print(list(zip(name, test)))
print("\n")
```

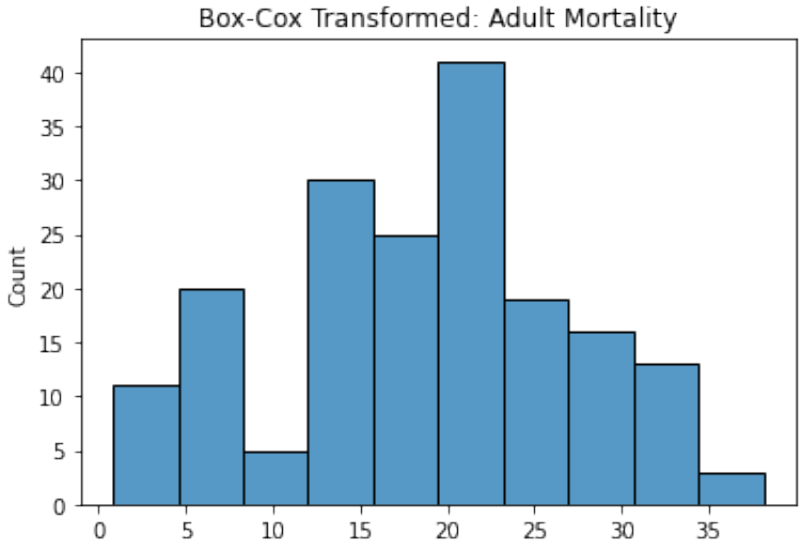
```
Linearity Test Results:
[('t-stat', 0.4494571989294599), ('p-value', 0.6536456239350792)]
```

To transform the variables, we use the Box-Cox Transformation which gives us the optimal value of lambda. Transforming the variables using Box-Cox transformation gives us a more normally distributed model.

In [108]:
```
import scipy
from scipy.special import boxcox1p
```

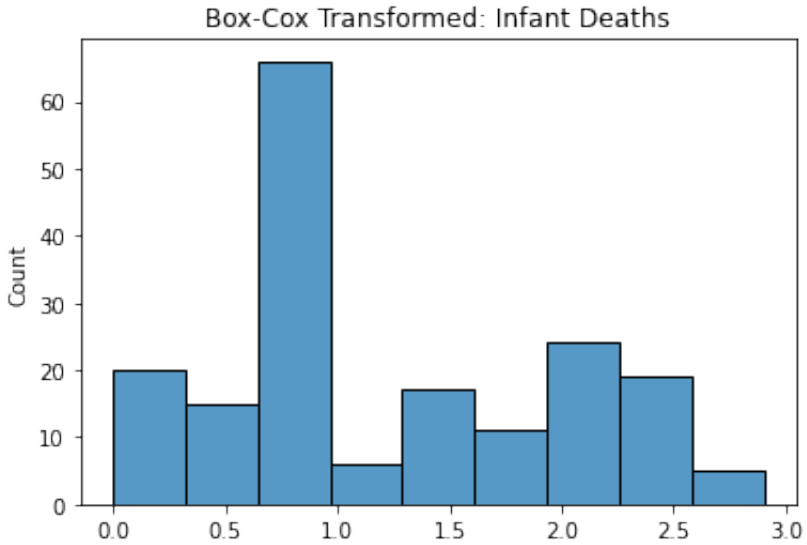
In [109]:
```
bc_adult_mortality,lambda_adult_mortality = scipy.stats.boxcox(df["adult_mortality"])
print(lambda_adult_mortality)

sns.histplot(bc_adult_mortality)
plt.title("Box-Cox Transformed: Adult Mortality")
plt.show()
```

```
0.47170957132662655
```



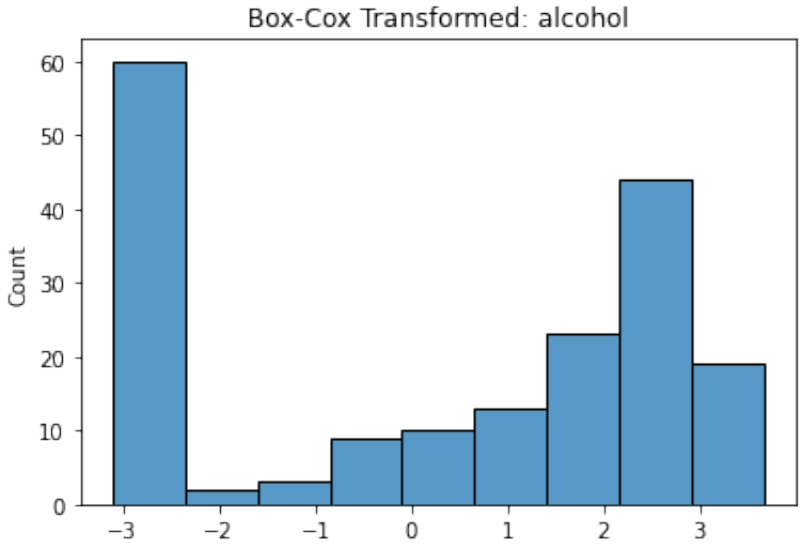
In [110]:
```
bc_infant_deaths,lambda_infant_deaths = scipy.stats.boxcox(df["infant_deaths"])
print(lambda_infant_deaths)

sns.histplot(bc_infant_deaths)
plt.title("Box-Cox Transformed: Infant Deaths")
plt.show()
```

```
-0.30360114856236115
```



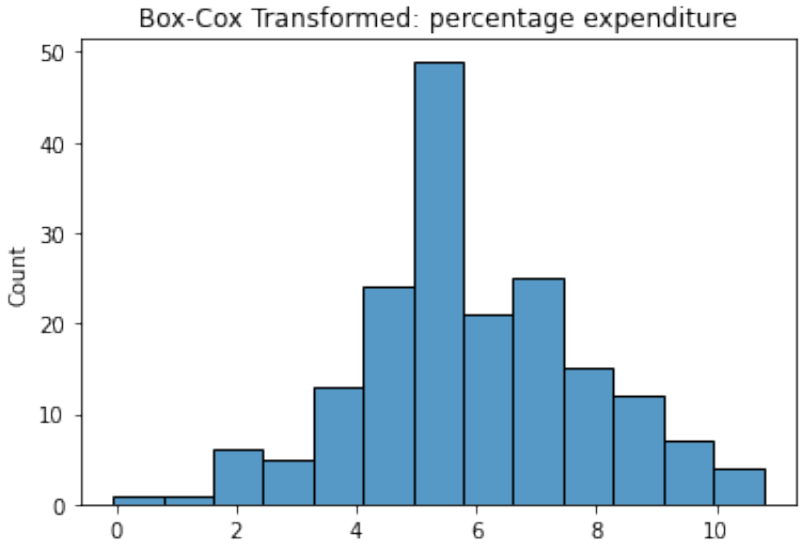
In [111]:
```
bc_alcohol,lambda_alcohol = scipy.stats.boxcox(df["alcohol"])
print(lambda_alcohol)

sns.histplot(bc_alcohol)
plt.title("Box-Cox Transformed: alcohol")
plt.show()
```

```
0.1847265016731399
```



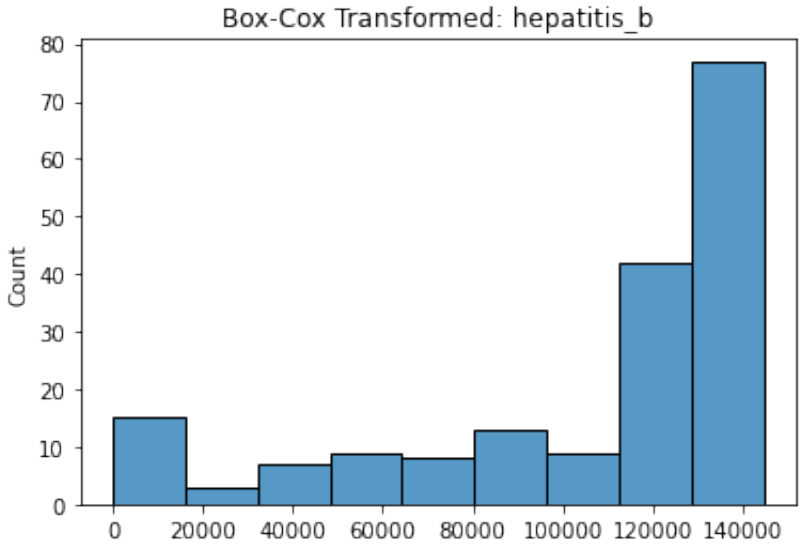
In [112]:
```
bc_percentage_expenditure, lambda_percentage_expenditure = scipy.stats.boxcox(df["percentage_expenditure
print(lambda_percentage_expenditure)

sns.histplot(bc_percentage_expenditure)
plt.title("Box-Cox Transformed: percentage expenditure")
plt.show()
```

```
0.01873402295579603
```



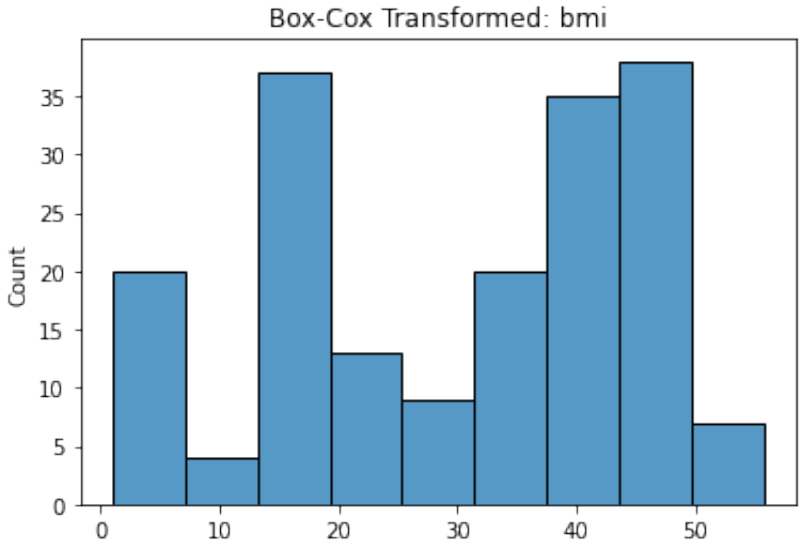
In [113]:
```
bc_hepatitis_b,lambda_hepatitis_b = scipy.stats.boxcox(df["hepatitis_b"])
print(lambda_hepatitis_b)

sns.histplot(bc_hepatitis_b)
plt.title("Box-Cox Transformed: hepatitis_b")
plt.show()
```

```
2.810556504064491
```



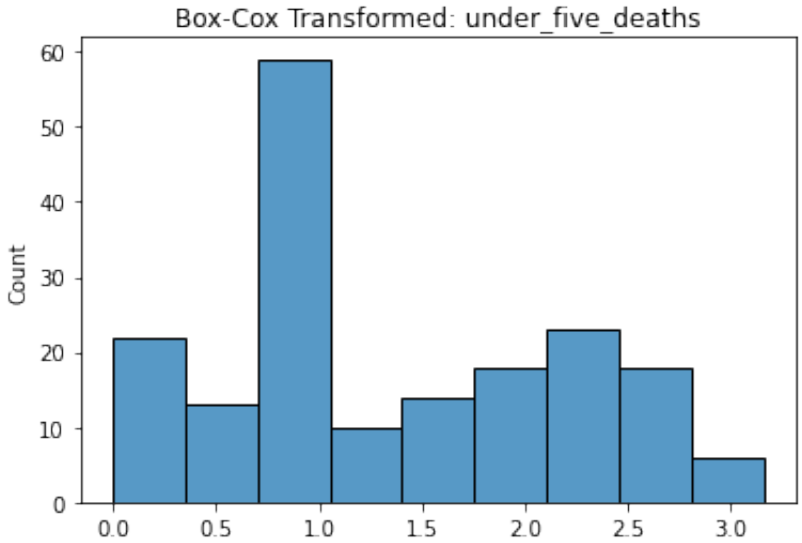
In [114]:
```
bc_bmi,lambda_bmi = scipy.stats.boxcox(df["bmi"])
print(lambda_bmi)

sns.histplot(bc_bmi)
plt.title("Box-Cox Transformed: bmi")
plt.show()
```

```
0.9103909529675764
```



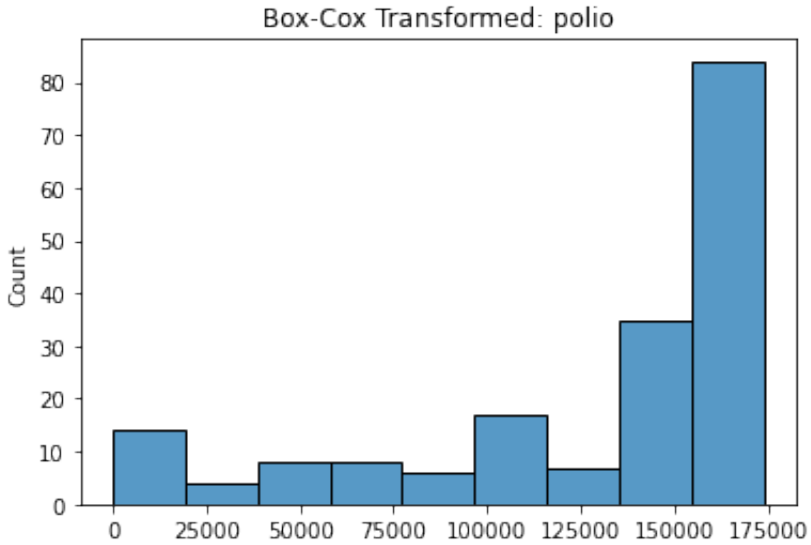
In [115]:
```
bc_under_five_deaths,lambda_under_five_deaths = scipy.stats.boxcox(df["under_five_deaths"])
print(lambda_under_five_deaths)

sns.histplot(bc_under_five_deaths)
plt.title("Box-Cox Transformed: under_five_deaths")
plt.show()
```

```
-0.2725934899464049
```



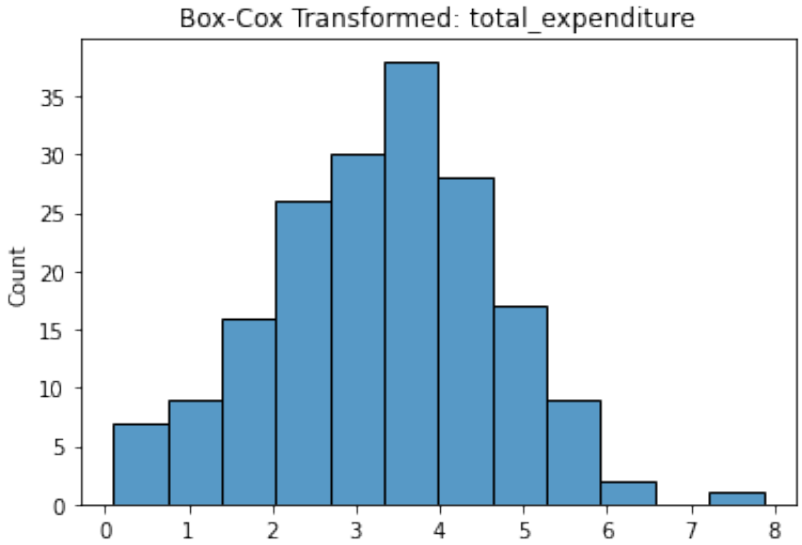
In [116]:
```
bc_polio,lambda_polio = scipy.stats.boxcox(df["polio"])
print(lambda_polio)

sns.histplot(bc_polio)
plt.title("Box-Cox Transformed: polio")
plt.show()
```

```
2.8540604341208673
```



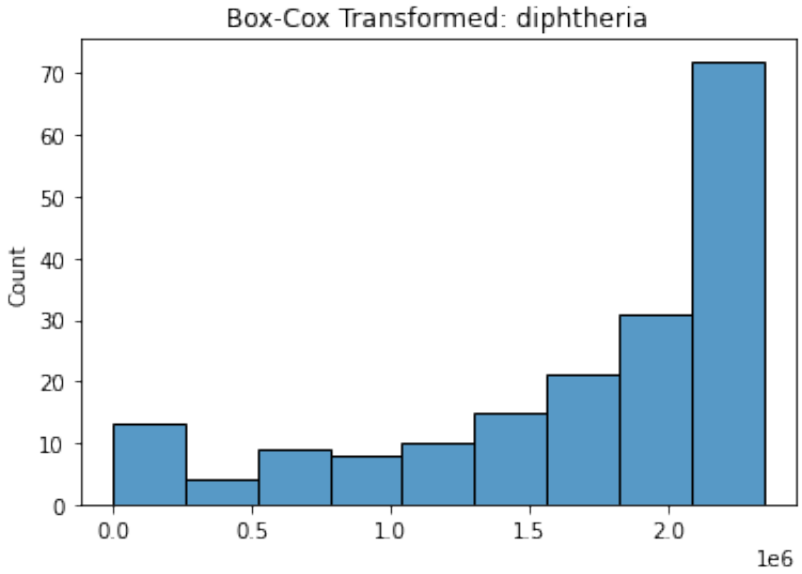
In [117]:
```
bc_total_expenditure,lambda_total_expenditure = scipy.stats.boxcox(df["total_expenditure"])
print(lambda_total_expenditure)

sns.histplot(bc_total_expenditure)
plt.title("Box-Cox Transformed: total_expenditure")
plt.show()
```

```
0.6246951004171188
```



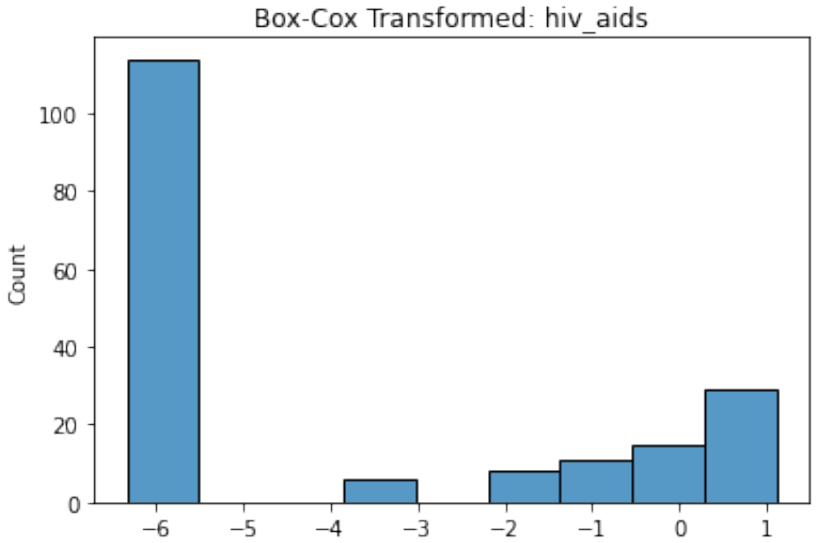
In [118]:
```
bc_diphtheria,lambda_diphtheria = scipy.stats.boxcox(df["diphtheria"])
print(lambda_diphtheria)

sns.histplot(bc_diphtheria)
plt.title("Box-Cox Transformed: diphtheria")
plt.show()
```

```
3.462173037741079
```



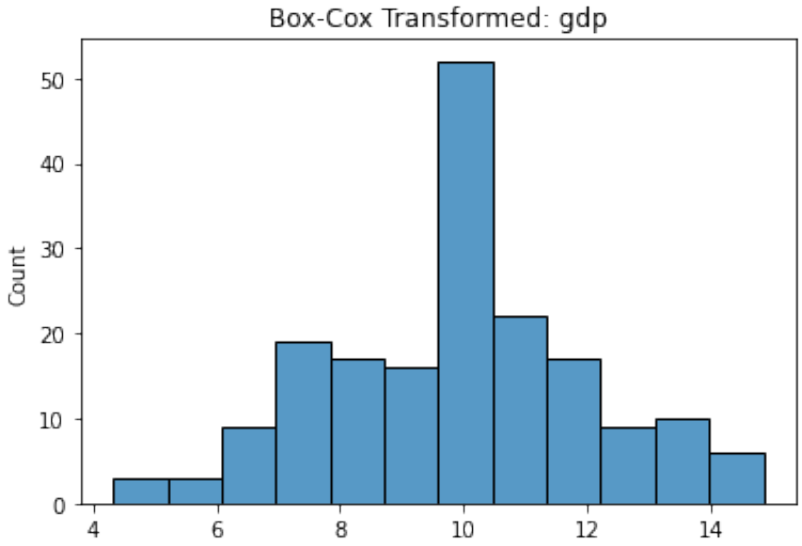
In [119]:
```
bc_hiv_aids,lambda_hiv_aids = scipy.stats.boxcox(df["hiv_aids"])
print(lambda_hiv_aids)

sns.histplot(bc_hiv_aids)
plt.title("Box-Cox Transformed: hiv_aids")
plt.show()
```

```
-0.7676422714148633
```



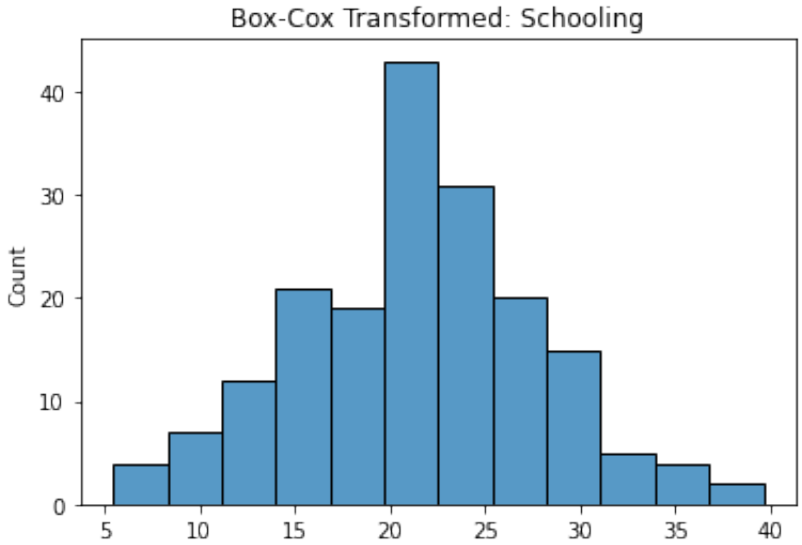
In [120]:
```
import statsmodels.stats.api as sms
bc_gdp,lambda_gdp = scipy.stats.boxcox(df["gdp"])
print(lambda_gdp)

sns.histplot(bc_gdp)
plt.title("Box-Cox Transformed: gdp")
plt.show()
```

```
0.04461711084201528
```



In [121]:
```
bc_schooling,lambda_schooling = scipy.stats.boxcox(df["schooling"])
print(lambda_schooling)

sns.histplot(bc_schooling)
plt.title("Box-Cox Transformed: Schooling")
plt.show()
```

