

covid19 data analysis notebook

June 5, 2021

0.1 # Welcome to Covid19 Data Analysis Notebook

0.1.1 Let's Import the modules

```
[1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
print('Modules are imported.')
```

Modules are imported.

0.2 Task 2

0.2.1 Task 2.1: importing covid19 dataset

importing "Covid19_Confirmed_dataset.csv" from "./Dataset" folder.

```
[2]: corona_dataset_csv = pd.read_csv("Datasets/covid19_Confirmed_dataset.csv")
corona_dataset_csv.head(6)
```

```
[2]: Province/State      Country/Region      Lat      Long  1/22/20  1/23/20  \
0      NaN      Afghanistan  33.0000  65.0000      0      0
1      NaN      Albania    41.1533  20.1683      0      0
2      NaN      Algeria    28.0339  1.6596      0      0
3      NaN      Andorra    42.5063  1.5218      0      0
4      NaN      Angola     -11.2027  17.8739      0      0
5      NaN  Antigua and Barbuda  17.0608 -61.7964      0      0

      1/24/20  1/25/20  1/26/20  1/27/20  ...  4/21/20  4/22/20  4/23/20  \
0      0      0      0      0  ...    1092    1176    1279
1      0      0      0      0  ...     609     634     663
2      0      0      0      0  ...    2811    2910    3007
3      0      0      0      0  ...     717     723     723
4      0      0      0      0  ...      24      25      25
5      0      0      0      0  ...      23      24      24

      4/24/20  4/25/20  4/26/20  4/27/20  4/28/20  4/29/20  4/30/20
0    1351    1463    1531    1703    1828    1939    2171
1     678     712     726     736     750     766     773
```

2	3127	3256	3382	3517	3649	3848	4006
3	731	738	738	743	743	743	745
4	25	25	26	27	27	27	27
5	24	24	24	24	24	24	24

[6 rows x 104 columns]

Let's check the shape of the dataframe

```
[3]: corona_dataset_csv.shape
```

```
[3]: (266, 104)
```

0.2.2 Task 2.2: Delete the useless columns

```
[4]: corona_dataset_csv.drop(["Lat", "Long"], axis=1, inplace = True)
```

```
[5]: corona_dataset_csv.head(6)
```

```
[5]: Province/State      Country/Region  1/22/20  1/23/20  1/24/20  1