```
1.3267549713354299
```



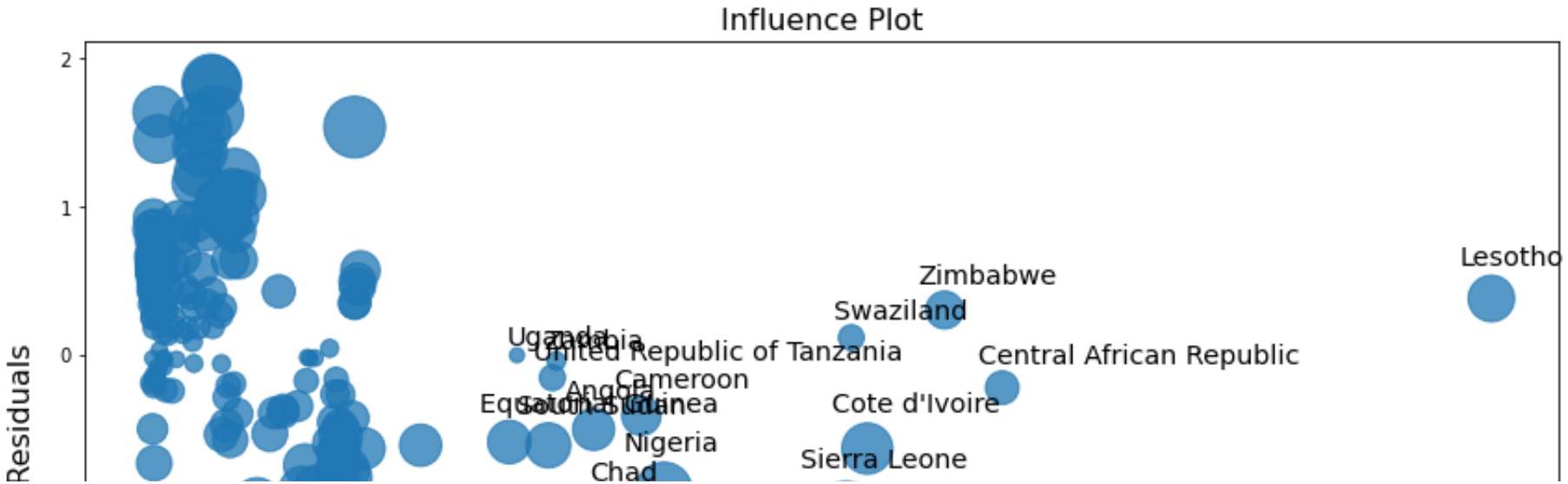
If we include non-linear variables in the regression model, we break the linearity assumption of the Ordinary Least Squares regression model. Due to this, our parameter estimates would be biased, the model would have poor performance and the results will be inaccurate.

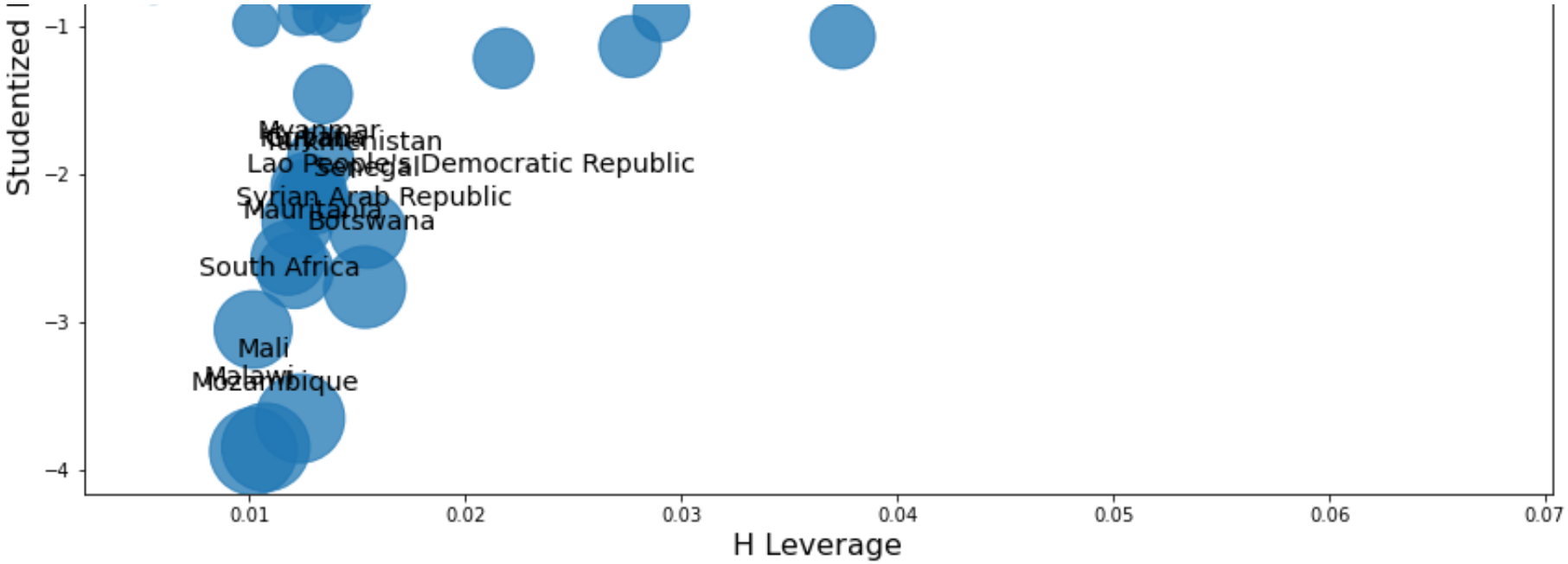
Part d

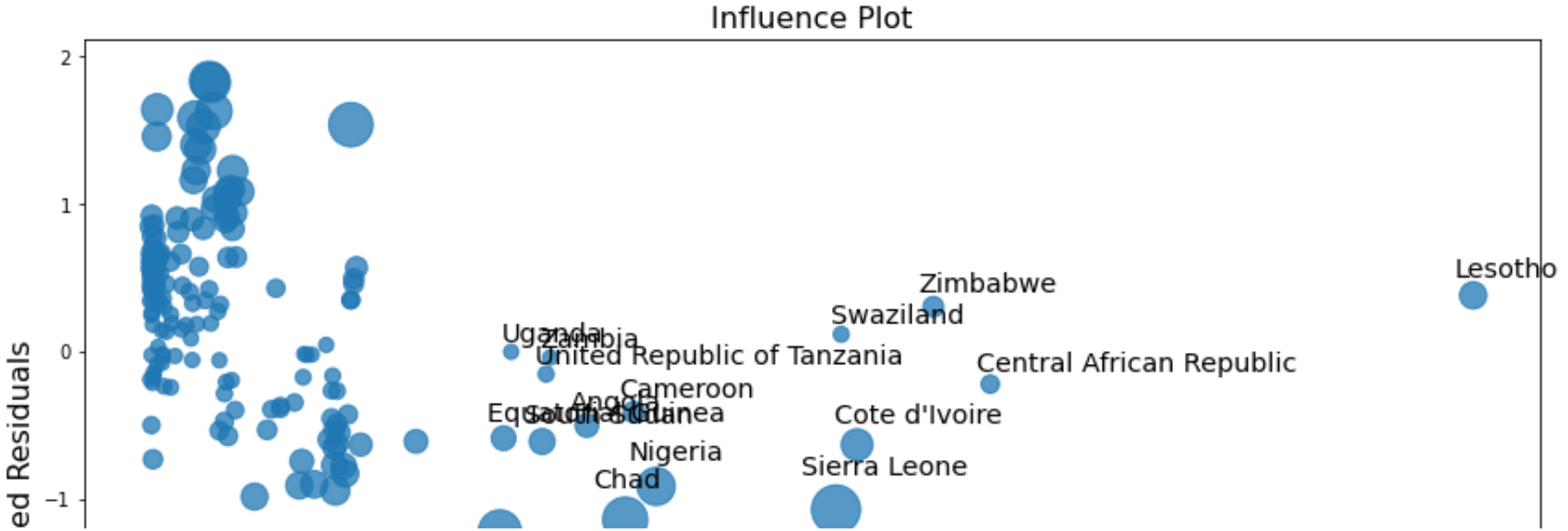
Outliers and Unusual Observations

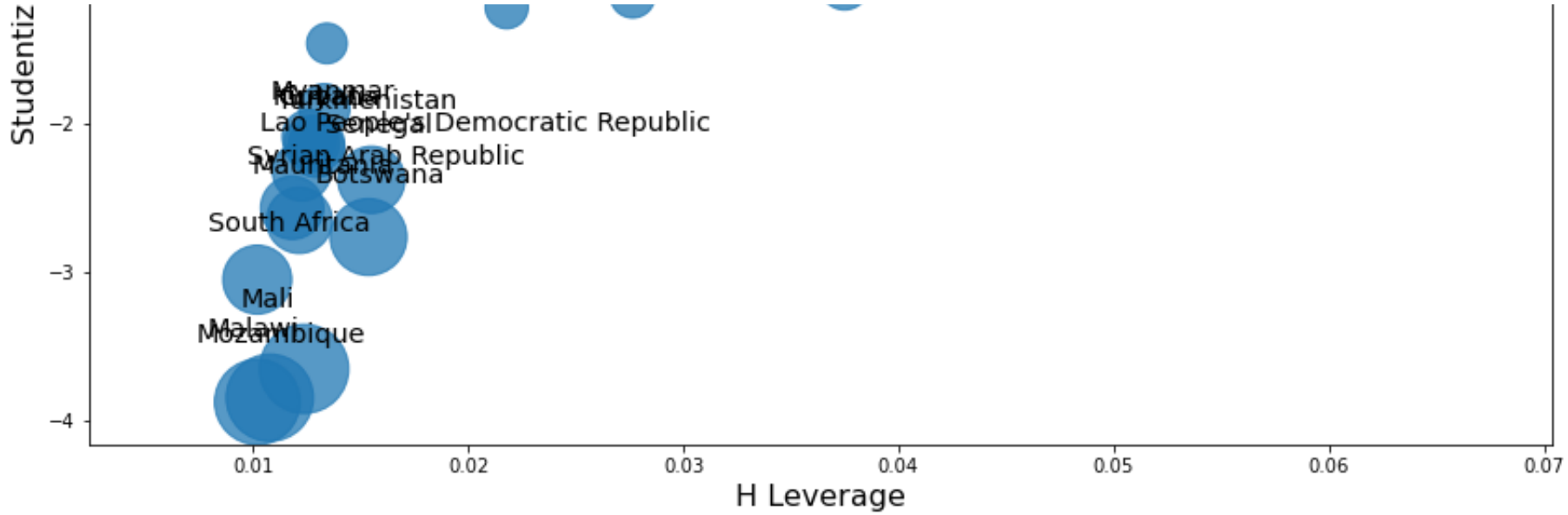
We use the combined Cook's plot to check for influence, outliers and observations with high influence. Observations with large residuals (approximately outside the +/-2 range) have larger residuals and are therefore outliers. Observations towards the extreme right of the graph have high leverage. Observations that have both high leverage and large residuals are influential observations.

In [124]:
```
#For adult mortality
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit1, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit1, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```



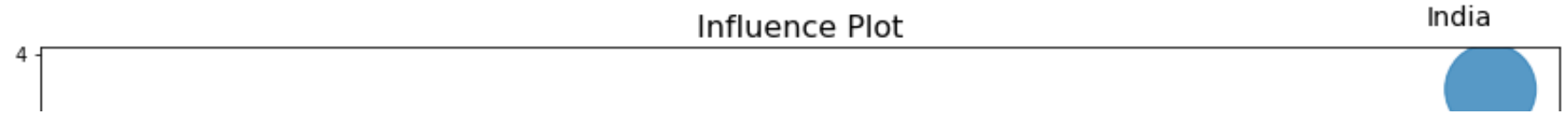

Studentized
−1
−2
−3
−4
Myanmar
Democratic Republic
Lao
Senegal
Syrian Arab Republic
Mauritania
Botswana
South Africa
Mali
Malawi
Mozambique
0.01
0.02
0.03
0.04
0.05
0.06
0.07
H Leverage

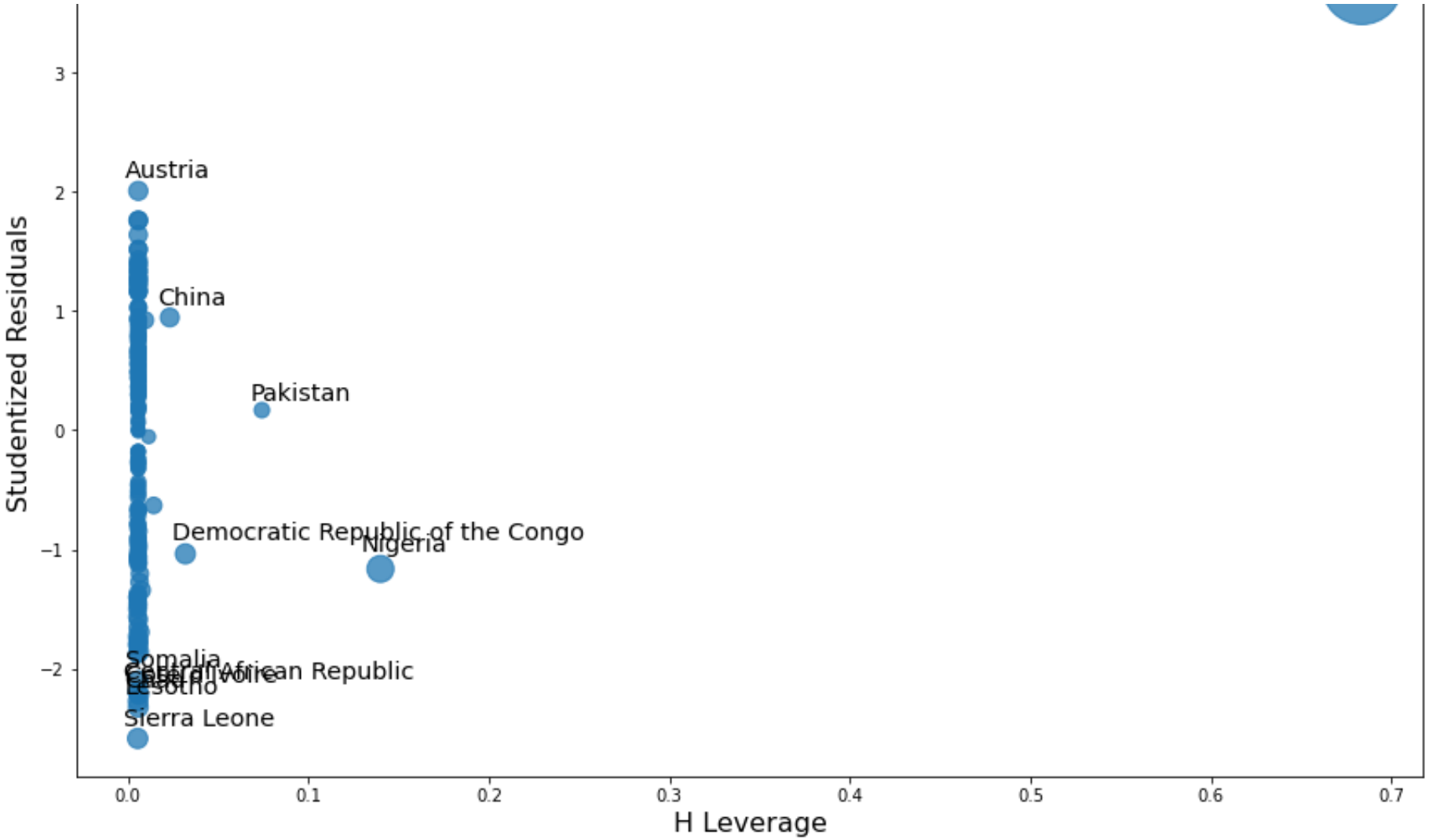

Influence Plot
ed Residuals
2
1
0
−1
Zimbabwe
Lesotho
Swaziland
Uganda
Zambia
United Republic of Tanzania
Central African Republic
Cameroon
Cote d'Ivoire
Nigeria
Sierra Leone
Chad



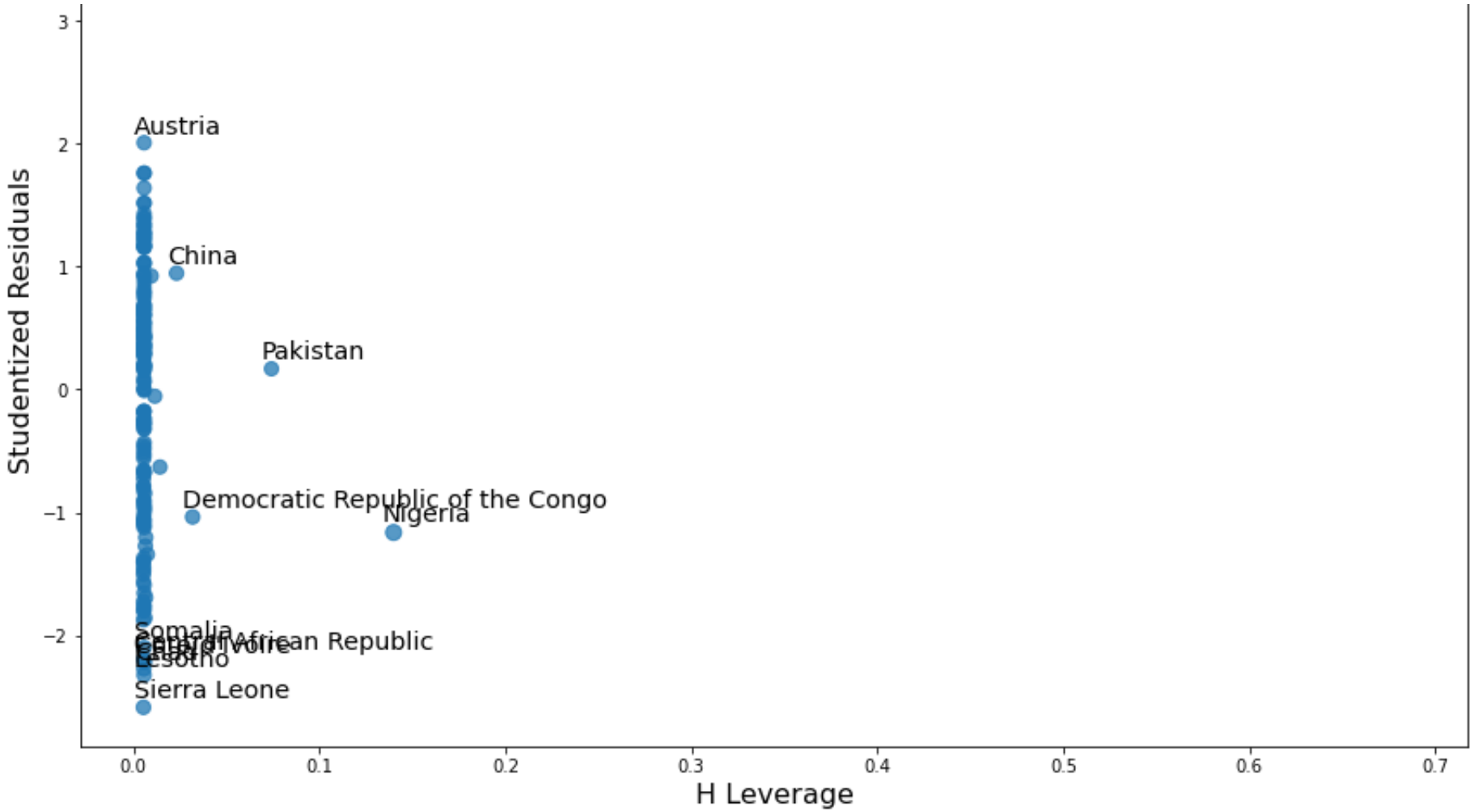
In [ ]:
```
# For adult mortality,
# Countries towards the right of the graph like Lisotho, Central African Republic have high leverage
#but smaller residuals
# Countries with residuals outside the (+/-2) range like Mozambique and Mali are outliers
```

In [125]:
```
# For infant deaths
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit2 , ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit2 , ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```




Studentized Residuals
H Leverage
Austria
China
Pakistan
Democratic Republic of the Congo
Nigeria
Somalia
Sierra Leone
3
2
1
0
−1
−2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

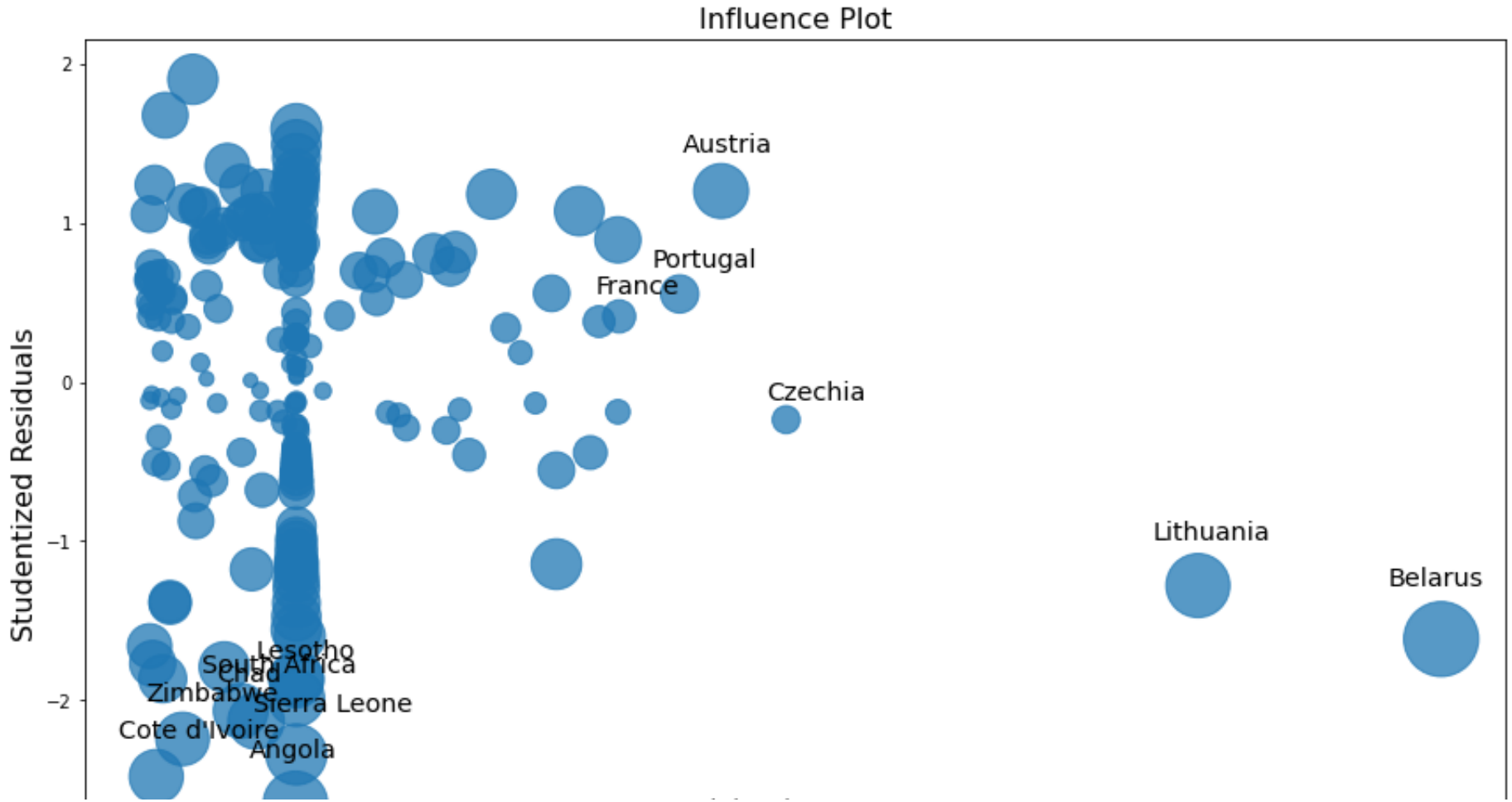

Influence Plot
India
4

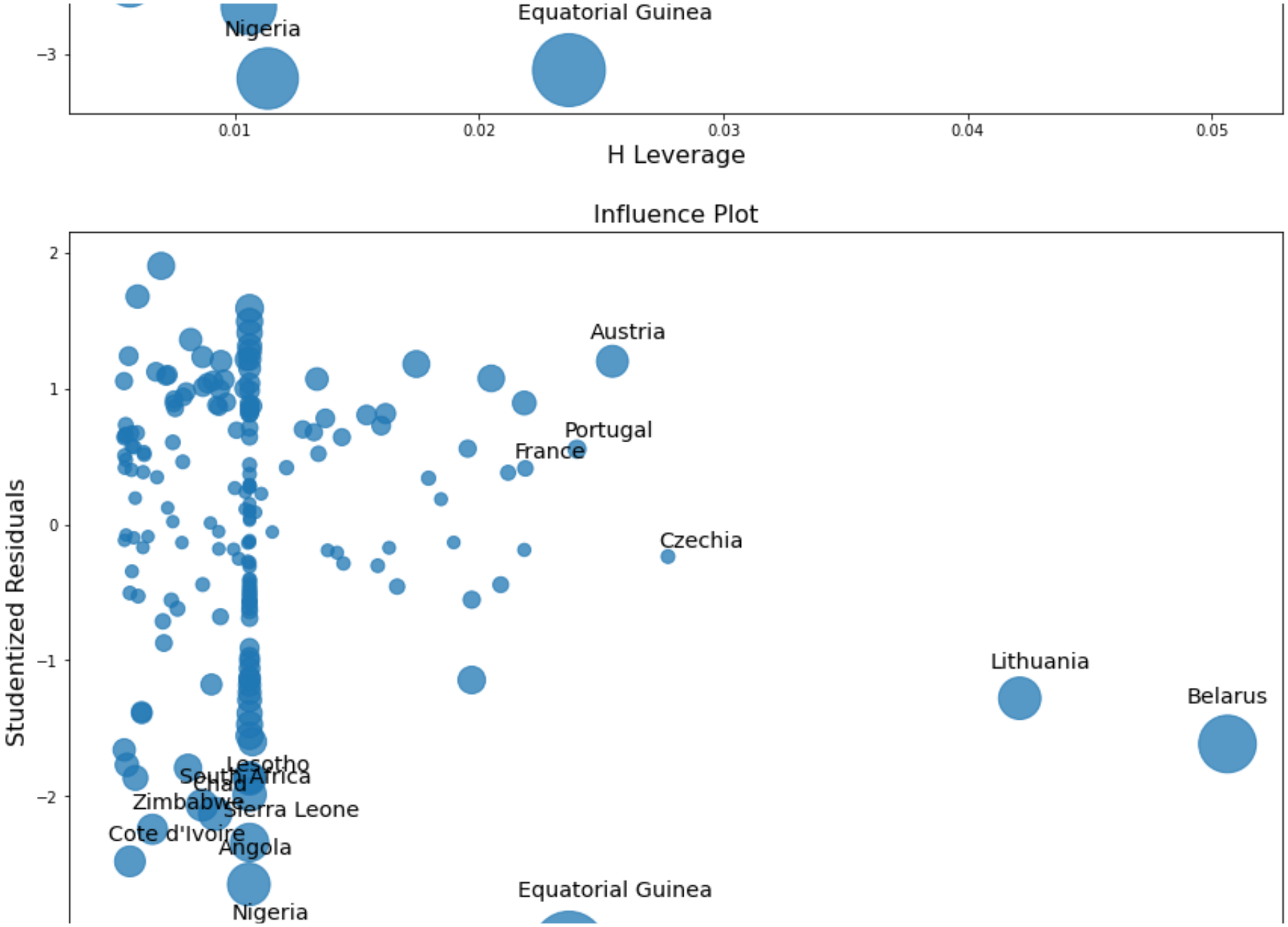


In [ ]:
```
# For infant deaths,
# India is an influential observation with high leverage as well as large residual
# Countries with residuals outside the (+/-2) range like Sierra Leone and Austria are outliers
```

In [126]:
```

```

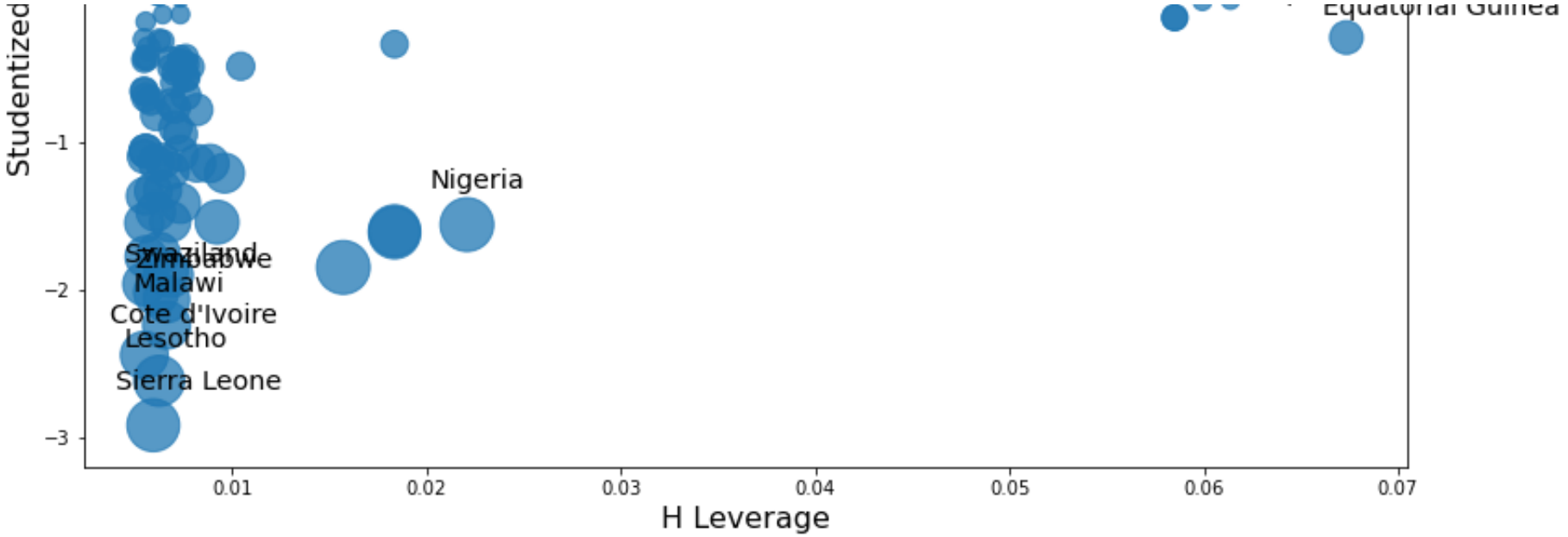
```
# For alcohol
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit3, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit3, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

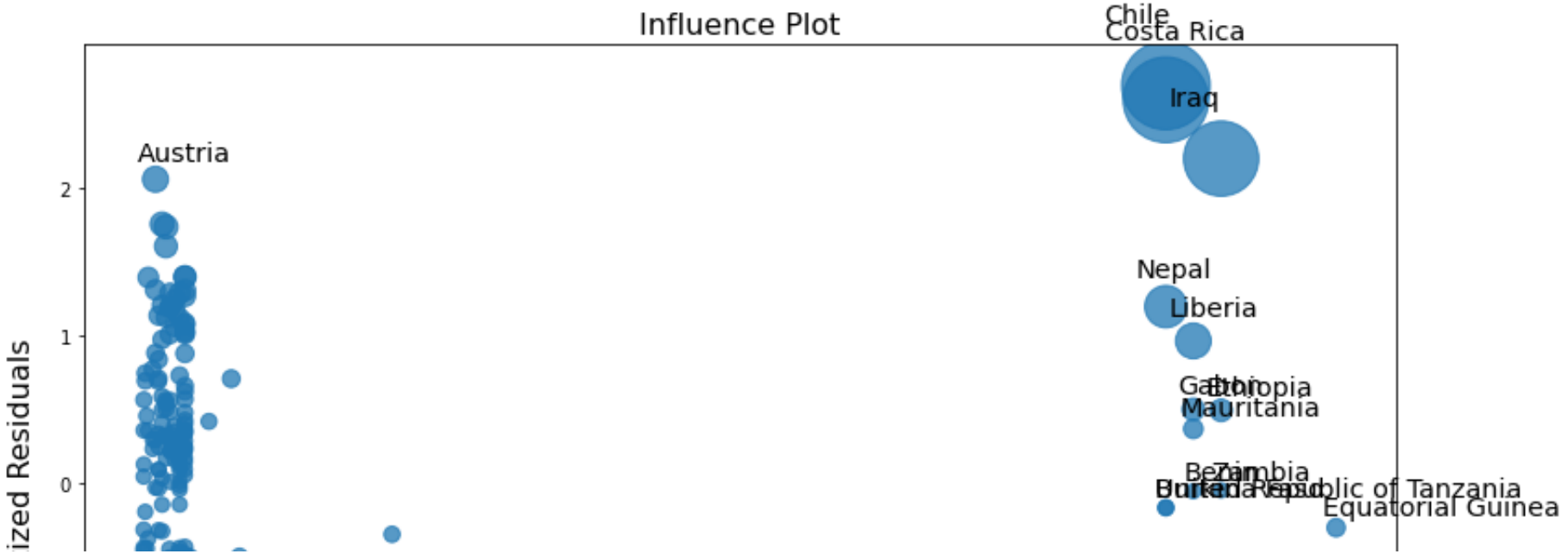


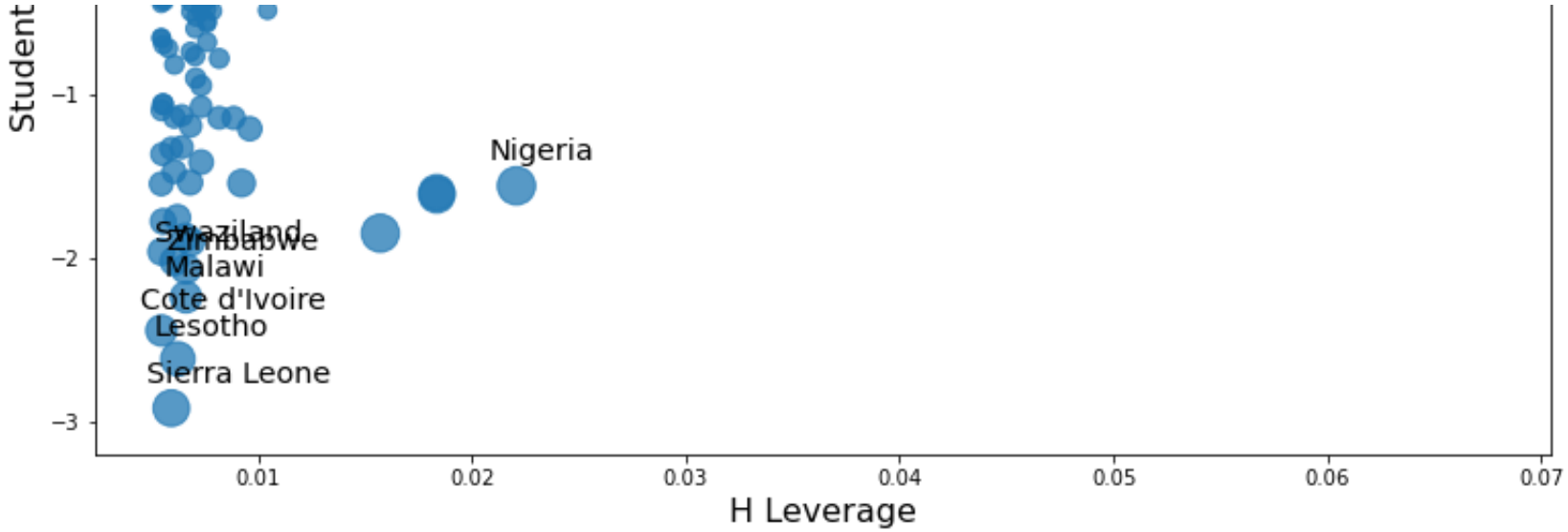

Nigeria
Equatorial Guinea
−3
0.01
0.02
0.03
0.04
0.05
H Leverage
Influence Plot
Studentized Residuals
2
1
0
−1
−2
Austria
Portugal
France
Czechia
Lithuania
Belarus
Lesotho
South Africa
Chad
Zimbabwe
Sierra Leone
Cote d'Ivoire
Angola
Nigeria
Equatorial Guinea

In [ ]:
```
# For alcohol,
# Belarus and Lithuania are influential observations with high leverage and large residuals
# Countries with residuals outside the (+/−2) range like Nigeria and Equatorial Guinea are outliers
```

In [127]:
```
# For percentage expenditure
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit4, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit4, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

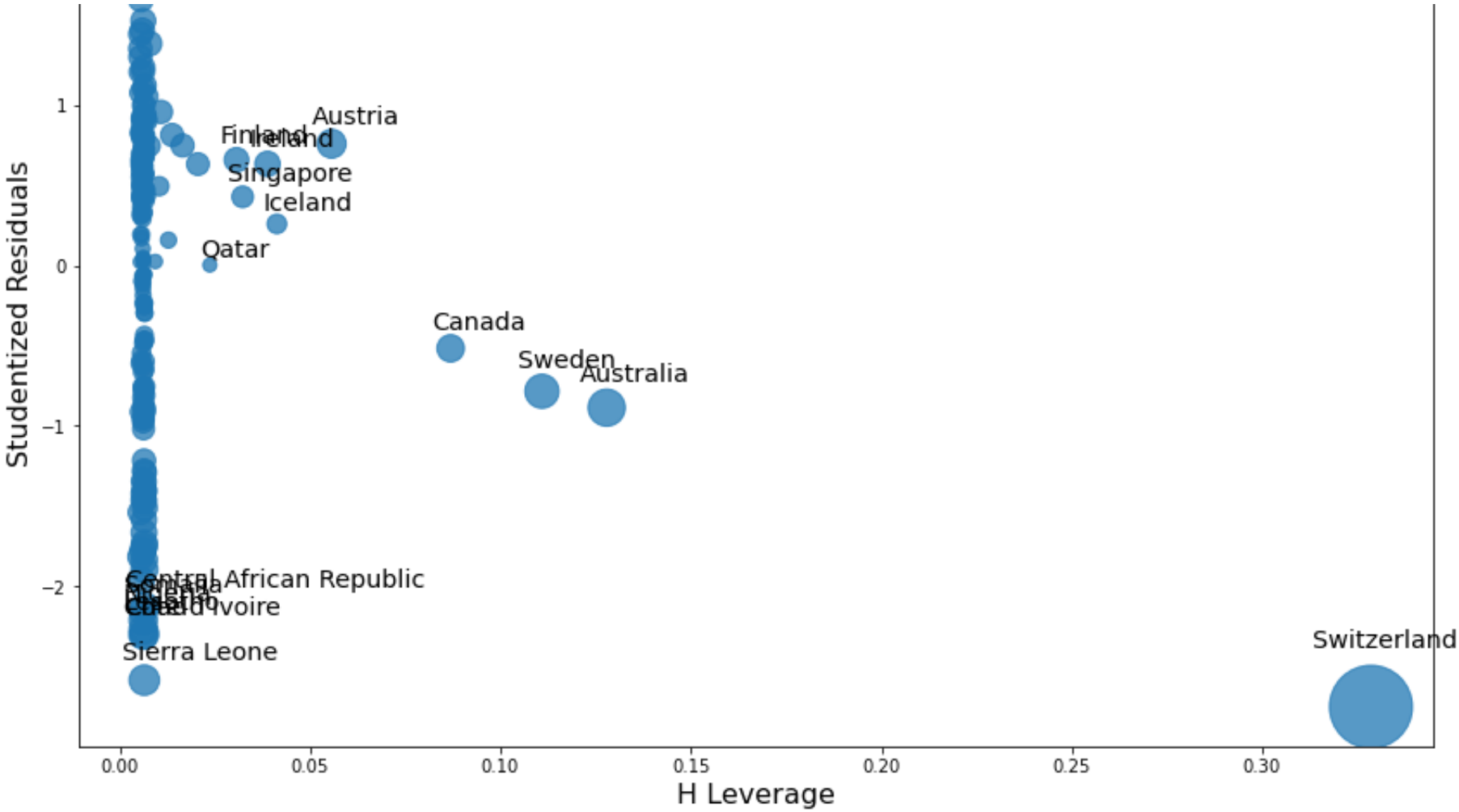

Studentized
−1
−2
−3
Nigeria
Malawi
Cote d'Ivoire
Lesotho
Sierra Leone
0.01
0.02
0.03
0.04
0.05
0.06
0.07
H Leverage

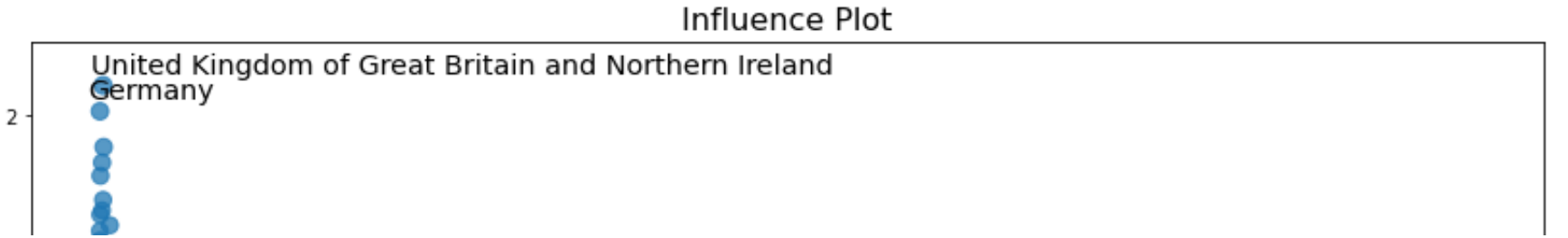

Influence Plot
Chile
Costa Rica
Iraq
Austria
Nepal
Liberia
Mauritania
Equatorial Guinea
2
1
0

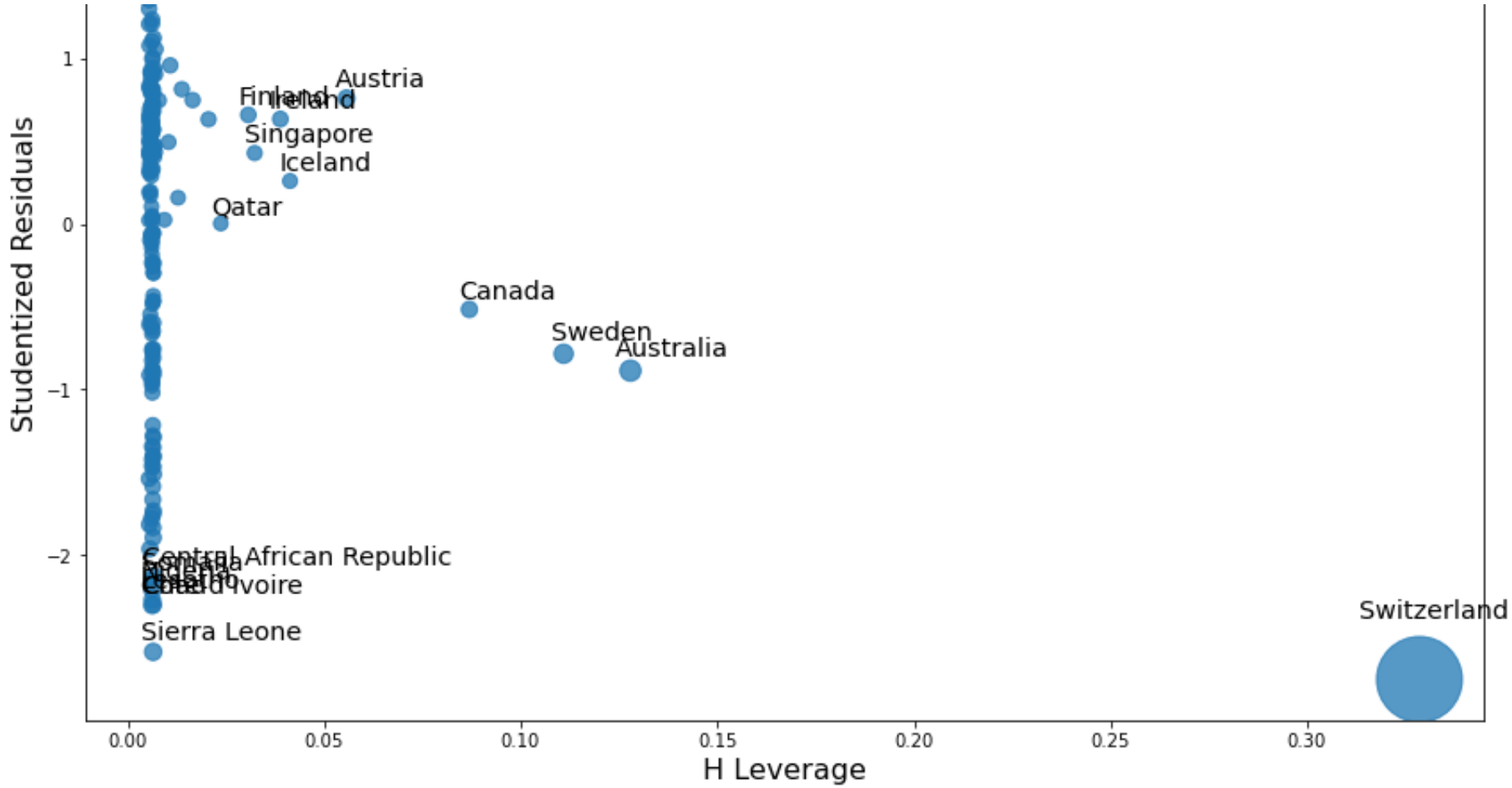


In [ ]:
```
# For percentage expenditure,
# Switzerland is an influential observation
# Countries like UK and Sierra Leone are outliers
```

In [128]:
```
# For hepatitis b
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit5, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit5, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```


Studentized Residuals
H Leverage
Austria
Finland
Ireland
Singapore
Iceland
Qatar
Canada
Sweden
Australia
Switzerland
Central African Republic
Sierra Leone
1
0
−1
−2
0.00
0.05
0.10
0.15
0.20
0.25
0.30

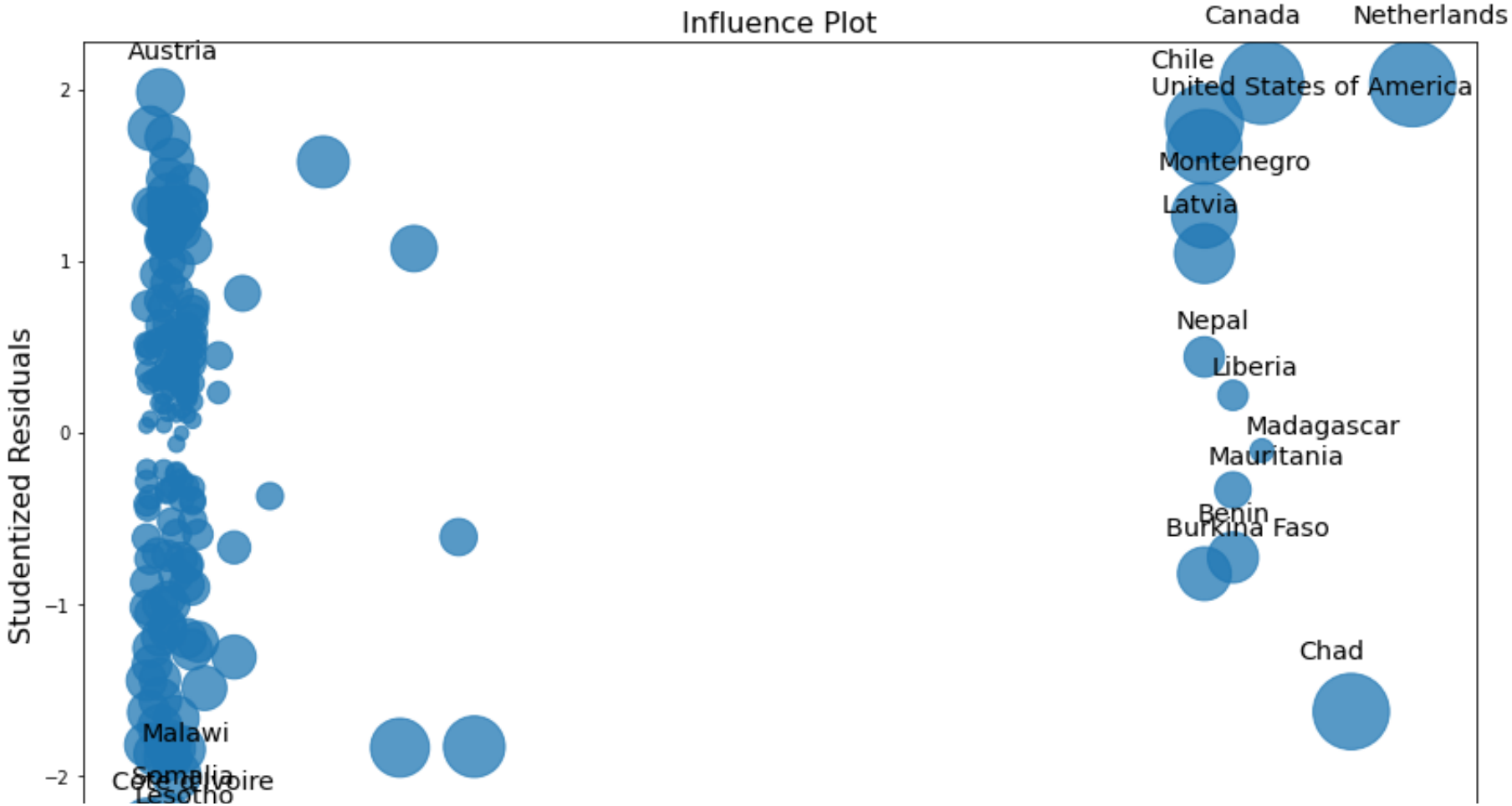

Influence Plot
United Kingdom of Great Britain and Northern Ireland
Germany
2

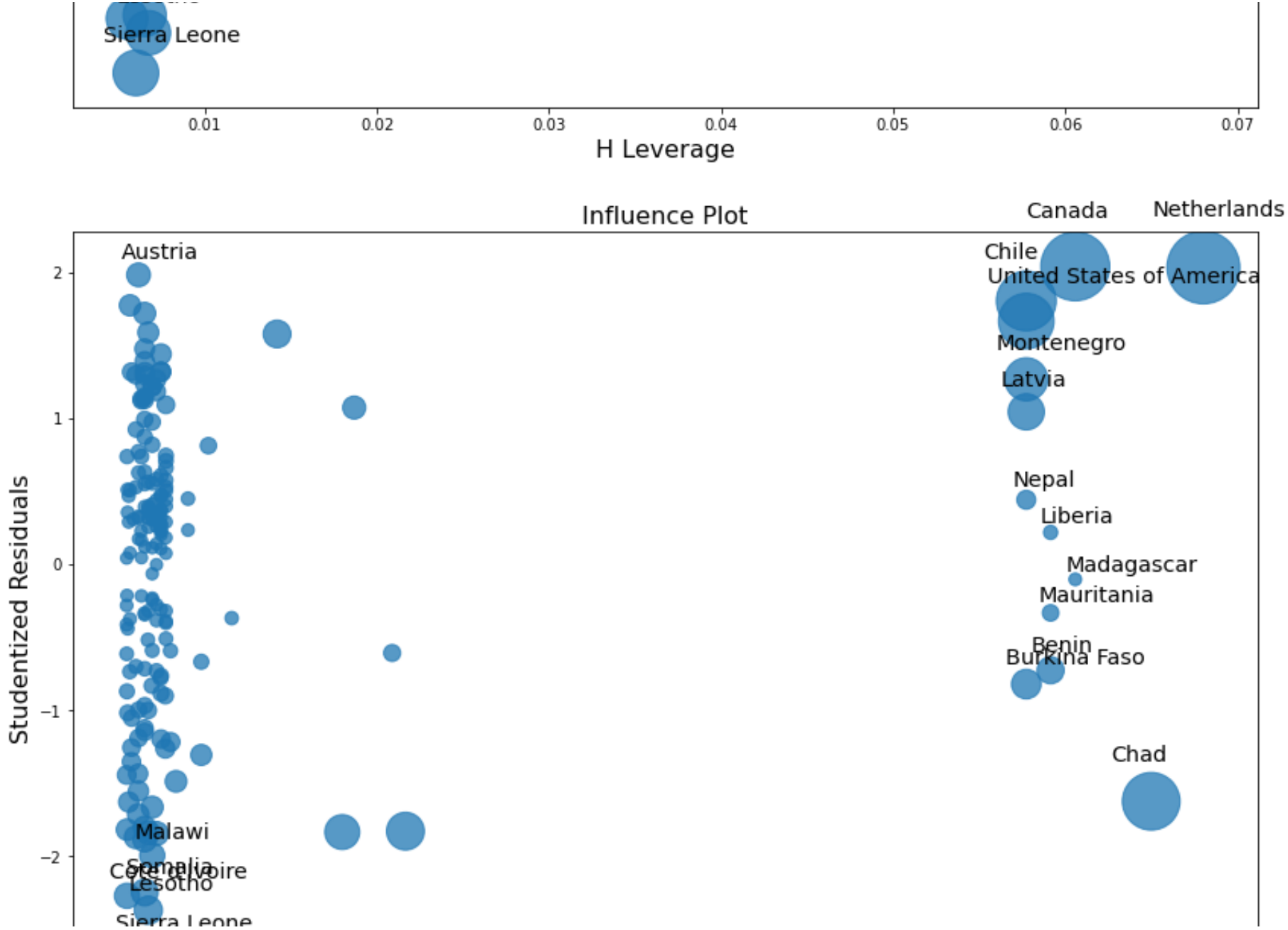


In [ ]:
```
# For hepatitis B,
# Chad, Canada, Netherlands etc are all influential observation
# Countries like Austria and Sierra Leone are outliers
```

In [129]:
```

```

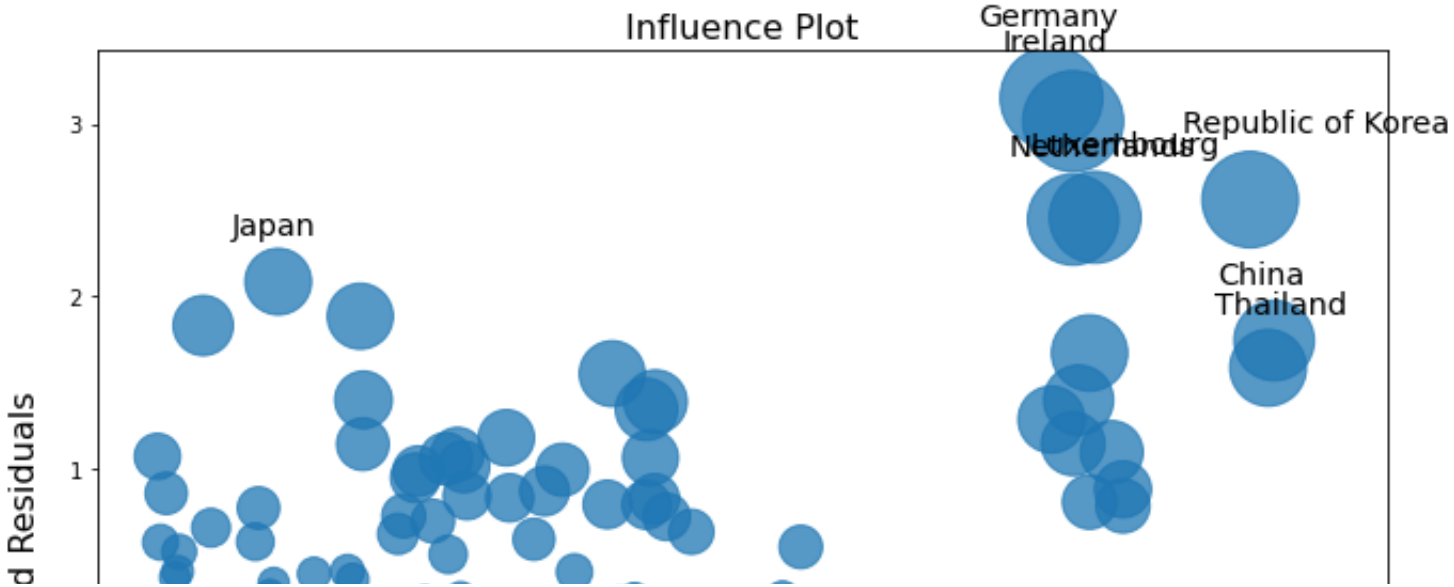
```
# For bmi
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit6, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit6, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

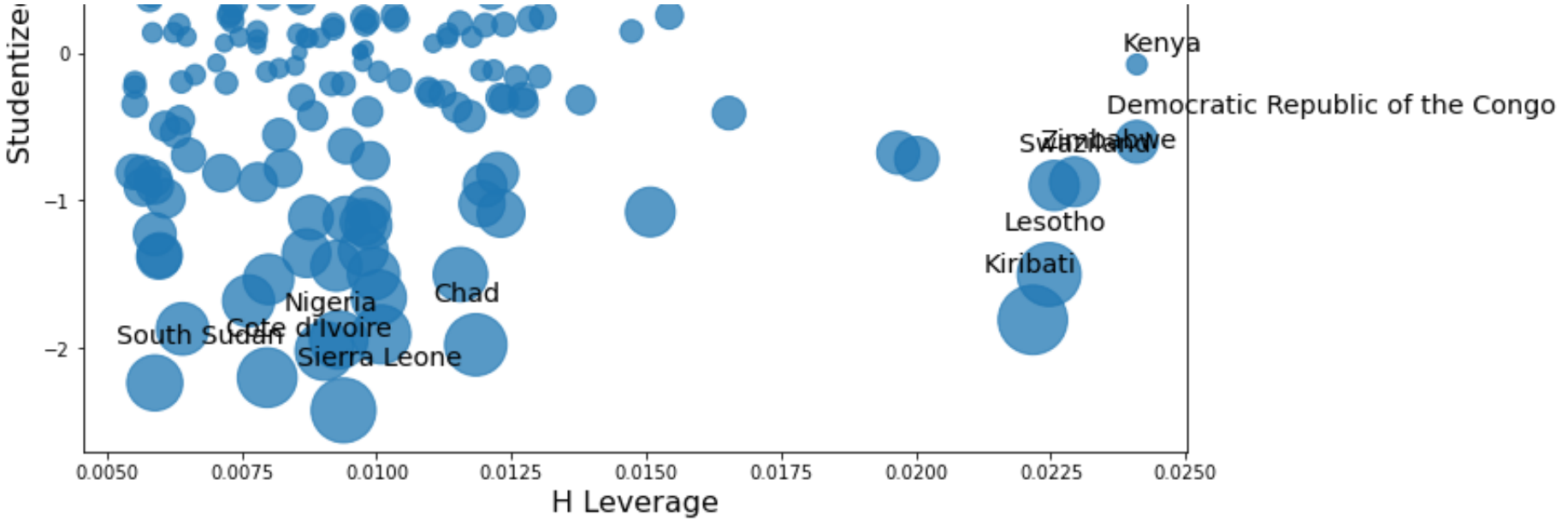


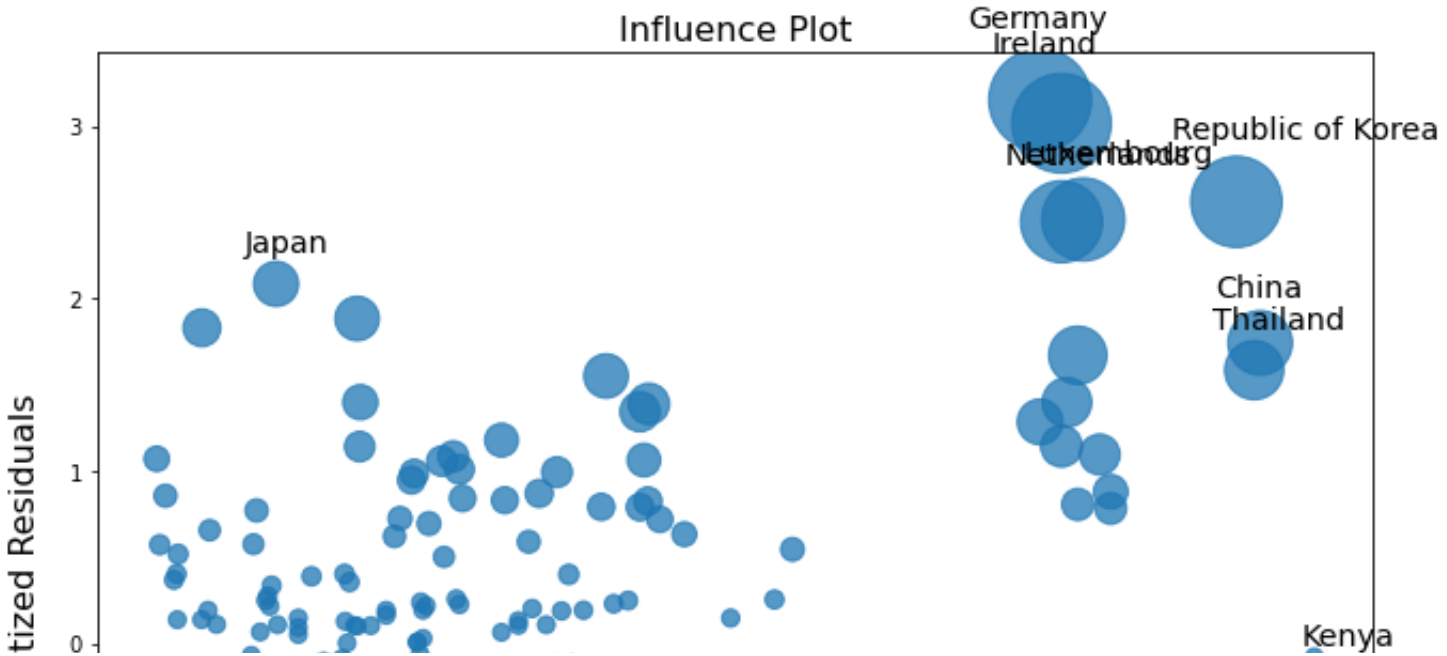

Sierra Leone
0.01
0.02
0.03
0.04
0.05
0.06
0.07
H Leverage
Influence Plot
Studentized Residuals
Austria
Canada
Netherlands
Chile
United States of America
Montenegro
Latvia
Nepal
Liberia
Madagascar
Mauritania
Benin
Burkina Faso
Chad
Malawi
Somalia
Lesotho
Sierra Leone
2
1
0
−1
−2

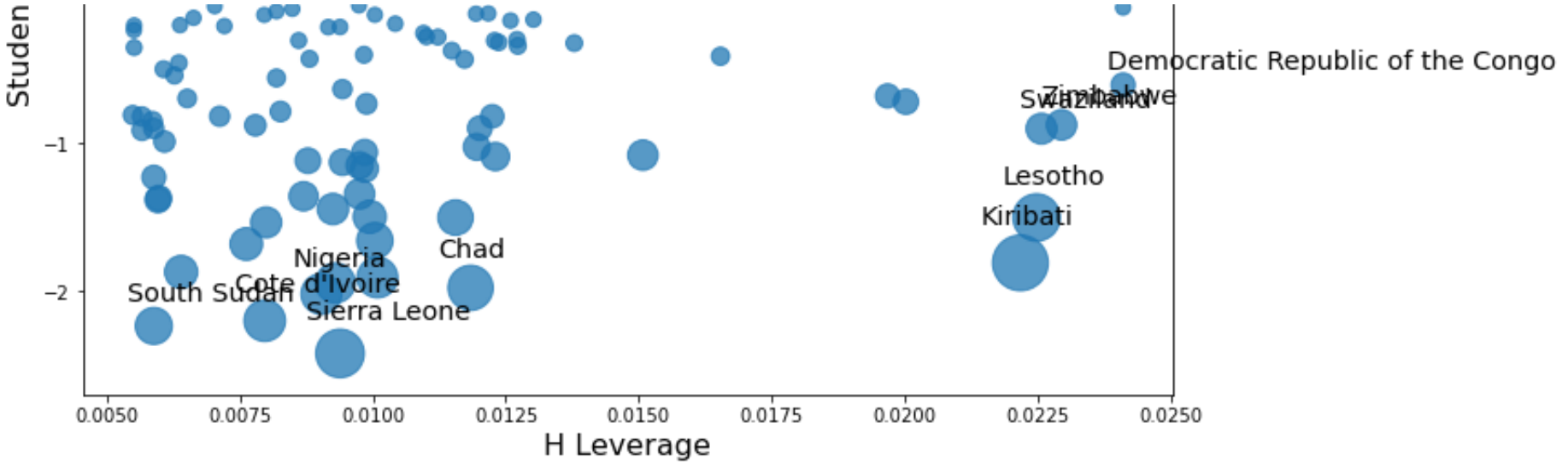
In [ ]:
```
# For bmi,
# Germany, Ireland, Republic of Korea etc are all influential observations
# Countries like Japan and Sierra Leone are outliers
```

In [130]:
```
# For under-five deaths
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit7, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit7, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```



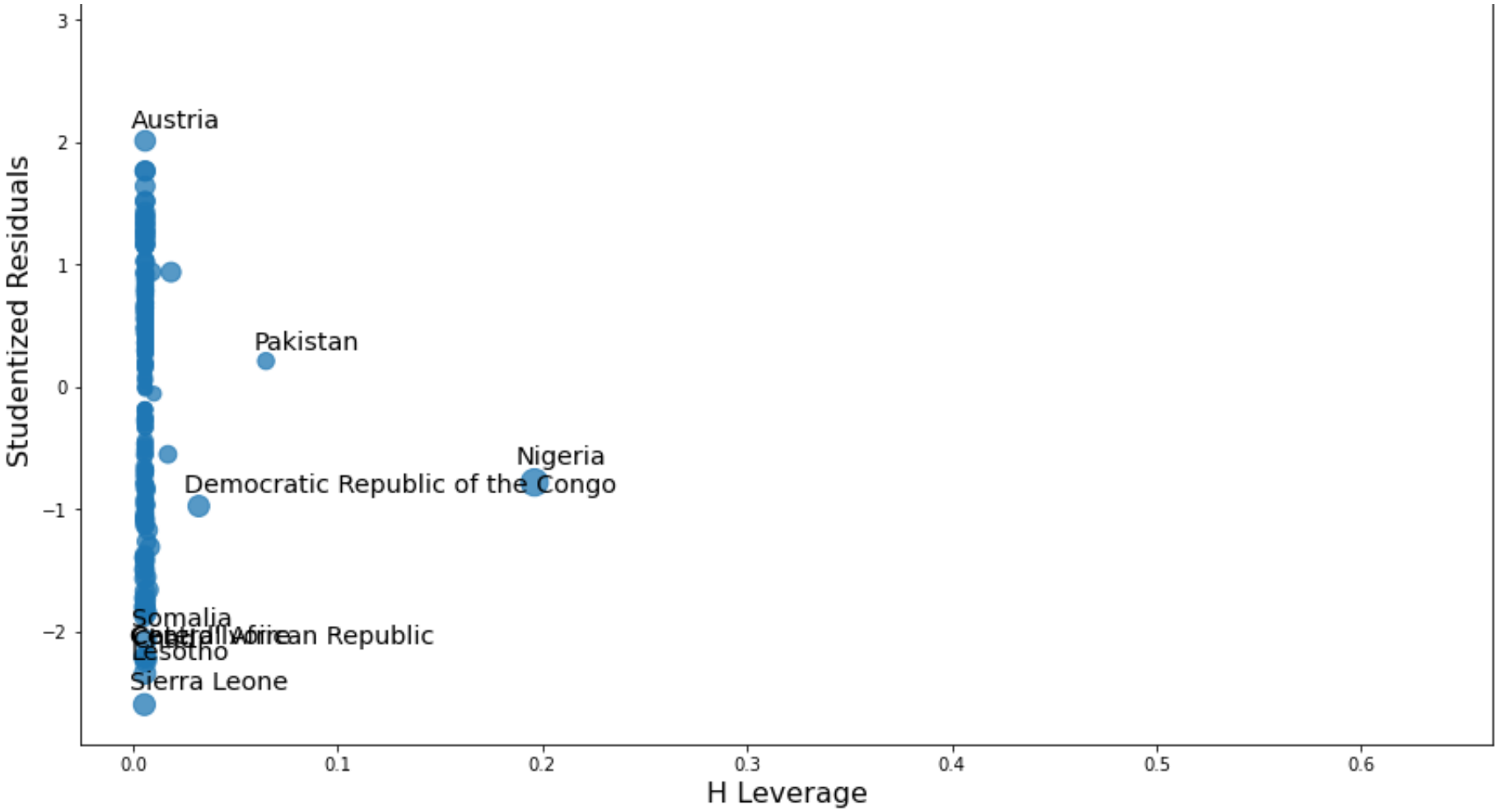

Studentize
0
−1
−2
Kenya
Democratic Republic of the Congo
Zimbabwe
Swaziland
Lesotho
Kiribati
Chad
Nigeria
Cote d'Ivoire
South Sudan
Sierra Leone
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
0.0225
0.0250
H Leverage

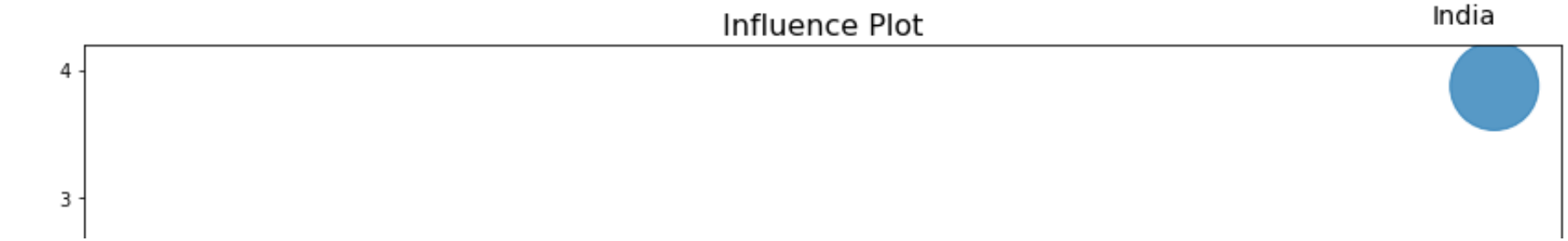

Influence Plot
Germany
Ireland
Republic of Korea
Netherlands
Luxembourg
Japan
China
Thailand
Kenya
tized Residuals
3
2
1
0

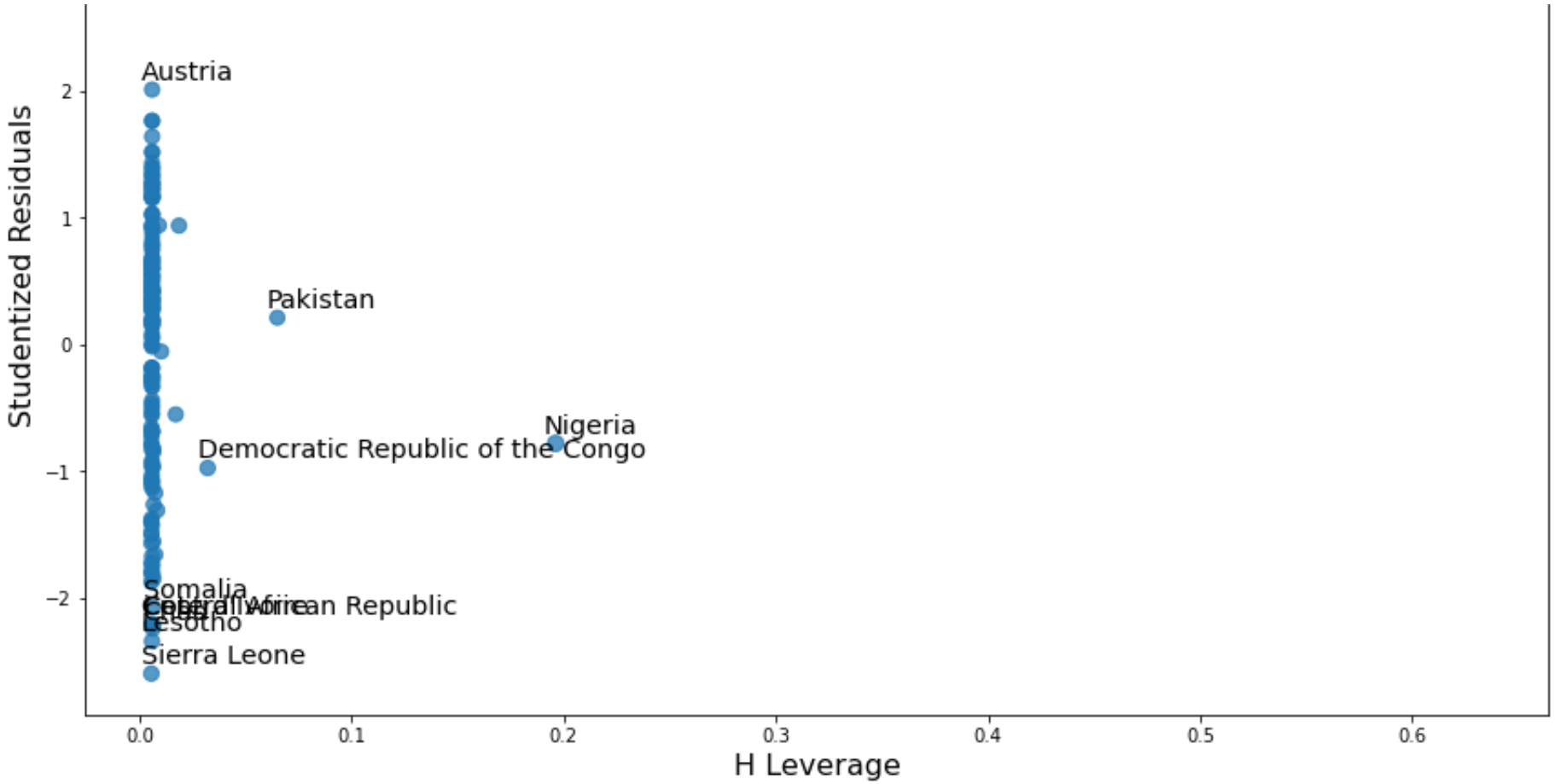


In [ ]:
```
# For under-five deaths,
# India is an influential observation
# Countries like Lesotho and Sierra Leone are outliers with large residuals
```

In [131]:
```
# For polio
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit8, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit8, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```


Studentized Residuals
H Leverage
Austria
Pakistan
Nigeria
Democratic Republic of the Congo
Somalia
Lesotho
Sierra Leone
3
2
1
0
−1
−2
0.0
0.1
0.2
0.3
0.4
0.5
0.6

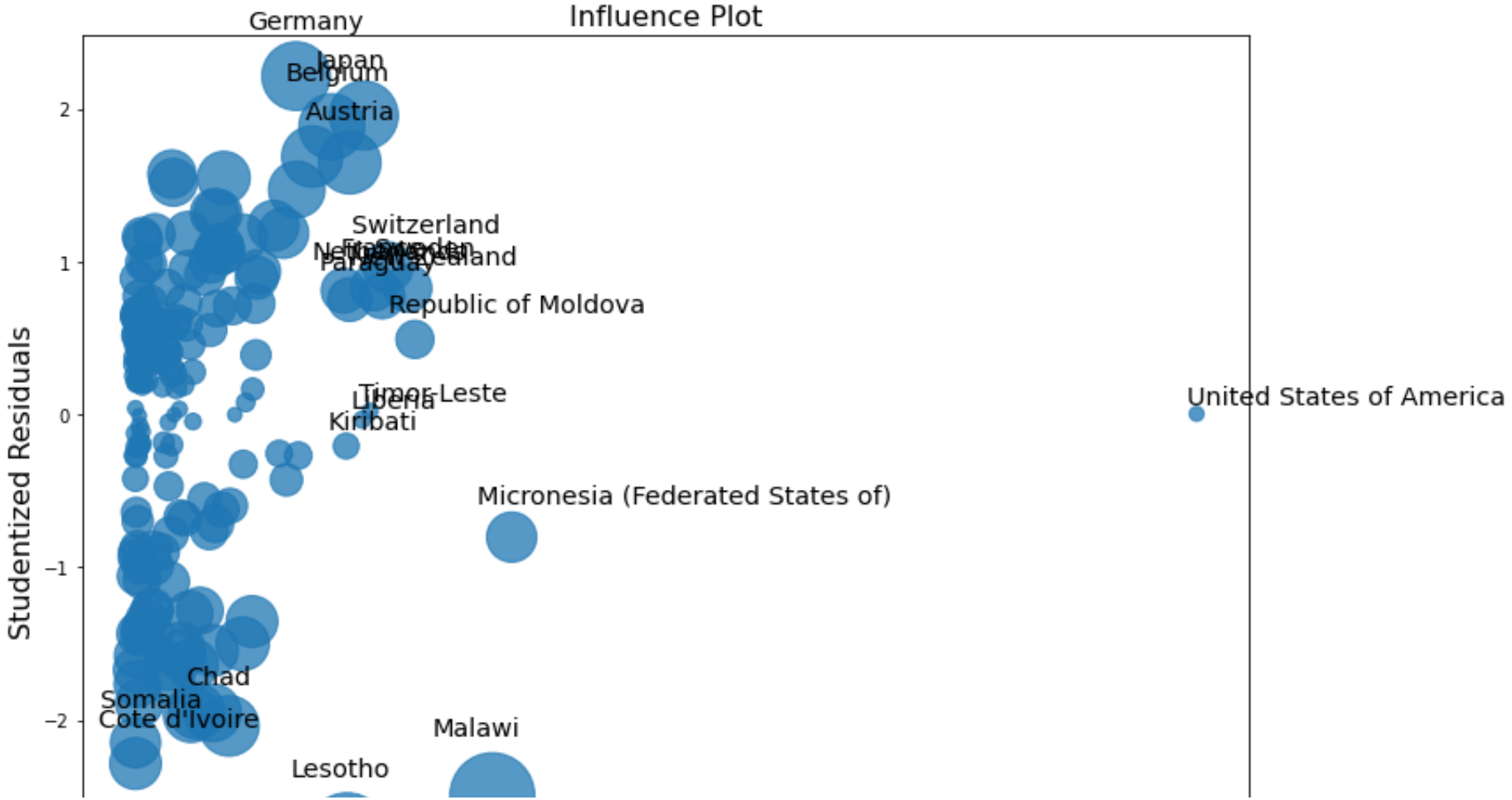

Influence Plot
India
4
3

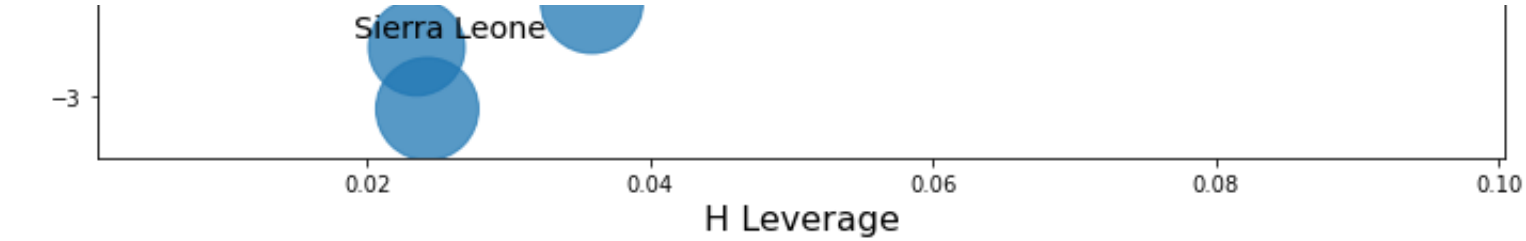


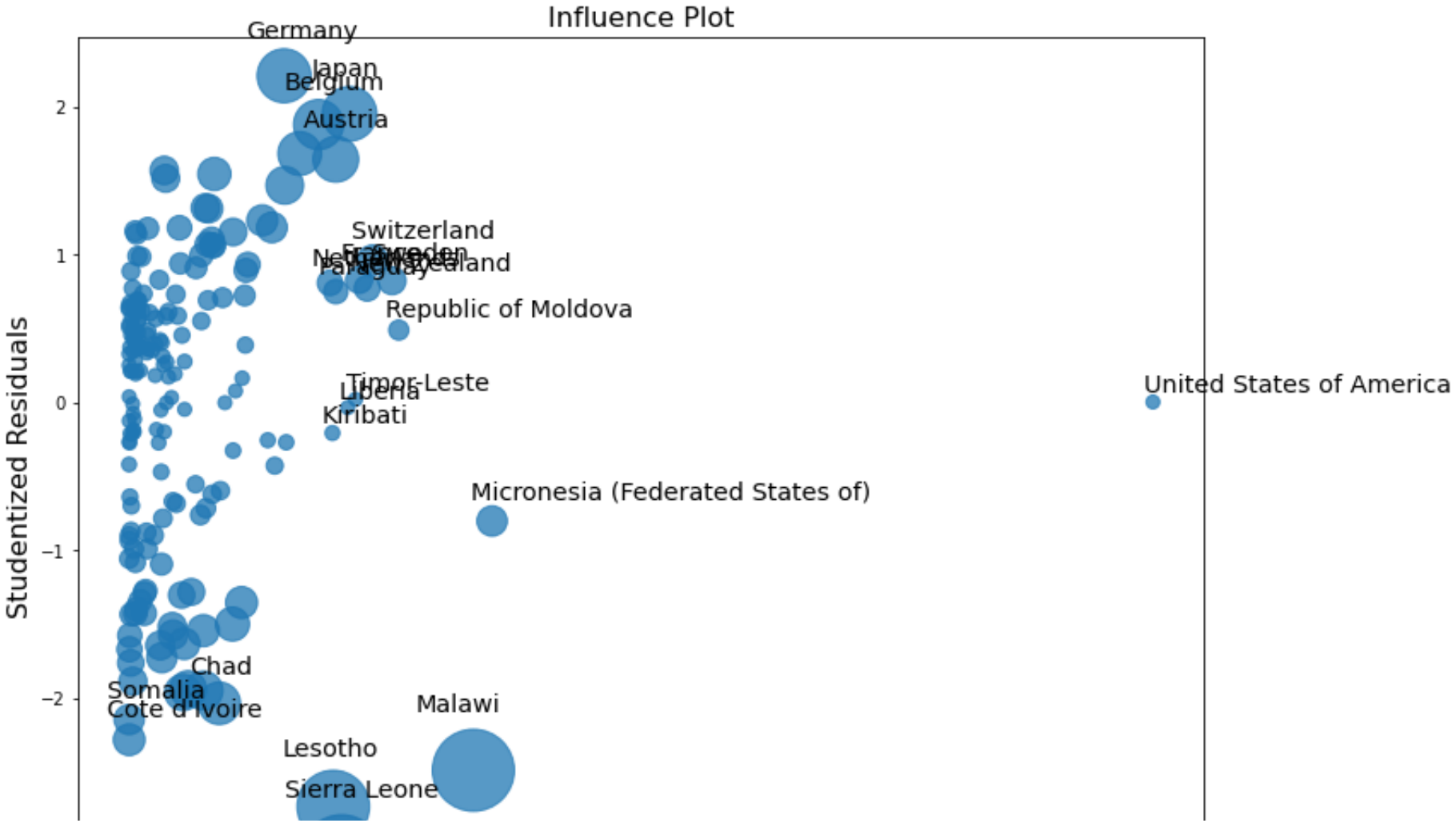
In [ ]:
```
# For polio,
# Chile, Costa Rica and Iraq are influential observations.
# Countries towards the right of the graph like Nepal, Liberia, Equitorial Guinea have high leverage
# Countries towards the top or bottom like Austria and Sierra Leone are outliers with large residuals
```

In [132]:
```

```

```
# For total expenditure
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit9, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit9, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

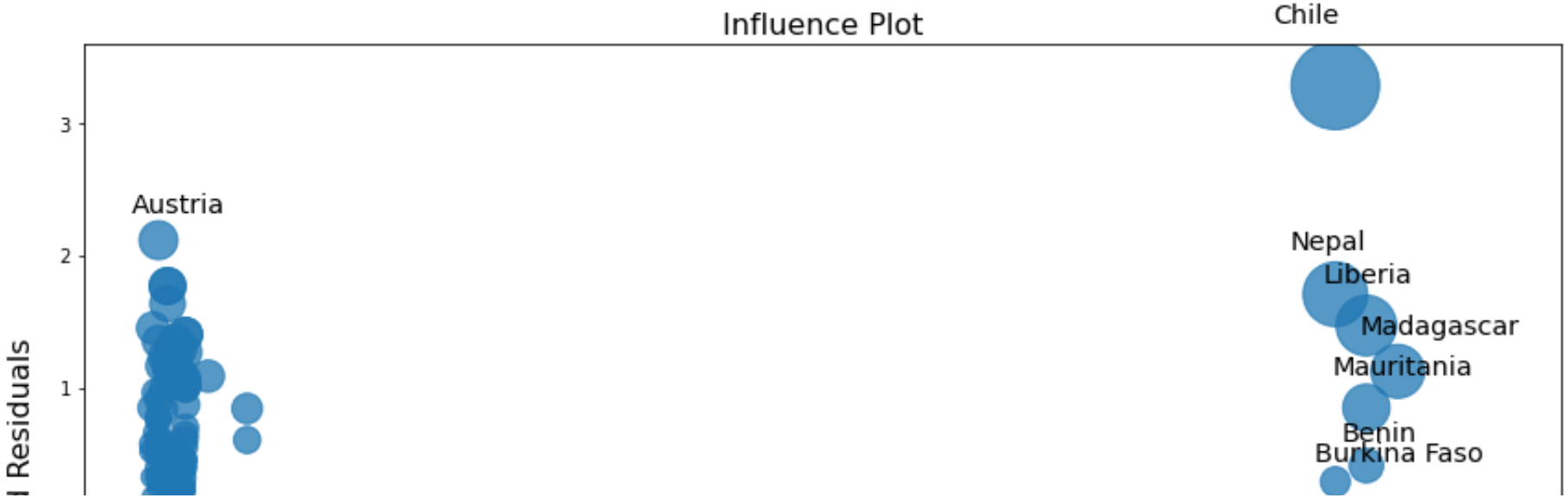


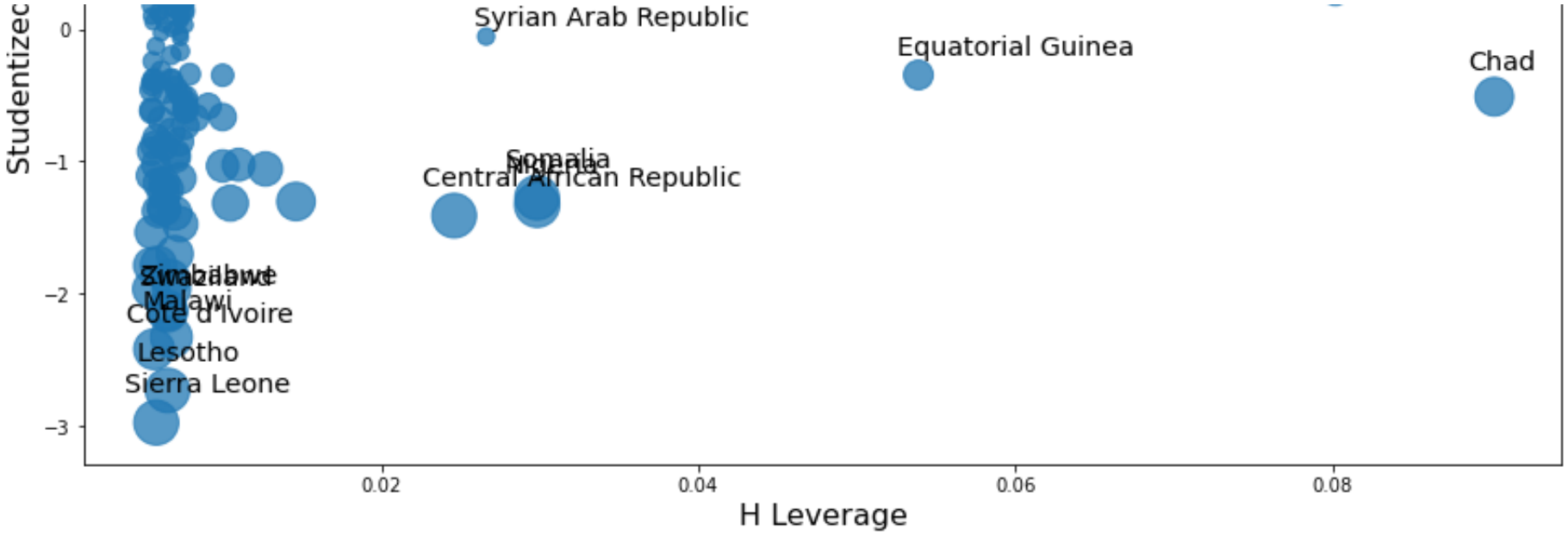

Sierra Leone
−3
0.02
0.04
0.06
0.08
0.10
H Leverage

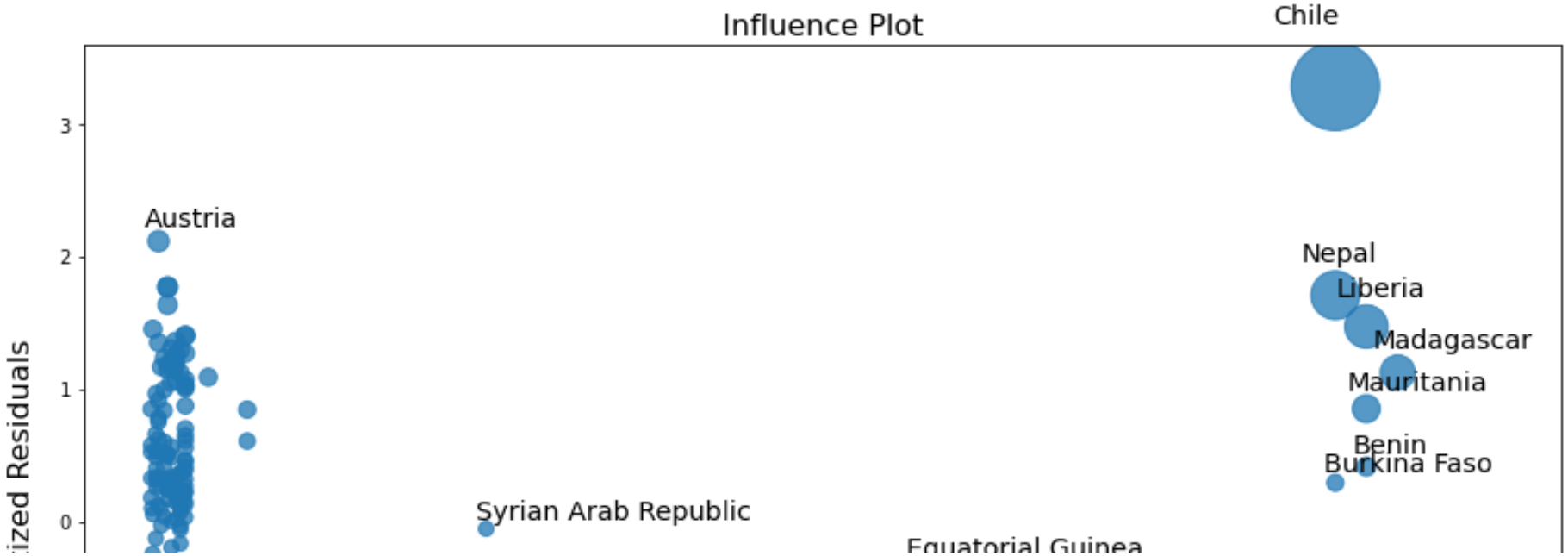

Influence Plot
Germany
Japan
Belgium
Austria
Switzerland
Republic of Moldova
Timor-Leste
Liberia
Kiribati
United States of America
Micronesia (Federated States of)
Chad
Somalia
Cote d'Ivoire
Malawi
Lesotho
Sierra Leone
Studentized Residuals
2
1
0
−1
−2

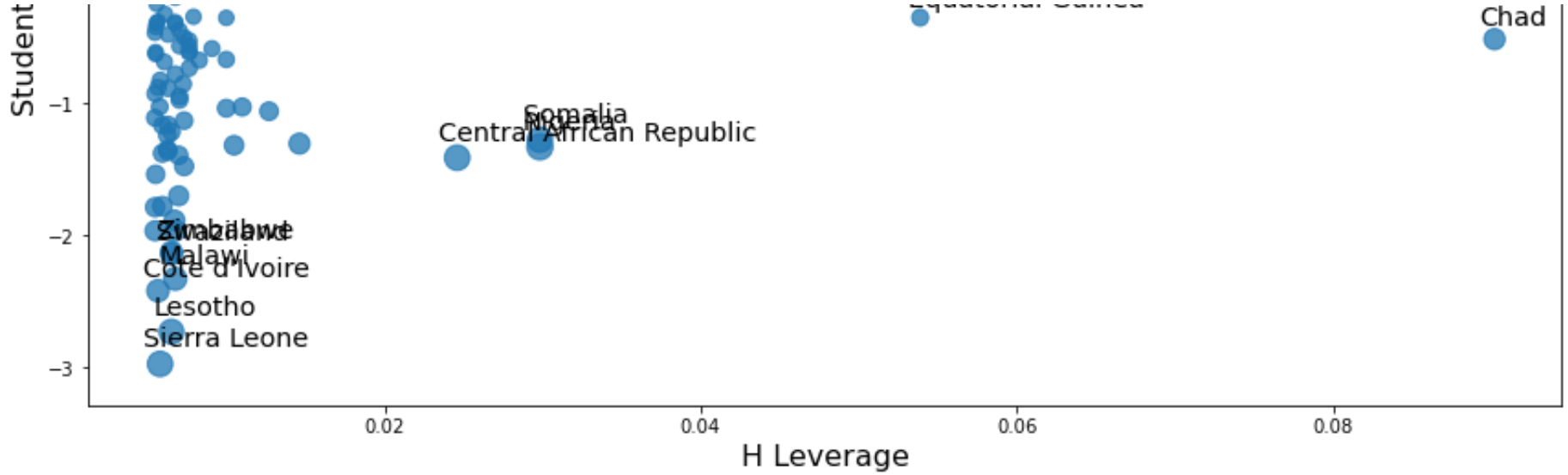
In [ ]:
```
# For total expenditure,
# USA has high leverage
# countries like Lesotho, Malawi, Germany, Japan are outliers as they have large residuals.
```

In [133]:
```
# For diphtheria
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit10, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit10, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```



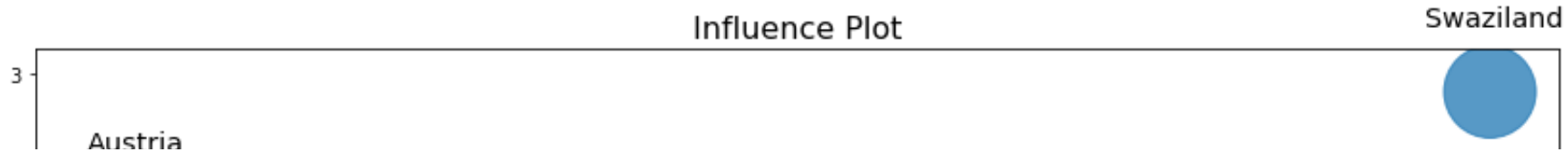

Studentizec
0
−1
−2
−3
Syrian Arab Republic
Equatorial Guinea
Chad
Somalia
Nigeria
Central African Republic
Zimbabwe
Swaziland
Malawi
Côte d'Ivoire
Lesotho
Sierra Leone
0.02
0.04
0.06
0.08
H Leverage

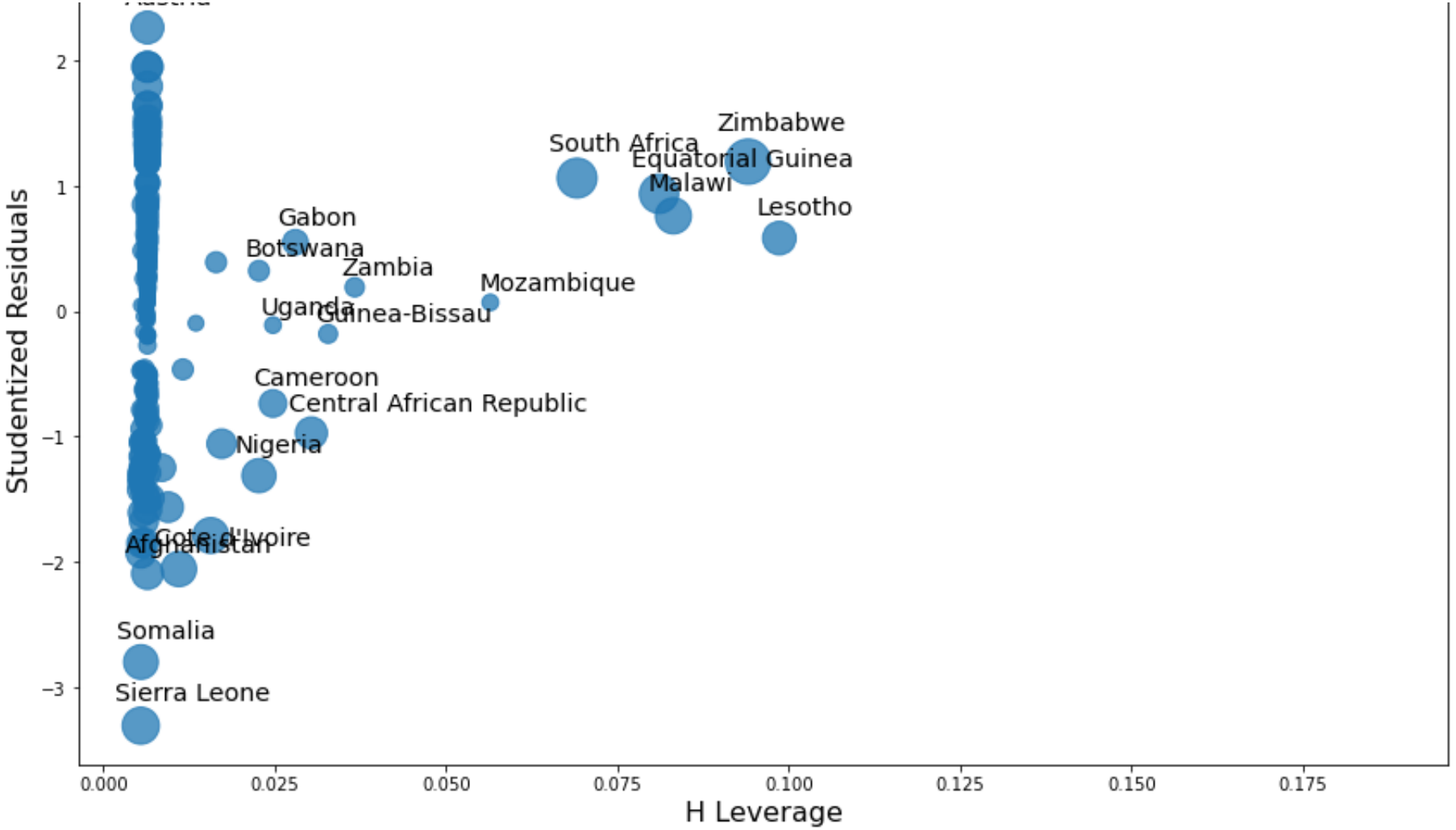

Influence Plot
Chile
3
2
1
0
Austria
Nepal
Liberia
Madagascar
Mauritania
Benin
Burkina Faso
Syrian Arab Republic
Equatorial Guinea



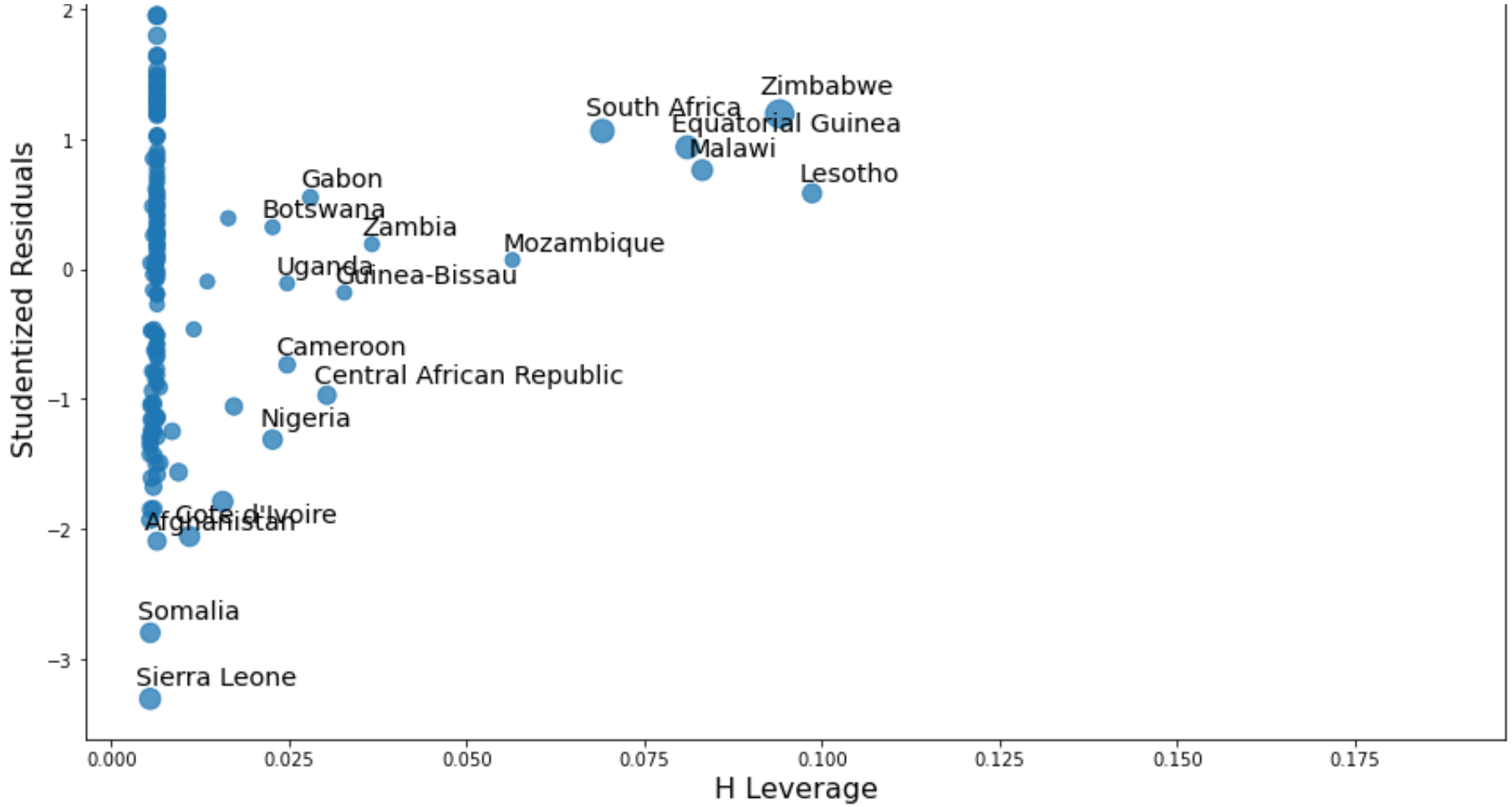
In [ ]:
```
# For diphtheria,
# Chile is an influential observation with high leverage as well as large residual.
# Countries like Chad, Nepal, Liberia, Madagascar also have high leverage but relatively smaller residua
# Lesotho, Sierra Leone, Austria etc are outliers
```

In [134]:
```
# For hiv/aids
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit11, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit11, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```




Studentized Residuals
H Leverage
Zimbabwe
South Africa
Equatorial Guinea
Malawi
Lesotho
Gabon
Botswana
Zambia
Mozambique
Uganda
Guinea-Bissau
Cameroon
Central African Republic
Nigeria
Somalia
Sierra Leone
2
1
0
−1
−2
−3
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175

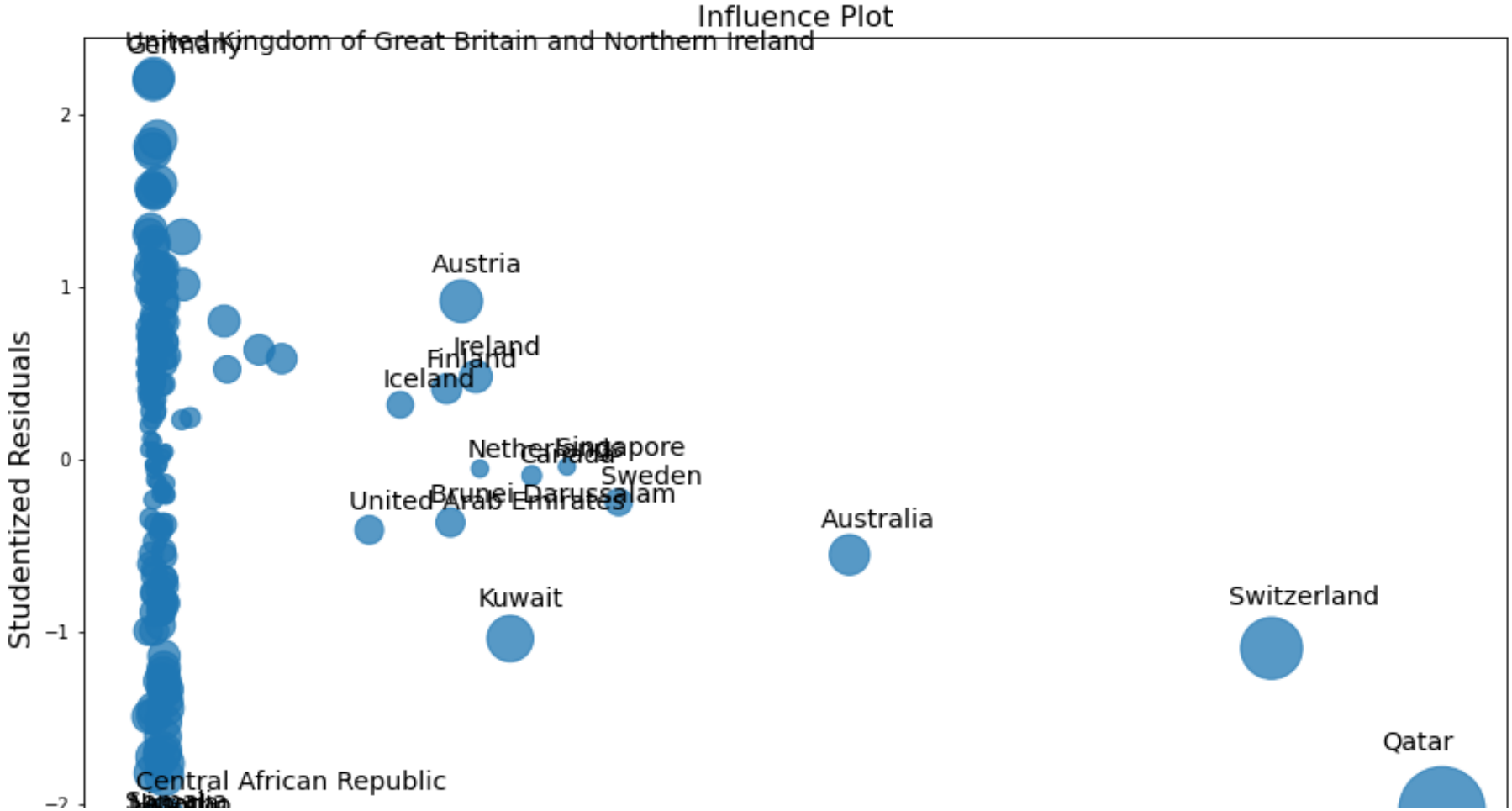

Influence Plot
Swaziland
Austria
3

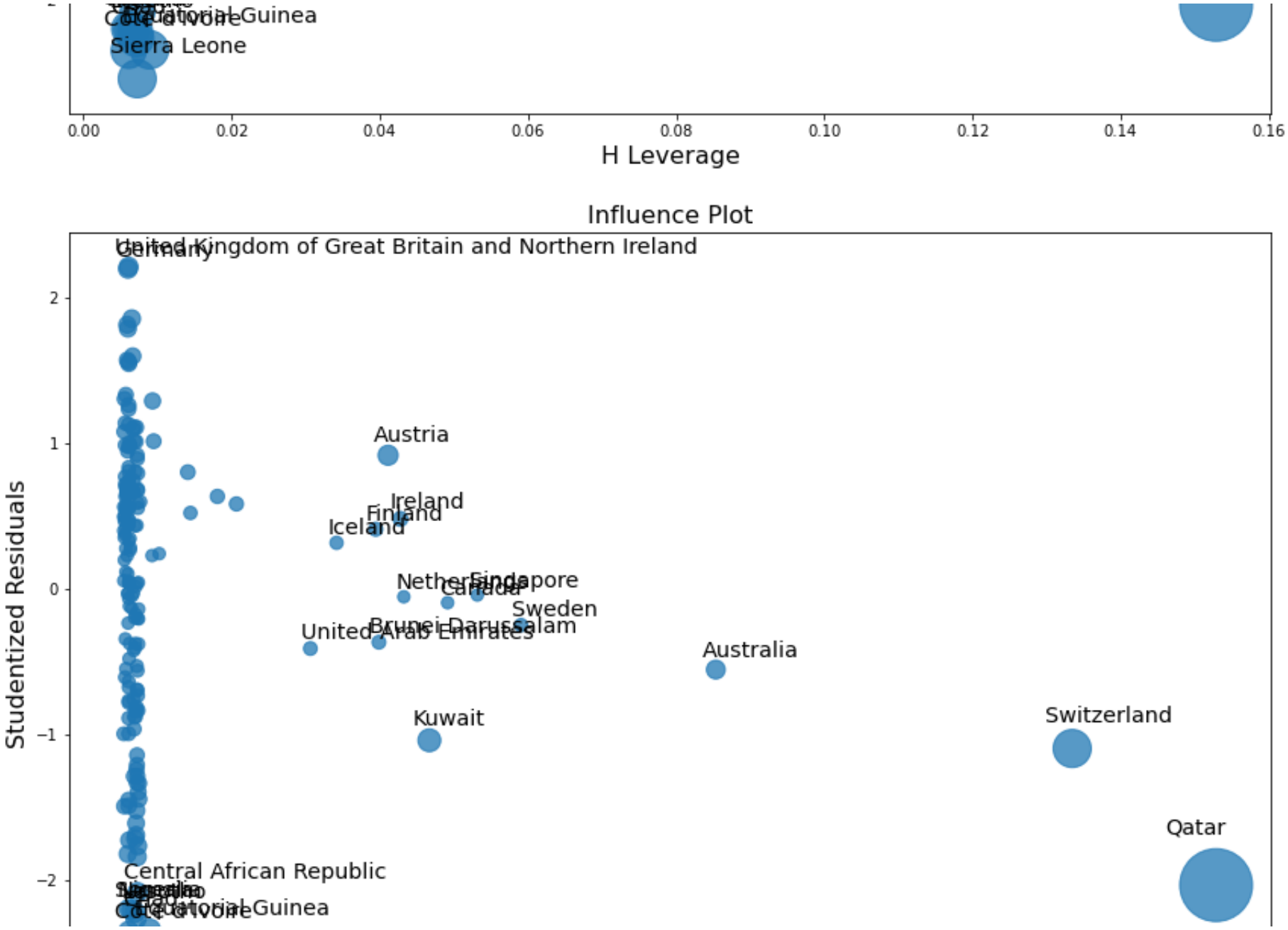


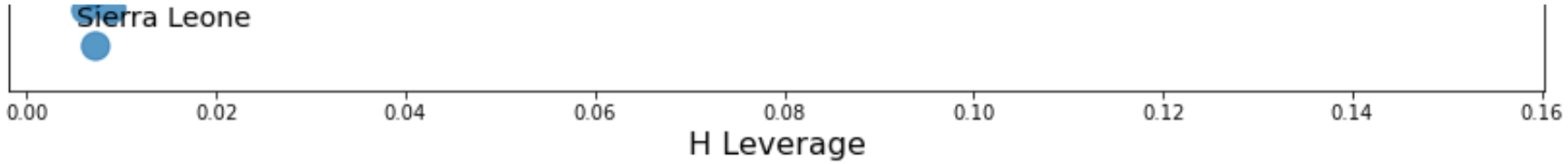
In [ ]:
```
# For hiv/aids,
# Swaziland is an influential observation with high leverage as well as large residual
```

In [135]:
```

```

```
# For gdp
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit12, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit12, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```

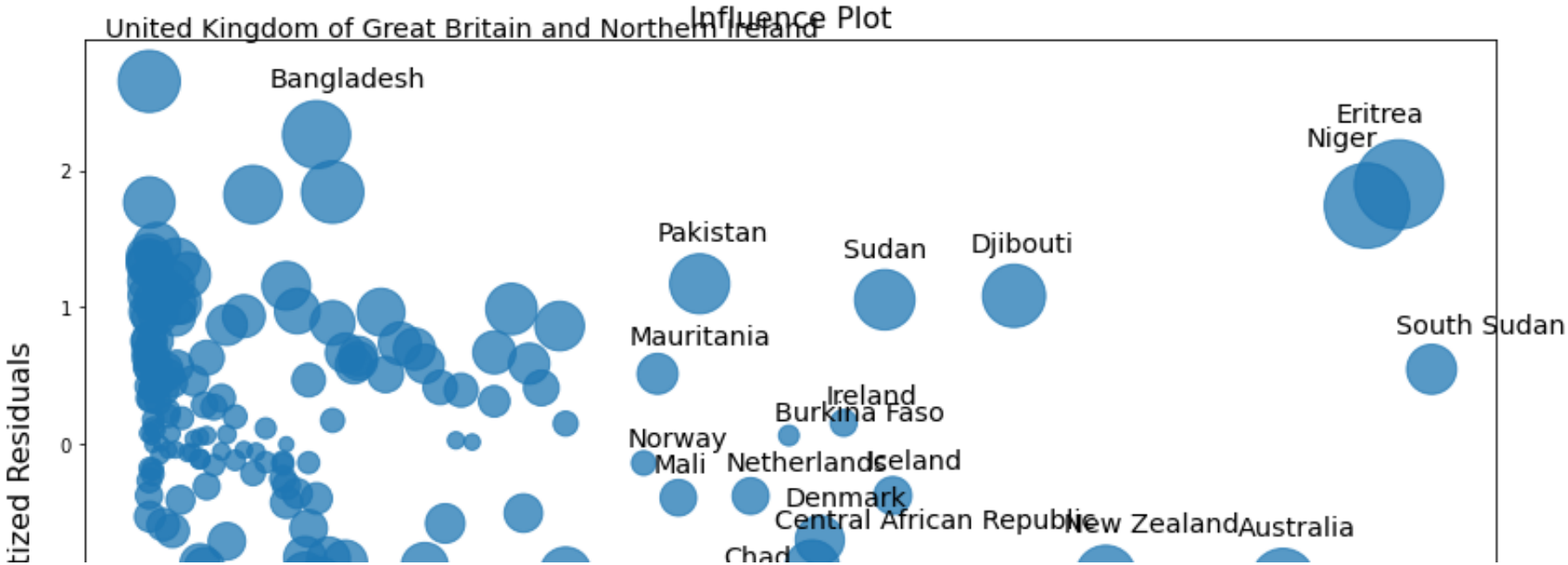


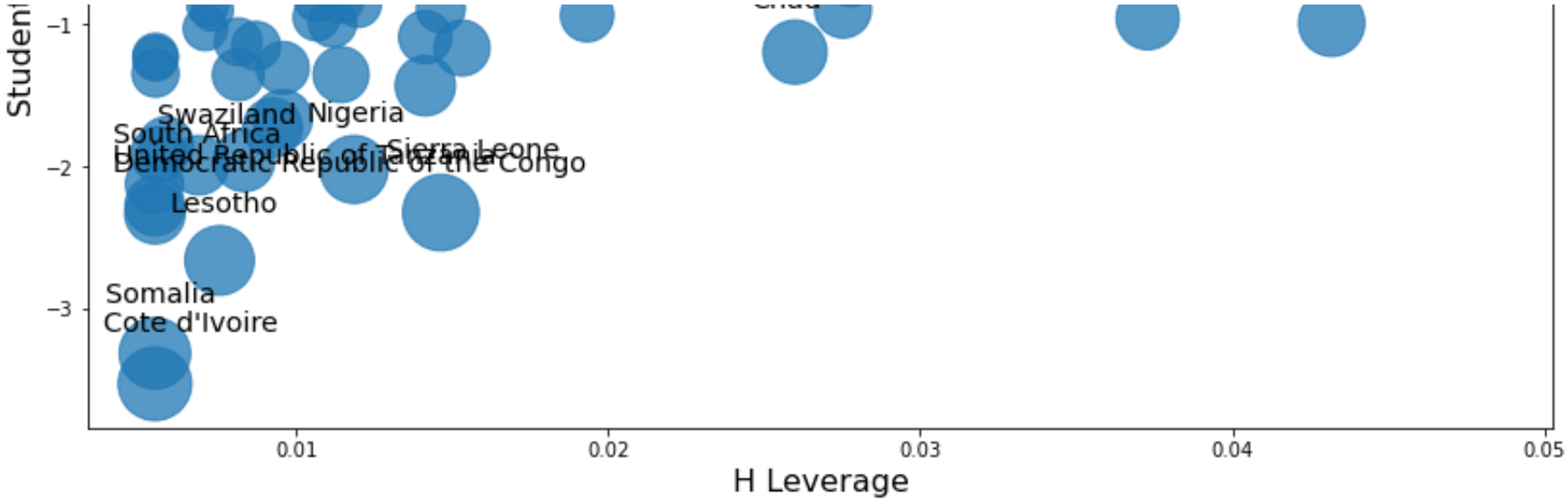

Sierra Leone
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
H Leverage
Influence Plot
United Kingdom of Great Britain and Northern Ireland
Germany
Austria
Ireland
Finland
Iceland
Sweden
Australia
Kuwait
Switzerland
Qatar
Central African Republic
Studentized Residuals
2
1
0
−1
−2

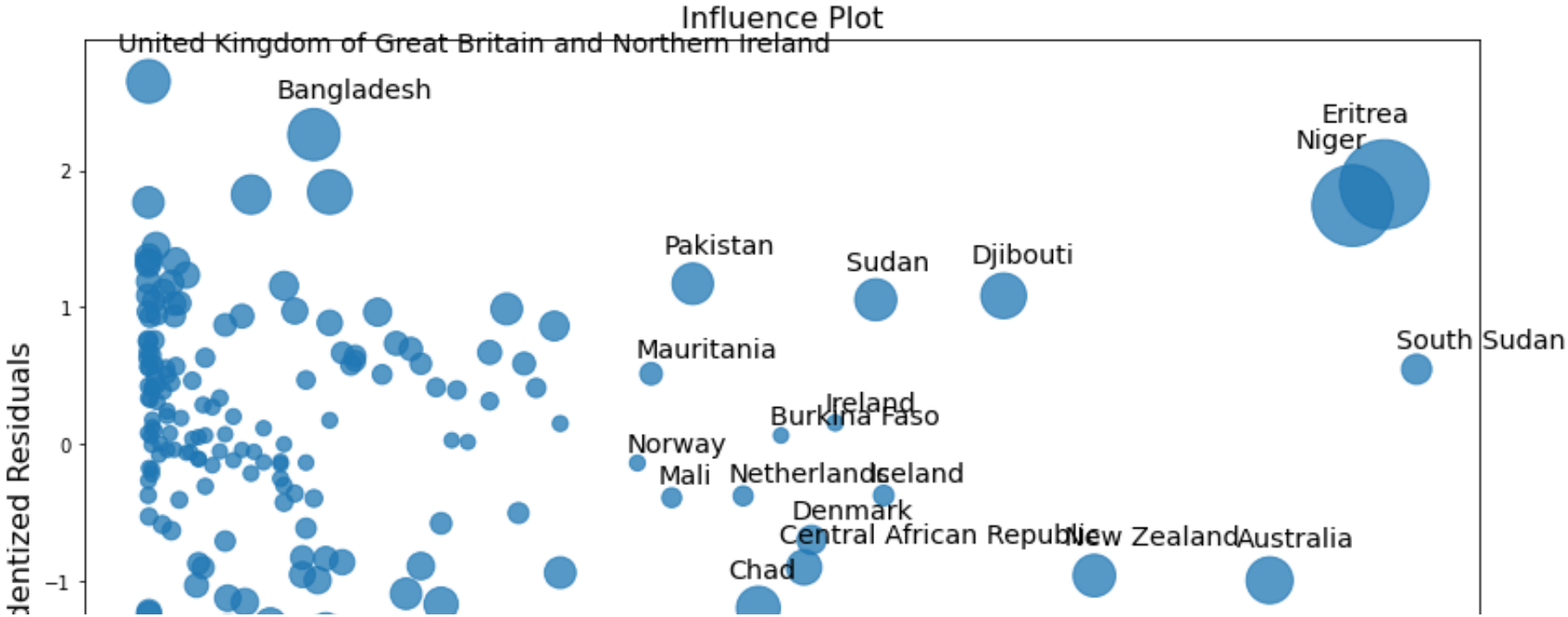


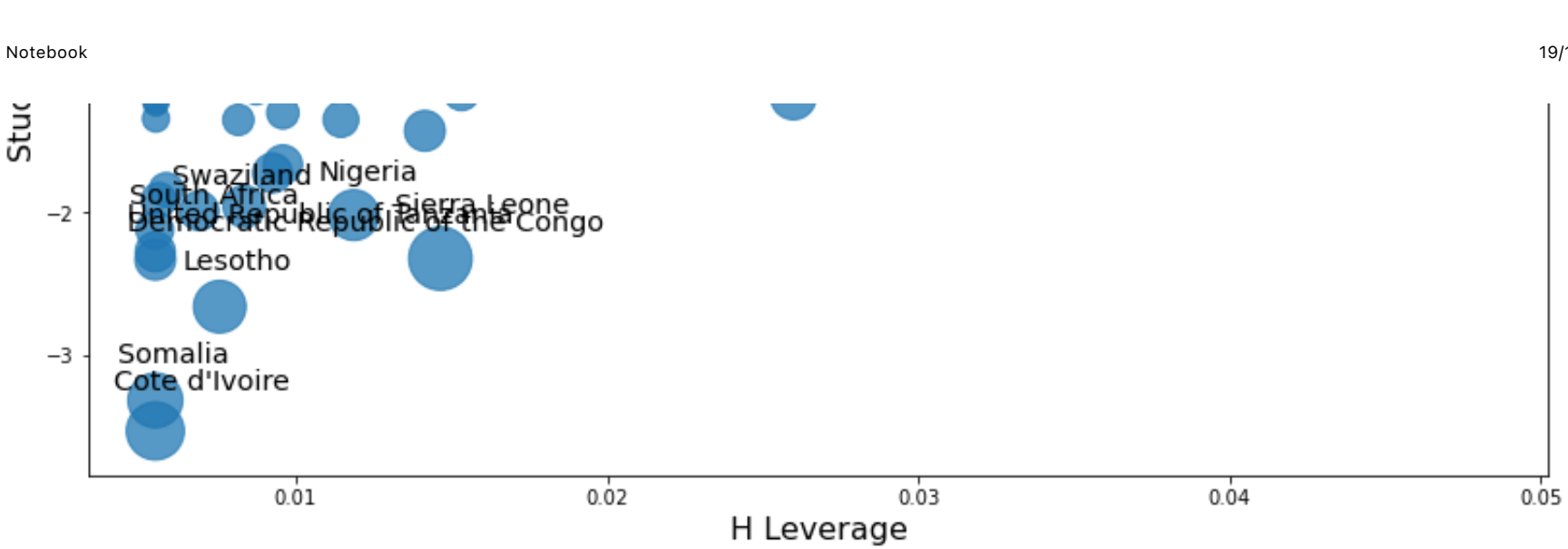
In [ ]:
```
# For gdp, we see that Qatar is an influential observation
```

In [136]:
```
# For schooling
figd, ax = plt.subplots(figsize=(12,8))
figd = sms.graphics.influence_plot(ols_fit13, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(12,8))
fige = sms.graphics.influence_plot(ols_fit13, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```




Swaziland
Nigeria
South Africa
Sierra Leone
Lesotho
Somalia
Cote d'Ivoire
0.01
0.02
0.03
0.04
0.05
H Leverage


Influence Plot
United Kingdom of Great Britain and Northern Ireland
Bangladesh
Eritrea
Niger
Pakistan
Sudan
Djibouti
South Sudan
Mauritania
Ireland
Burkina Faso
Norway
Mali
Netherlands
Iceland
Denmark
Central African Republic
New Zealand
Australia
Chad
dentized Residuals



In [ ]:
```
# For schooling,
# Eritrea, Niger are influential observations with high leverage as well as large residuals
# South Sudan, Australia have high leverage
# Lesotho, Bangladesh, Sierra Leone etc are outliers
```

Part e

Our data has several NA values, and having NA values in the data can adversely affect the analysis. A common method used to impute the NA values is to replace them by the mean. In our dataset, however, the mean is not a good measure of central tendency because our data is not normally distributed and there are several extreme values as shown in the analysis above. Therefore, we replaced the NA values with the median.

Question 2

Part a

Boruta's Algorithm

In [137]:
```
boruta_data = df[["life_expectancy","schooling", "gdp","hiv_aids", "diphtheria", "total_expenditure","po
```

In [138]:
```
pip install BorutaShap
```

```
Requirement already satisfied: BorutaShap in ./opt/anaconda3/lib/python3.9/site-packages (1.0.16)
Requirement already satisfied: tqdm in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaShap) (4
.64.0)
Requirement already satisfied: shap>=0.34.0 in ./opt/anaconda3/lib/python3.9/site-packages (from Boruta
Shap) (0.41.0)
Requirement already satisfied: statsmodels in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaS
hap) (0.13.2)
Requirement already satisfied: scikit-learn in ./opt/anaconda3/lib/python3.9/site-packages (from Boruta
Shap) (1.0.2)
Requirement already satisfied: pandas in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaShap)
(1.4.2)
Requirement already satisfied: scipy in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaShap) (
1.7.3)
Requirement already satisfied: numpy in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaShap) (
1.21.5)
Requirement already satisfied: seaborn in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaShap)
(0.11.2)
Requirement already satisfied: matplotlib in ./opt/anaconda3/lib/python3.9/site-packages (from BorutaSh
ap) (3.5.1)
Requirement already satisfied: slicer==0.0.7 in ./opt/anaconda3/lib/python3.9/site-packages (from shap>
=0.34.0->BorutaShap) (0.0.7)
```

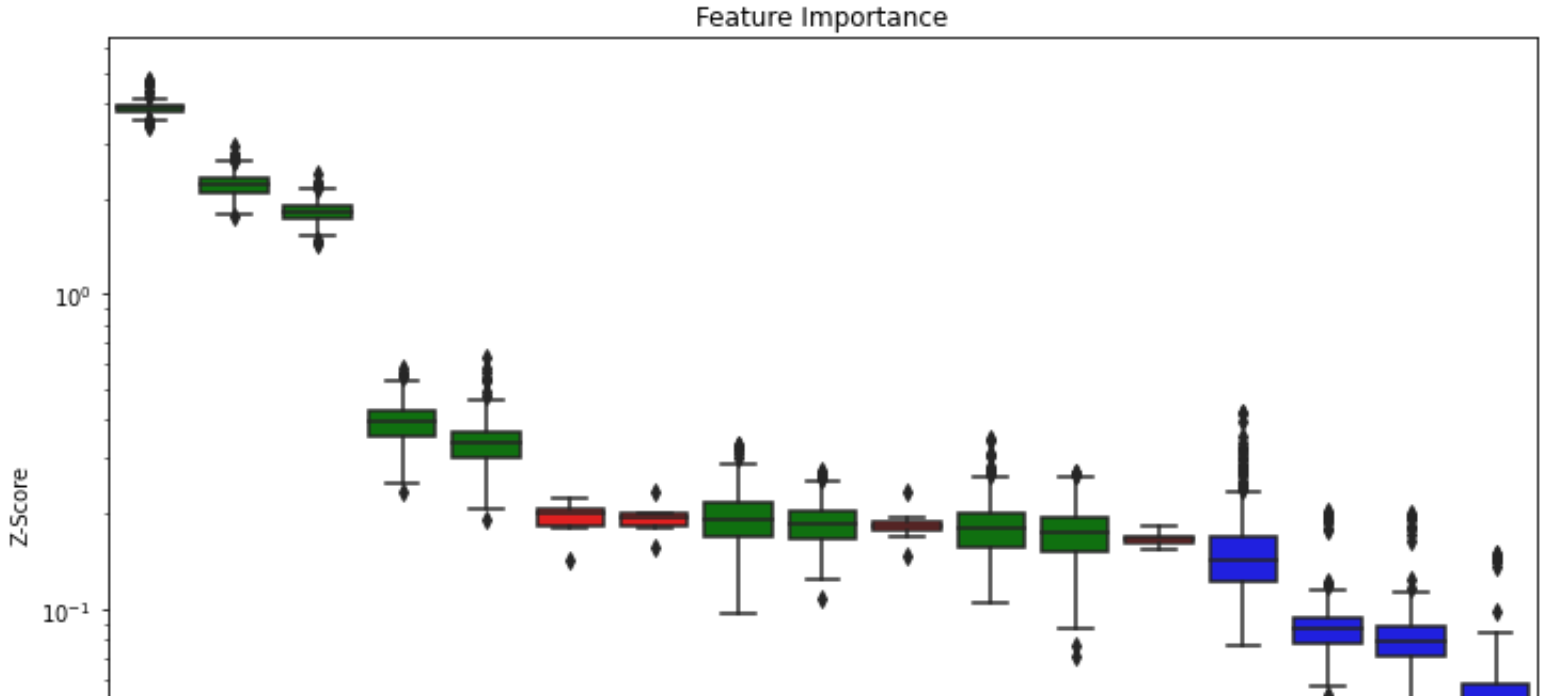
```
Requirement already satisfied: numba in ./opt/anaconda3/lib/python3.9/site-packages (from shap>=0.34.0-
>BorutaShap) (0.55.1)
Requirement already satisfied: packaging>20.9 in ./opt/anaconda3/lib/python3.9/site-packages (from shap
>=0.34.0->BorutaShap) (21.3)
Requirement already satisfied: cloudpickle in ./opt/anaconda3/lib/python3.9/site-packages (from shap>=0
.34.0->BorutaShap) (2.0.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in ./opt/anaconda3/lib/python3.9/site-packages
(from packaging>20.9->shap>=0.34.0->BorutaShap) (3.0.4)
Requirement already satisfied: pillow>=6.2.0 in ./opt/anaconda3/lib/python3.9/site-packages (from matpl
otlib->BorutaShap) (9.0.1)
Requirement already satisfied: python-dateutil>=2.7 in ./opt/anaconda3/lib/python3.9/site-packages (fro
m matplotlib->BorutaShap) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in ./opt/anaconda3/lib/python3.9/site-packages (from m
atplotlib->BorutaShap) (1.3.2)
Requirement already satisfied: cycler>=0.10 in ./opt/anaconda3/lib/python3.9/site-packages (from matplo
tlib->BorutaShap) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./opt/anaconda3/lib/python3.9/site-packages (from m
atplotlib->BorutaShap) (4.25.0)
Requirement already satisfied: six>=1.5 in ./opt/anaconda3/lib/python3.9/site-packages (from python-dat
eutil>=2.7->matplotlib->BorutaShap) (1.16.0)
Requirement already satisfied: setuptools in ./opt/anaconda3/lib/python3.9/site-packages (from numba->s
hap>=0.34.0->BorutaShap) (61.2.0)
Requirement already satisfied: llvmlite<0.39,>=0.38.0rc1 in ./opt/anaconda3/lib/python3.9/site-packages
(from numba->shap>=0.34.0->BorutaShap) (0.38.0)
Requirement already satisfied: pytz>=2020.1 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas
->BorutaShap) (2021.3)
Requirement already satisfied: joblib>=0.11 in ./opt/anaconda3/lib/python3.9/site-packages (from scikit
-learn->BorutaShap) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (fro
m scikit-learn->BorutaShap) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in ./opt/anaconda3/lib/python3.9/site-packages (from statsm
odels->BorutaShap) (0.5.2)
Note: you may need to restart the kernel to use updated packages.
```

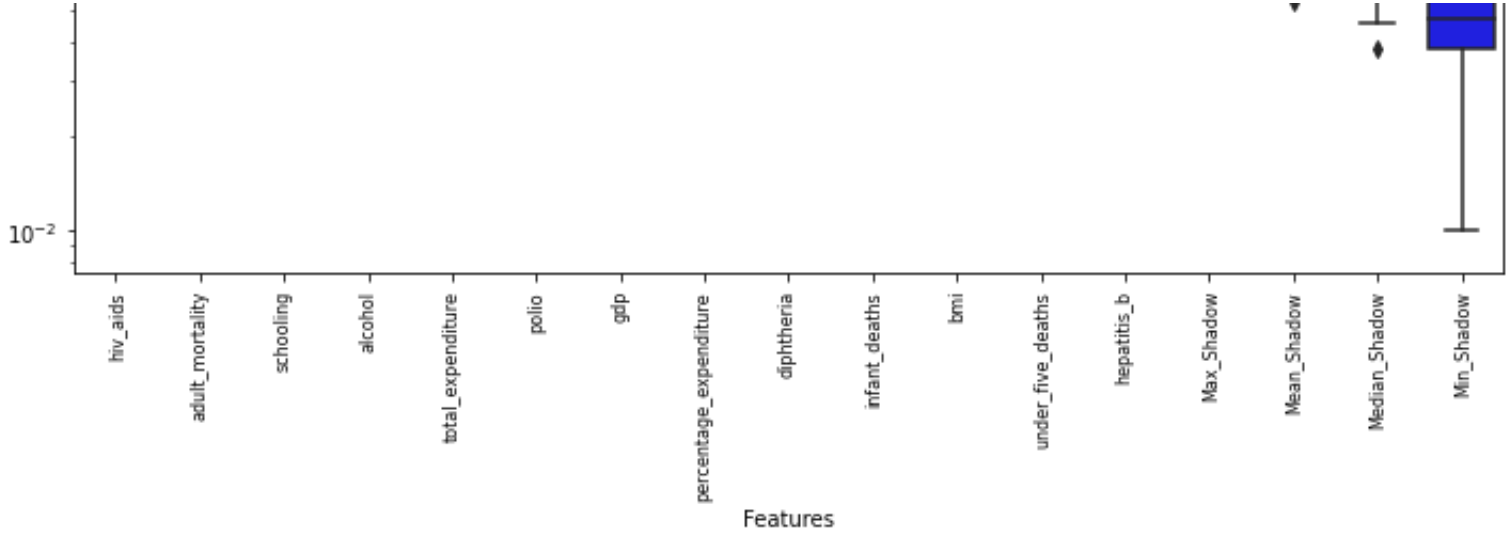
In [139]:

```
from BorutaShap import BorutaShap
x = boruta_data.iloc[:, 1:]
y = boruta_data['life_expectancy']

Feature_Selector = BorutaShap(importance_measure='shap', classification=False)
Feature_Selector.fit(X=x, y=y, n_trials= 500, random_state=0)
Feature_Selector.plot(which_features='all')
```

100% 500/500 [04:09<00:00, 2.01it/s]

```
9 attributes confirmed important: ['total_expenditure', 'diphtheria', 'adult_mortality', 'hiv_aids', 'u
nder_five_deaths', 'bmi', 'percentage_expenditure', 'schooling', 'alcohol']
4 attributes confirmed unimportant: ['polio', 'gdp', 'hepatitis_b', 'infant_deaths']
0 tentative attributes remains: []
```





On running the Boruta algorithm we get hiv/aids and adult mortality as the top two predictors.We get the top two predictors by studying the box-plots plotted on the Boruta algorithm.

Part b

Mallows CP

Mallows CP is another metric that we can use to tell us the best subset of features to include in our model. Using Mallows CP, we get schooling and adult mortality as the top two predictors.

In [140]:
```
pip install RegscorePy
```

```
Requirement already satisfied: RegscorePy in ./opt/anaconda3/lib/python3.9/site-packages (1.1)
Requirement already satisfied: pandas in ./opt/anaconda3/lib/python3.9/site-packages (from RegscorePy)
(1.4.2)
Requirement already satisfied: numpy in ./opt/anaconda3/lib/python3.9/site-packages (from RegscorePy) (
1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in ./opt/anaconda3/lib/python3.9/site-packages (f
rom pandas->RegscorePy) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas
->RegscorePy) (2021.3)
Requirement already satisfied: six>=1.5 in ./opt/anaconda3/lib/python3.9/site-packages (from python-dat
eutil>=2.8.1->pandas->RegscorePy) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [141]:
```
from RegscorePy import mallow
```

In [142]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ schooling', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[142]:
```
261.76377022345514
```

In [143]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ gdp', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[143]:
```
609.9217193557142
```

In [144]:
```
 = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ adult_m
ts = model.fit()
f['life_expectancy']
d=results.fittedvalues

b = smf.ols(formula='life_expectancy ~ diphtheria', data=df)
b_fit = mr_sub.fit()
=mr_sub_fit.fittedvalues

4 # number of parameters in orginal model (includes y-intercept)
 # number of parameters in the subset model (includes y-intercept)

w.mallow(y, y_pred,y_sub, k, p)
```

Out[144]:
```
623.75117539648
```

In [145]:
```
formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ adult_mortality+ t
fit()
ectancy']
ittedvalues

(formula='life_expectancy ~ hiv_aids', data=df)
sub.fit()
.fittedvalues

of parameters in orginal model (includes y-intercept)
f parameters in the subset model (includes y-intercept)

 y_pred,y_sub, k, p)
```

Out[145]: 426.69529261376056

In [146]:
```
odel = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ adu
esults = model.fit()
 = df['life_expectancy']
_pred=results.fittedvalues

r_sub = smf.ols(formula='life_expectancy ~ under_five_deaths', data=df)
r_sub_fit = mr_sub.fit()
_sub=mr_sub_fit.fittedvalues

 = 14 # number of parameters in orginal model (includes y-intercept)
 = 2 # number of parameters in the subset model (includes y-intercept)

allow.mallow(y, y_pred,y_sub, k, p)
```

Out[146]: 775.1130733802776

In [147]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ adult_mortality', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[147]: 298.69481167738405

In [148]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ total_expenditure', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[148]: 784.2583567244051

In [149]:
```
= smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ adult_m
s = model.fit()
['life_expectancy']
=results.fittedvalues

 = smf.ols(formula='life_expectancy ~ infant_deaths', data=df)
_fit = mr_sub.fit()
mr_sub_fit.fittedvalues

 # number of parameters in orginal model (includes y-intercept)
# number of parameters in the subset model (includes y-intercept)

.mallow(y, y_pred,y_sub, k, p)
```

Out[149]: 788.370216388592

In [150]:
```
l = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ adult_
lts = model.fit()
df['life_expectancy']
ed=results.fittedvalues

ub = smf.ols(formula='life_expectancy ~ hepatitis_b', data=df)
ub_fit = mr_sub.fit()
b=mr_sub_fit.fittedvalues

14 # number of parameters in orginal model (includes y-intercept)
2 # number of parameters in the subset model (includes y-intercept)

ow.mallow(y, y_pred,y_sub, k, p)
```

Out[150]: 790.3923743871304

In [151]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ polio', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[151]: 659.9068423460919

In [152]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ bmi', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[152]: 558.4253895110463

In [153]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ alcohol', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[153]:
```
570.5906276740914
```

In [154]:
```
model = smf.ols(formula='life_expectancy ~ schooling + gdp + diphtheria + hiv_aids+ under_five_deaths+ a
results = model.fit()
y = df['life_expectancy']
y_pred=results.fittedvalues

mr_sub = smf.ols(formula='life_expectancy ~ percentage_expenditure', data=df)
mr_sub_fit = mr_sub.fit()
y_sub=mr_sub_fit.fittedvalues

k = 14 # number of parameters in orginal model (includes y-intercept)
p = 2 # number of parameters in the subset model (includes y-intercept)

mallow.mallow(y, y_pred,y_sub, k, p)
```

Out[154]:
```
639.8949873910532
```

Question 3

OLS Model Selection

After running the Boruta Algorithm and Mallow's CP, our top 3 predictors are Adult Mortality, HIV/AIDS and Schooling. We explore several competing OLS models using these 3 predictors to test Goodness-of-Fit and arrive at the best performing model.

We use the following 4 types of OLS models for each of the 3 predictors and perform our evaluations:

1. Both of them are untransformed
2. Both X and Y variables are transformed
3. Only X is transformed
4. Only Y is transformed

In [155]:
```
# Boxcox Transformations for transformed predictor values
bc_life_expectancy,lambda_life_expectancy = scipy.stats.boxcox(df["life_expectancy"])
print(lambda_life_expectancy)
bc_hiv_aids,lambda_hiv_aids = scipy.stats.boxcox(df["hiv_aids"])
print(lambda_hiv_aids)
bc_adult_mortality,lambda_adult_mortality = scipy.stats.boxcox(df["adult_mortality"])
print(lambda_adult_mortality)
bc_schooling,lambda_schooling = scipy.stats.boxcox(df["schooling"])
print(lambda_schooling)
```

```
2.6415607532809915
-0.7676422714148633
0.47170957132662655
1.3267549713354299
```

In [156]:
```
# Adult Mortality untransformed
ols_1 = smf.ols('life_expectancy ~ adult_mortality', data = df)
res1 = ols_1.fit()
res1.summary()
```

Out[156]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.529
Model:	OLS	**Adj. R-squared:**	0.527
Method:	Least Squares	**F-statistic:**	203.6
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	1.91e-31
Time:	13:58:08	**Log-Likelihood:**	-583.15
No. Observations:	183	**AIC:**	1170.
Df Residuals:	181	**BIC:**	1177.
Df Model:	1		

Covariance Type:	nonrobust					
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	79.4513	0.740	107.406	0.000	77.992	80.911
adult_mortality	-0.0574	0.004	-14.270	0.000	-0.065	-0.049

Omnibus:	52.431	**Durbin-Watson:**	1.845
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	105.766
Skew:	-1.346	**Prob(JB):**	1.08e-23
Kurtosis:	5.574	**Cond. No.**	313.

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

In [157]:
```
# Adult Mortality: box cox transformation on both x and y
ols_bc = smf.ols('bc_life_expectancy ~ bc_adult_mortality', data = df)
res_bc = ols_bc.fit()
res_bc.summary()
```

Out[157]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.384
Model:	OLS	**Adj. R-squared:**	0.381
Method:	Least Squares	**F-statistic:**	112.8
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	8.67e-21
Time:	13:58:09	**Log-Likelihood:**	-1881.8

No. Observations:	183	**AIC:**	3768.
Df Residuals:	181	**BIC:**	3774.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.256e+04	1275.541	33.367	0.000	4e+04	4.51e+04
bc_adult_mortality	-666.3571	62.742	-10.621	0.000	-790.156	-542.558

Omnibus:	14.100	**Durbin-Watson:**	1.897
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	15.239
Skew:	-0.626	**Prob(JB):**	0.000491
Kurtosis:	3.658	**Cond. No.**	49.4

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

In [159]:
```
# Adult mortality: box cox on x only
ols_bcx = smf.ols('life_expectancy ~ bc_adult_mortality', data = df)
res_bcx = ols_bcx.fit()
res_bcx.summary()
```

Out[159]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.393
Model:	OLS	**Adj. R-squared:**	0.389

Method:	Least Squares	**F-statistic:**	117.1
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	2.29e-21
Time:	13:58:11	**Log-Likelihood:**	-606.47
No. Observations:	183	**AIC:**	1217.
Df Residuals:	181	**BIC:**	1223.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	82.7471	1.200	68.965	0.000	80.380	85.115
bc_adult_mortality	-0.6387	0.059	-10.822	0.000	-0.755	-0.522

Omnibus:	23.689	**Durbin-Watson:**	1.915
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	28.579
Skew:	-0.898	**Prob(JB):**	6.22e-07
Kurtosis:	3.725	**Cond. No.**	49.4

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

In [160]:

```
# Adult Mortality transformed box cox on y only
ols_bcy = smf.ols('bc_life_expectancy ~ adult_mortality', data = df)
res_bcy = ols_bcy.fit()
res_bcy.summary()
```

Out[160]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.502
Model:	OLS	**Adj. R-squared:**	0.499
Method:	Least Squares	**F-statistic:**	182.1
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	3.59e-29
Time:	13:58:13	**Log-Likelihood:**	-1862.4
No. Observations:	183	**AIC:**	3729.
Df Residuals:	181	**BIC:**	3735.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.899e+04	803.504	48.519	0.000	3.74e+04	4.06e+04
adult_mortality	-58.9171	4.366	-13.495	0.000	-67.531	-50.303

Omnibus:	32.693	**Durbin-Watson:**	1.824
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53.604
Skew:	-0.933	**Prob(JB):**	2.29e-12
Kurtosis:	4.885	**Cond. No.**	313.

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

In [161]:
```
# Schooling untransformed
ols_2 = smf.ols('life_expectancy ~ schooling', data = df)
res2 = ols_2.fit()
res2.summary()
```

Out[161]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.566
Model:	OLS	**Adj. R-squared:**	0.563
Method:	Least Squares	**F-statistic:**	235.9
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	1.27e-34
Time:	13:58:14	**Log-Likelihood:**	-575.79
No. Observations:	183	**AIC:**	1156.
Df Residuals:	181	**BIC:**	1162.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	41.7443	1.945	21.463	0.000	37.907	45.582
schooling	2.2909	0.149	15.358	0.000	1.997	2.585

Omnibus:	15.292	**Durbin-Watson:**	2.106
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	17.207

Skew:	-0.633	**Prob(JB):**	0.000183
Kurtosis:	3.807	**Cond. No.**	61.0

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

In [162]:
```
# Schooling transformed: box cox on both x and y
ols_bc2 = smf.ols('bc_life_expectancy ~ bc_schooling', data = df)
res_bc2 = ols_bc2.fit()
res_bc2.summary()
```

Out[162]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.590
Model:	OLS	**Adj. R-squared:**	0.588
Method:	Least Squares	**F-statistic:**	260.2
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	7.30e-37
Time:	13:58:15	**Log-Likelihood:**	-1844.6
No. Observations:	183	**AIC:**	3693.
Df Residuals:	181	**BIC:**	3700.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6758.7768	1516.250	4.458	0.000	3766.978	9750.576
bc_schooling	1089.8875	67.562	16.132	0.000	956.576	1223.199

Omnibus:	3.067	**Durbin-Watson:**	2.116
Prob(Omnibus):	0.216	**Jarque-Bera (JB):**	2.733
Skew:	-0.201	**Prob(JB):**	0.255
Kurtosis:	3.443	**Cond. No.**	79.5

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

In [163]:
```
#Schooling Transformed: box cox on x only
ols_bc2x = smf.ols('life_expectancy ~ bc_schooling', data = df)
res_bc2x = ols_bc2x.fit()
res_bc2x.summary()
```

Out[163]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.568
Model:	OLS	**Adj. R-squared:**	0.566
Method:	Least Squares	**F-statistic:**	238.4
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	7.39e-35
Time:	13:58:16	**Log-Likelihood:**	-575.25
No. Observations:	183	**AIC:**	1154.
Df Residuals:	181	**BIC:**	1161.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**

Intercept	49.0946	1.474	33.315	0.000	46.187	52.002
bc_schooling	1.0138	0.066	15.439	0.000	0.884	1.143

Omnibus:	14.158	**Durbin-Watson:**	2.103
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	15.467
Skew:	-0.616	**Prob(JB):**	0.000438
Kurtosis:	3.715	**Cond. No.**	79.5

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

In [164]:
```
#Schooling Transformed: box cox on y only
ols_bc2y = smf.ols('bc_life_expectancy ~ schooling', data = df)
res_bc2y = ols_bc2y.fit()
res_bc2y.summary()
```

Out[164]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.584
Model:	OLS	**Adj. R-squared:**	0.582
Method:	Least Squares	**F-statistic:**	254.1
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	2.59e-36
Time:	13:58:17	**Log-Likelihood:**	-1845.9
No. Observations:	183	**AIC:**	3696.
Df Residuals:	181	**BIC:**	3702.
Df Model:	1		

Covariance Type:	nonrobust					
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-1060.4933	2009.200	-0.528	0.598	-5024.961	2903.975
schooling	2456.3835	154.093	15.941	0.000	2152.333	2760.434

Omnibus:	3.398	**Durbin-Watson:**	2.112
Prob(Omnibus):	0.183	**Jarque-Bera (JB):**	3.093
Skew:	-0.216	**Prob(JB):**	0.213
Kurtosis:	3.467	**Cond. No.**	61.0

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

In [165]:
```
#HIV/AIDS Untransformed
ols_3 = smf.ols('life_expectancy ~ hiv_aids', data = df)
res3 = ols_3.fit()
res3.summary()
```

Out[165]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.403
Model:	OLS	**Adj. R-squared:**	0.400
Method:	Least Squares	**F-statistic:**	122.4
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	4.67e-22
Time:	13:58:18	**Log-Likelihood:**	-604.88

No. Observations:	183	**AIC:**	1214.
Df Residuals:	181	**BIC:**	1220.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	73.4875	0.543	135.452	0.000	72.417	74.558
hiv_aids	-2.7737	0.251	-11.062	0.000	-3.268	-2.279

Omnibus:	2.452	**Durbin-Watson:**	1.954
Prob(Omnibus):	0.293	**Jarque-Bera (JB):**	2.366
Skew:	-0.277	**Prob(JB):**	0.306
Kurtosis:	2.948	**Cond. No.**	2.51

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

In [166]:
```
#HIV/AIDS Transformation on both x and y
ols_3bc = smf.ols('bc_life_expectancy ~ bc_hiv_aids', data = df)
res3bc = ols_3bc.fit()
res3bc.summary()
```

Out[166]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.605
Model:	OLS	**Adj. R-squared:**	0.603

Method:	Least Squares	**F-statistic:**	277.8
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	2.12e-38
Time:	13:58:19	**Log-Likelihood:**	-1841.0
No. Observations:	183	**AIC:**	3686.
Df Residuals:	181	**BIC:**	3692.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.069e+04	709.912	29.143	0.000	1.93e+04	2.21e+04
bc_hiv_aids	-2337.1847	140.236	-16.666	0.000	-2613.893	-2060.477

Omnibus:	1.757	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.415	**Jarque-Bera (JB):**	1.361
Skew:	0.156	**Prob(JB):**	0.506
Kurtosis:	3.286	**Cond. No.**	8.76

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

In [167]:

```
#HIV/AIDS Transformation on x only
ols_3bcx = smf.ols('life_expectancy ~ bc_hiv_aids', data = df)
res3bcx = ols_3bcx.fit()
res3bcx.summary()
```

Out[167]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.641
Model:	OLS	**Adj. R-squared:**	0.639
Method:	Least Squares	**F-statistic:**	322.6
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	4.44e-42
Time:	13:58:20	**Log-Likelihood:**	-558.49
No. Observations:	183	**AIC:**	1121.
Df Residuals:	181	**BIC:**	1127.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	61.6289	0.642	95.995	0.000	60.362	62.896
bc_hiv_aids	-2.2779	0.127	-17.962	0.000	-2.528	-2.028

Omnibus:	1.239	**Durbin-Watson:**	2.023
Prob(Omnibus):	0.538	**Jarque-Bera (JB):**	0.877
Skew:	-0.121	**Prob(JB):**	0.645
Kurtosis:	3.237	**Cond. No.**	8.76

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

In [168]:
```
#HIV/AIDS Transformation on y only
ols_3bcy = smf.ols('bc_life_expectancy ~ hiv_aids', data = df)
res3bcy = ols_3bcy.fit()
res3bcy.summary()
```

Out[168]:

OLS Regression Results

Dep. Variable:	bc_life_expectancy	**R-squared:**	0.354
Model:	OLS	**Adj. R-squared:**	0.351
Method:	Least Squares	**F-statistic:**	99.27
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	6.43e-19
Time:	13:58:21	**Log-Likelihood:**	-1886.1
No. Observations:	183	**AIC:**	3776.
Df Residuals:	181	**BIC:**	3783.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.276e+04	595.701	54.996	0.000	3.16e+04	3.39e+04
hiv_aids	-2743.0577	275.318	-9.963	0.000	-3286.303	-2199.813

Omnibus:	1.275	**Durbin-Watson:**	1.929
Prob(Omnibus):	0.529	**Jarque-Bera (JB):**	1.286

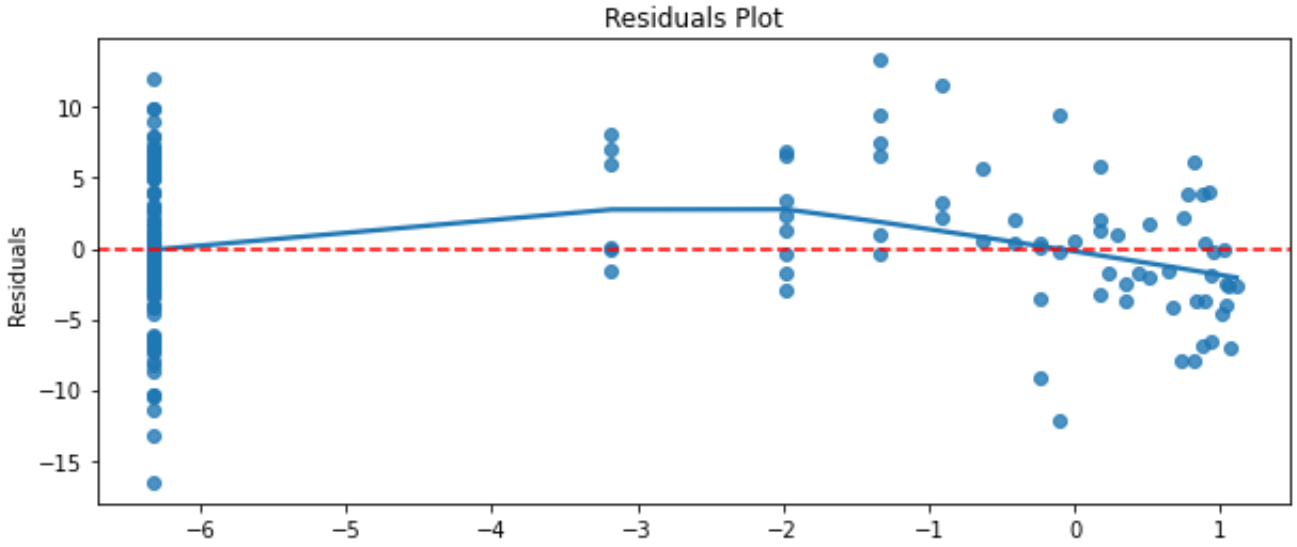
Skew:	0.114	**Prob(JB):**	0.526
Kurtosis:	2.659	**Cond. No.**	2.51

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

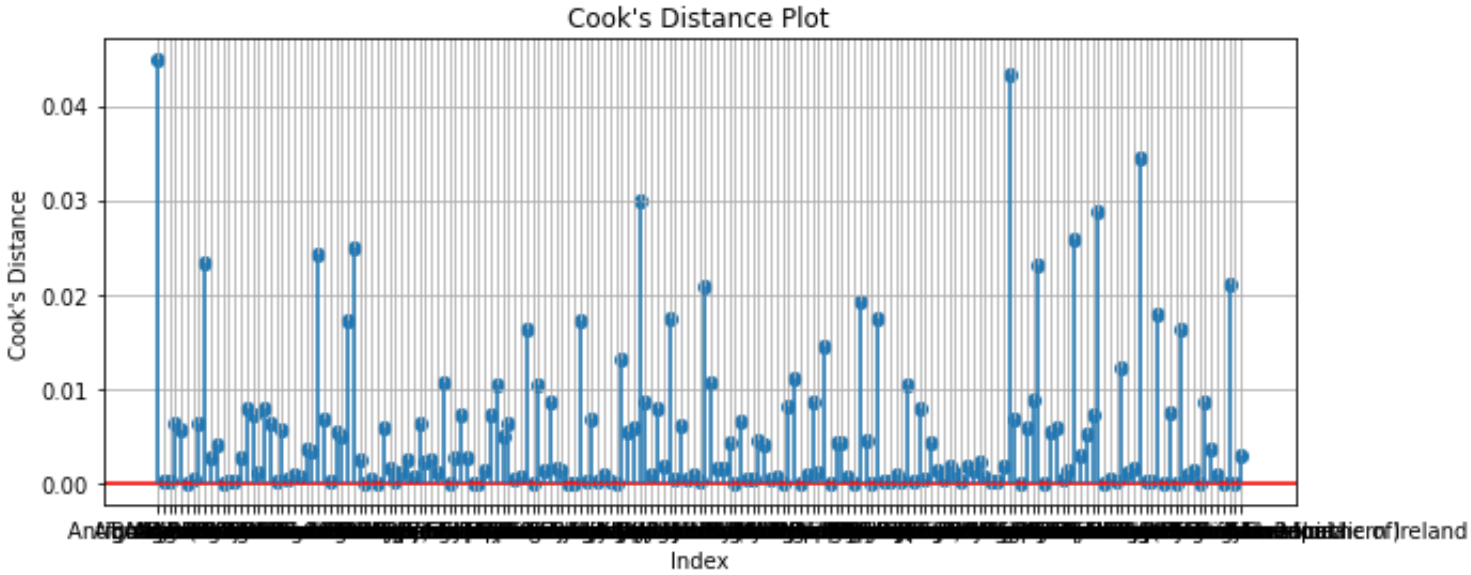
Upon running the OLS model for HIV/AIDS, with just the X variable transformed, we get the best fit. Some of the evaluations that helped us reach that conclusion are the following:

1. The R^2 is 0.641. This means that 64% of the regression is explained by the model.
2. Furthermore, B0 is 61.6289 and B1 is -2.2779. The negative B1 slope shows that there is a negative relationship between HIV/AIDS and Life expectancy. Using B0, we can expect life expectancy to be ~62 years when our X is O.
3. Further analysis tells us that the Jarque-Bera is 0.877. A low JB test tells us that the residuals are normally distributed.
4. The skewness is -0.121 and the kurtosis is 3.237. We know that a normal distribution has skewness 0 and kurtosis as 3. The values are pretty close to 0 and 3 respectively. Therefore, it is a good fit.
5. As a final test for our model, we see that the standard errors are also pretty low, having a value of 0.127.

In [169]:
```
#Residuals Plot for chosen model:
plt.figure(figsize = (10, 4))
sns.regplot(x = bc_hiv_aids, y = res3bcx.resid, lowess = True)
plt.axhline(0, linestyle = '--', color = "red")
plt.ylabel("Residuals")
plt.title("Residuals Plot")
plt.show()
```

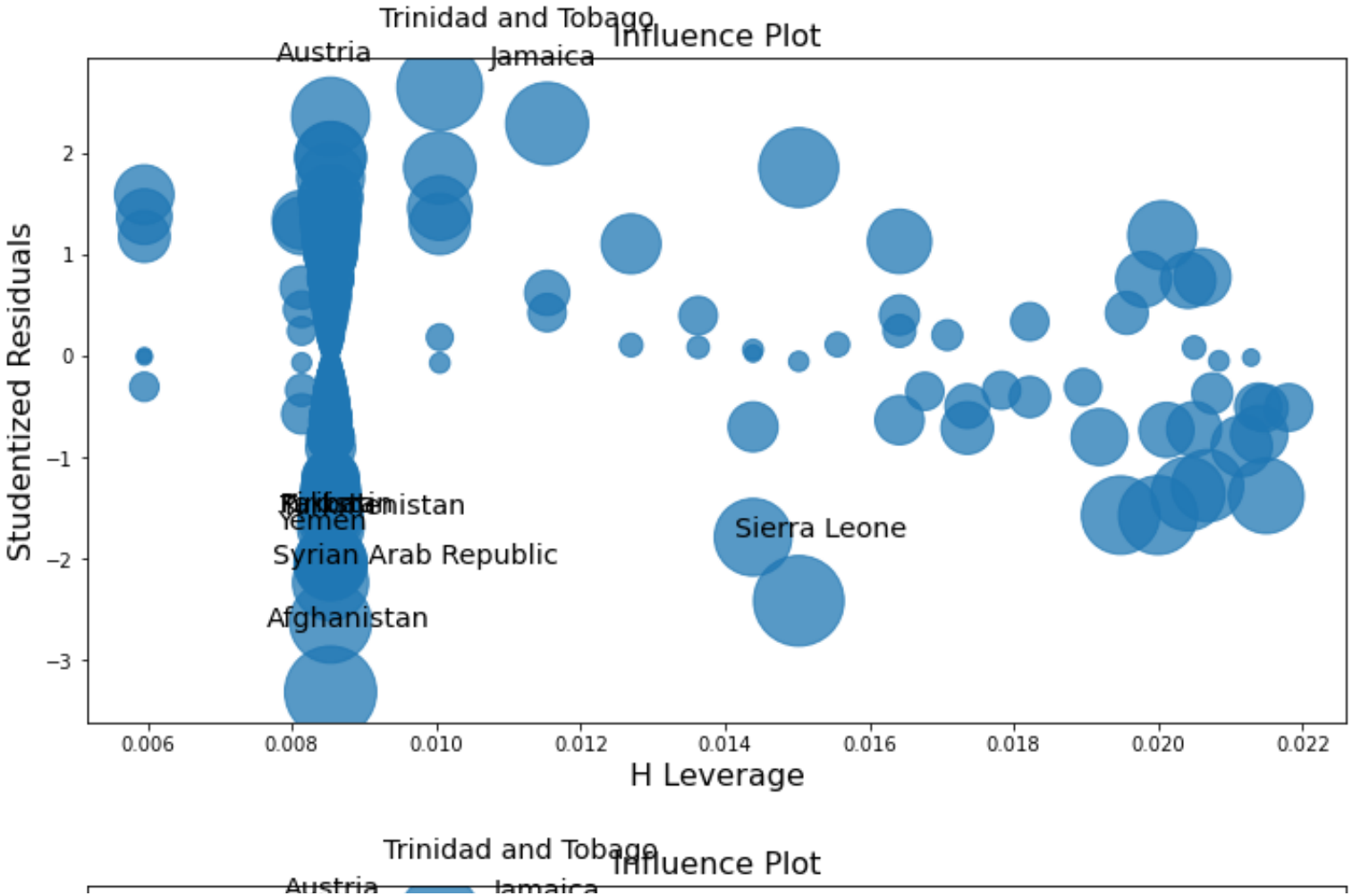


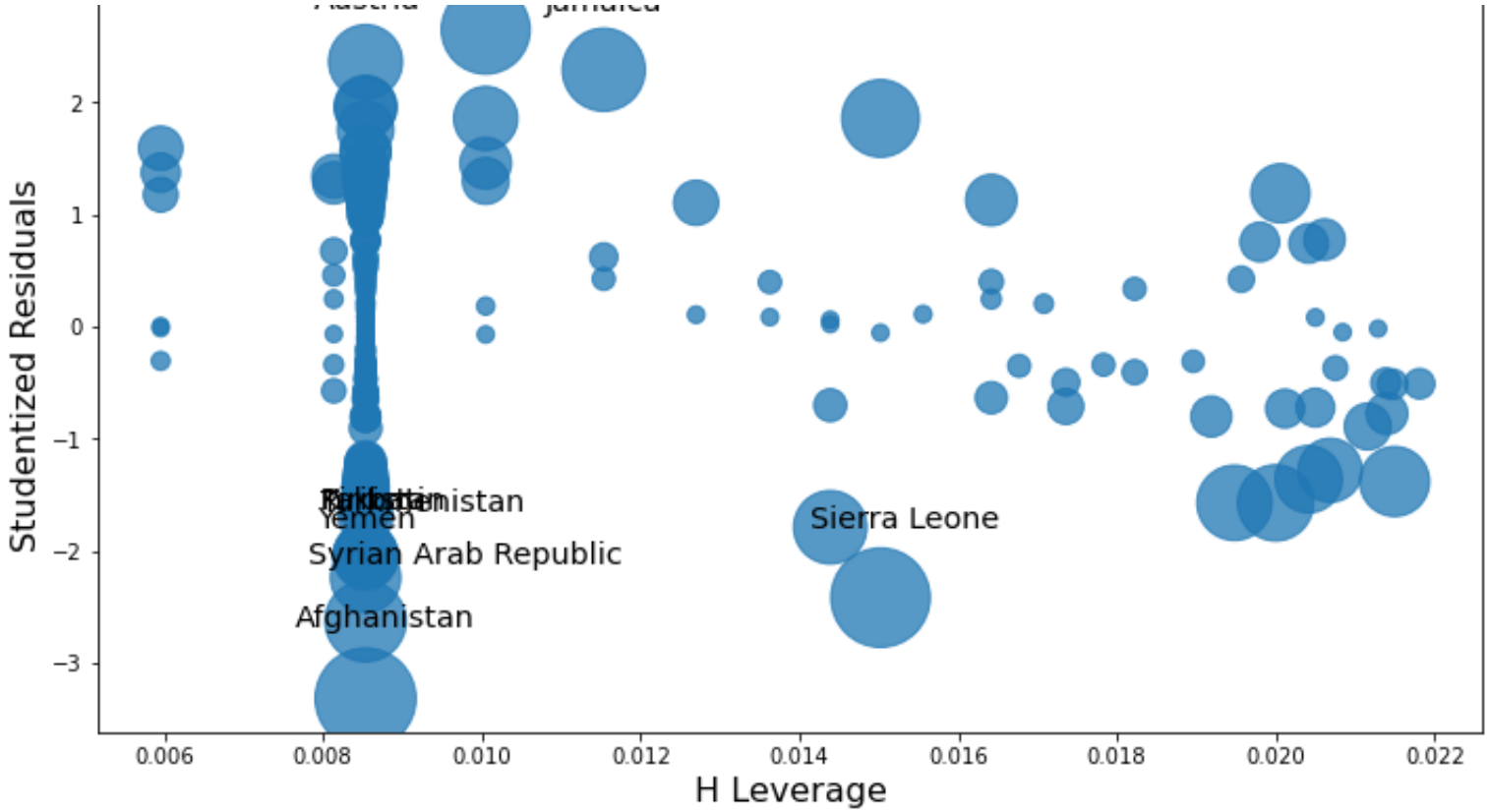
In [170]:
```
cooks_distance = res3bcx.get_influence().cooks_distance

plt.figure(figsize = (10, 4))
plt.scatter(df.index, cooks_distance[0])
plt.axhline(0, color = 'red')
plt.vlines(x = df.index, ymin = 0, ymax = cooks_distance[0])
plt.xlabel('Index')
plt.ylabel('Cook\'s Distance')
plt.title("Cook's Distance Plot")
plt.grid()
```



In [171]:

```
figd, ax = plt.subplots(figsize=(10,6))
figd = sms.graphics.influence_plot(res3bcx, ax = ax, criterion="DFFITS")
figd.tight_layout(pad=1.0)

fige, ax = plt.subplots(figsize=(10,6))
fige = sms.graphics.influence_plot(res3bcx, ax = ax, criterion="cooks")
fige.tight_layout(pad=1.0)
```





In [172]:
```
#Dropping outliers
df.drop(["Trinidad and Tobago", "Austria","Jamaica","Sierra Leone","Yemen","Afghanistan","Syrian Arab Re
```

In [173]:
```
df
```

Out[173]:

Country	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	polio	total_e
Albania	76.9	86	3	5.14	412.443356	99.0	55.8	1	99	
Algeria	75.1	113	21	0.66	555.926083	95.0	56.1	24	95	
Angola	56.0	358	72	8.24	256.122524	75.0	21.5	110	75	
Antigua and Barbuda	75.9	134	3	8.18	2156.229842	98.0	45.7	3	97	
Argentina	75.9	12	9	8.35	1133.558003	91.0	61.0	10	99	
...	...	...	...	...	...	...	...	...	...	
Vanuatu	71.4	138	3	0.01	427.988522	64.0	5.8	3	65	
Venezuela (Bolivarian Republic of)	73.7	161	9	6.70	196.915250	81.0	6.4	10	73	
Viet Nam	75.6	13	29	4.12	196.915250	97.0	15.3	36	97	
Zambia	59.2	349	29	2.59	196.915250	78.0	21.7	43	7	
Zimbabwe	56.6	429	26	6.09	92.602336	97.0	3.3	39	95	

176 rows × 14 columns

Now, using the Cook's distance plot, we dropped outliers. After dropping the outliers, we again calculated the above measures and checked our model. We got the following results:

1. The R^2 is 0.707. This tells us that after dropping the outliers, about 70% of the regression is explained by the model.
2. Skewness is 0.02
3. Kurtosis is 2.603
4. JB: 1.17
5. SE: .114

Dropping outliers therefore improves our model.

In [174]:
```
final_bc_hiv_aids,final_lambda_hiv_aids = scipy.stats.boxcox(df["hiv_aids"])
print(final_lambda_hiv_aids)
```

```
-0.768431442143658
```

In [175]:
```
ols_final = smf.ols('life_expectancy ~ final_bc_hiv_aids', data = df)
res_final = ols_final.fit()
res_final.summary()
```

Out[175]:

OLS Regression Results

Dep. Variable:	life_expectancy	**R-squared:**	0.707
Model:	OLS	**Adj. R-squared:**	0.705
Method:	Least Squares	**F-statistic:**	418.9
Date:	Wed, 19 Oct 2022	**Prob (F-statistic):**	3.43e-48
Time:	13:58:33	**Log-Likelihood:**	-515.53
No. Observations:	176	**AIC:**	1035.

Df Residuals:	174	**BIC:**	1041.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	61.4897	0.579	106.283	0.000	60.348	62.632
final_bc_hiv_aids	-2.3321	0.114	-20.467	0.000	-2.557	-2.107

Omnibus:	1.286	**Durbin-Watson:**	2.103
Prob(Omnibus):	0.526	**Jarque-Bera (JB):**	1.170
Skew:	0.020	**Prob(JB):**	0.557
Kurtosis:	2.603	**Cond. No.**	8.78

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

In [176]:
```
#Replacing HIV/AIDS column with its transformed values
df["hiv_aids"]= final_bc_hiv_aids
df
```

Out[176]:

Country	life_expectancy	adult_mortality	infant_deaths	alcohol	percentage_expenditure	hepatitis_b	bmi	under_five_deaths	polio	total_e
Albania	76.9	86	3	5.14	412.443356	99.0	55.8	1	99	
Algeria	75.1	113	21	0.66	555.926083	95.0	56.1	24	95	
Angola	56.0	358	72	8.24	256.122524	75.0	21.5	110	75	
Antigua and Barbuda	75.9	134	3	8.18	2156.229842	98.0	45.7	3	97	
Argentina	75.9	12	9	8.35	1133.558003	91.0	61.0	10	99	
...	...	...	...	...	...	...	...	...	...	
Vanuatu	71.4	138	3	0.01	427.988522	64.0	5.8	3	65	
Venezuela (Bolivarian Republic of)	73.7	161	9	6.70	196.915250	81.0	6.4	10	73	
Viet Nam	75.6	13	29	4.12	196.915250	97.0	15.3	36	97	
Zambia	59.2	349	29	2.59	196.915250	78.0	21.7	43	7	
Zimbabwe	56.6	429	26	6.09	92.602336	97.0	3.3	39	95	

176 rows × 14 columns

Bootstrapping

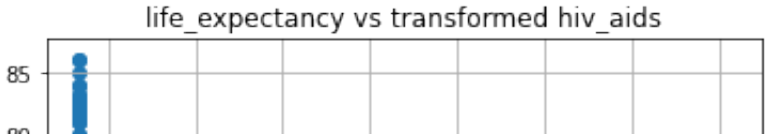
By bootstrapping our model, we run 100 resamples of our data to estimate the coefficient of our parameters. We show the bootstrap estimates on a histogram for the slope of the regression, the intercept and the Adjusted R^2 and compare it with the OLS estimates. We can see that the values of our OLS estimates and the bootstrap estimates are closely aligned.

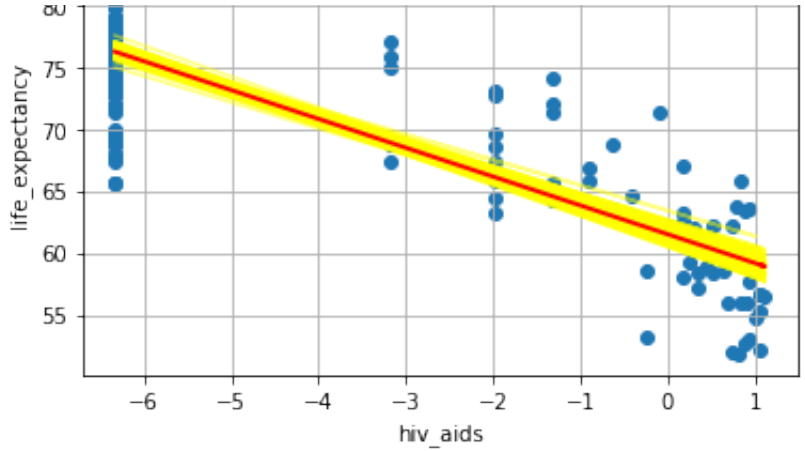
In [186]:
```
# bootstrapped slope CI

def reg_boot_b1(x,y):

    # bootstrap function gives us a 1d array, need 2d
    x = x.reshape((len(x),1))
    y = y.reshape((len(y),1))
    reg = LinearRegression().fit(x,y)

    # Pull out beta1
    return reg.coef_[0][0]

from scipy.stats import bootstrap
Y = df["life_expectancy"]
X = df["hiv_aids"]
res = bootstrap((X,Y), reg_boot_b1, confidence_level=0.95, vectorized = False, method = 'BCa',
                paired = True)
print('Bootstrapped Slope', res.confidence_interval)
```

```
Bootstrapped Slope ConfidenceInterval(low=-2.548733983151849, high=-2.109845340358303)
```

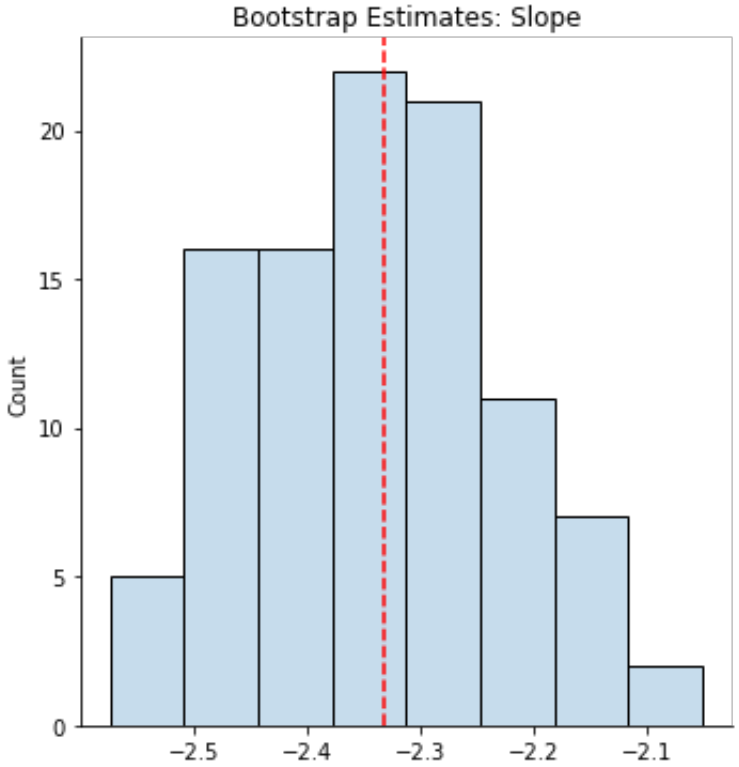
In [179]:
```
# bootstrapped scatterplot

# resample with replacement each row
boot_slopes = []
boot_interc = []
boot_adiR2 = []
```

```
n_boots = 100
n_points = df.shape[0]
plt.figure()
for _ in range(n_boots):
 # sample the rows, same size, with replacement
    sample_df = df.sample(n=n_points, replace=True)
 # fit a linear regression
    ols_model_temp = smf.ols(formula = 'life_expectancy ~ hiv_aids', data=sample_df)
    results_temp = ols_model_temp.fit()

 # append coefficients
    boot_interc.append(results_temp.params[0])
    boot_slopes.append(results_temp.params[1])
    boot_adjR2.append(results_temp.rsquared_adj)

 # plot a greyed out line
    y_pred_temp = ols_model_temp.fit().predict(sample_df['hiv_aids'])
    plt.plot(sample_df['hiv_aids'], y_pred_temp, color='yellow', alpha=0.5)

# Specify the Model
    ols_mod = smf.ols(formula='life_expectancy ~ hiv_aids', data=df)

# add data points
y_pred = ols_mod.fit().predict(df['hiv_aids'])
plt.scatter(df['hiv_aids'], df['life_expectancy'])
plt.plot(df['hiv_aids'], y_pred, linewidth=2,color = 'red')
plt.grid(True)
plt.xlabel('hiv_aids')
plt.ylabel('life_expectancy')
plt.title('life_expectancy vs transformed hiv_aids')
plt.show()
```



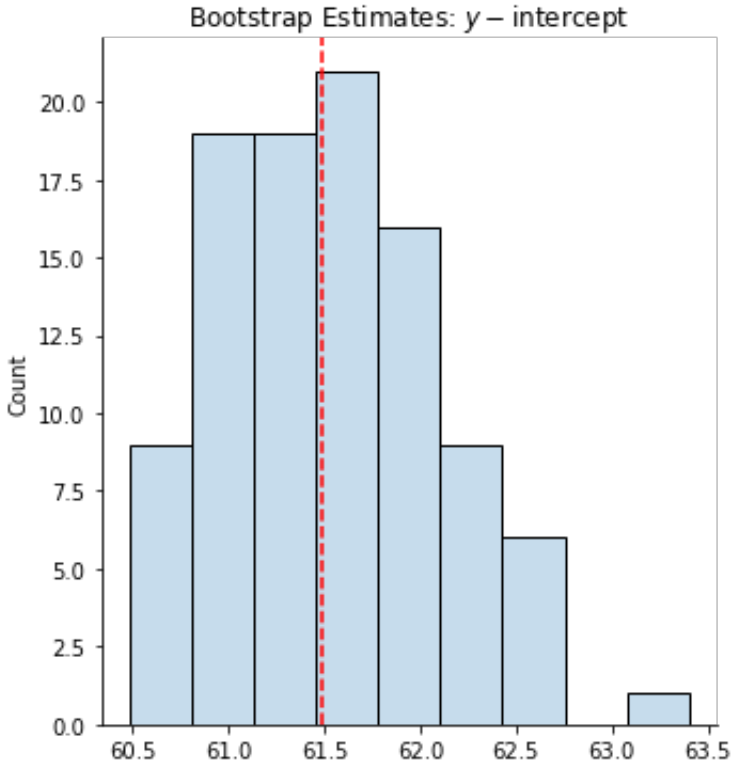

life_expectancy
hiv_aids
80
75
70
65
60
55
−6
−5
−4
−3
−2
−1
0
1

In [180]:
```
# bootstrapped histogram - slope

sns.displot(boot_slopes, alpha = 0.25)
plt.axvline(x=-2.3321,color='red', linestyle='--')
plt.title('Bootstrap Estimates: Slope')
plt.show()

# The vertical red line = LS estimate
```

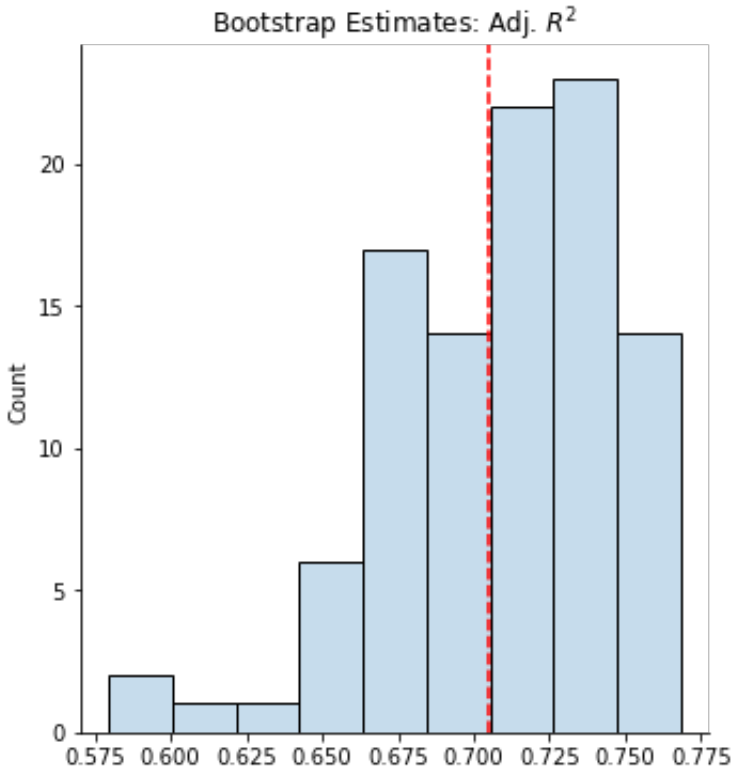


We are 95% confident that the value of ß1(slope) falls between -2.5487 and -2.1098. Our OLS estimate falls within that range.

In [181]:
```
# bootstrapped histogram - y intercept

sns.displot(boot_interc, alpha = 0.25)
plt.axvline(x=61.4897,color='red', linestyle='--')
plt.title('Bootstrap Estimates: $y-$intercept')
plt.show()
# The vertical red line = LS estimate
```



In [182]:
```
# bootstrapped histogram - adj r squared

sns.displot(boot_adjR2, alpha = 0.25)
plt.axvline(x= 0.705,color='red', linestyle='--')
plt.title('Bootstrap Estimates: Adj. $R^2$')
plt.show()

# The vertical red line = LS estimate
```



K-Fold Cross Validation

By using k-fold cross validation, we split our data into 5 subsets, trained our model on 4, and predicted the remaining fifth subset over all combinations of the other four

In [190]:
```
# k-fold cross validation for hiv_aids

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score

x = df[['hiv_aids']]
y = df[['life_expectancy']]
# Perform an OLS fit using all the data
regr = LinearRegression()
model = regr.fit(x,y)
regr.coef_
regr.intercept_

# Split the data into train  (70%)/test(30%) samples:
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

# Train the model:
regr = LinearRegression()
regr.fit(x_train, y_train)

# Make predictions based on the test sample
y_pred = regr.predict(x_test)

# Evaluate Performance
```

```
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Perform a 5-fold CV
# Use MSE as the scoring function (there are other options as shown here:
# https://scikit-learn.org/stable/modules/model_evaluation.html

regr = linear_model.LinearRegression()
scores = cross_val_score(regr, x, y, cv=5, scoring='neg_root_mean_squared_error')
print('5-Fold CV MSE Scores:', scores)
print('Mean of 5-Fold CV MSE Scores:',scores.mean())
```

```
MAE: 4.193608851759803
MSE: 25.911446473065308
RMSE: 5.090328719548995
5-Fold CV MSE Scores: [-4.36475916 -4.70678251 -4.74710629 -4.06157585 -5.01461986]
Mean of 5-Fold CV MSE Scores: -4.578968731825226
```

In [191]:
```
# k fold cross validation for adult_mortality

# k-fold cross validation for hiv_aids

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score

x = df[['adult_mortality']]
y = df[['life_expectancy']]
# Perform an OLS fit using all the data
regr = LinearRegression()
model = regr.fit(x,y)
```

```
regr.coef_
regr.intercept_

# Split the data into train  (70%)/test(30%) samples:
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

# Train the model:
regr = LinearRegression()
regr.fit(x_train, y_train)

# Make predictions based on the test sample
y_pred = regr.predict(x_test)

# Evaluate Performance

print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Perform a 5-fold CV
# Use MSE as the scoring function (there are other options as shown here:
# https://scikit-learn.org/stable/modules/model_evaluation.html

regr = linear_model.LinearRegression()
scores = cross_val_score(regr, x, y, cv=5, scoring='neg_root_mean_squared_error')
print('5-Fold CV MSE Scores:', scores)
print('Mean of 5-Fold CV MSE Scores:',scores.mean())
```

```
MAE: 4.451102223933389
MSE: 35.899321431481
RMSE: 5.9916042452319065
5-Fold CV MSE Scores: [-4.75407457 -5.03074922 -7.8508662  -6.1412637  -5.2188698 ]
Mean of 5-Fold CV MSE Scores: -5.799164700190926
```

The RMSE for hiv/aids is low at 5.9916. This value indicates a good out-of-sample performance for our model.

From our analysis, we conclude that the number of HIV/AIDS deaths per thousand people is the variable which shows the strongest relationship with Life Expectancy. This can be mainly understood by the R^2 values of the OLS Model and the results from our bootstrap estimates and k-fold cross validation. There is a negative relationship between HIV/AIDS deaths per thousand people and life expectancy.

For countries with high number of deaths caused due to HIV/AIDS, it puts an increasing pressure on people's spending as there is a higher burden of medical expenditure. One policy recommendation could be to perhaps put more focus on and channel more resources towards HIV/AIDS vaccination, proper medication and awareness can contribute towards increasing life expectancy.

References

1. Statistical Analysis on factors influencing Life Expectancy:(https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who (https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who)
2. Econ 430 lecture material: Professor Randall R. Rojas
3. Econ 442A lecture material: Professor Nathan Kunz

In [ ]: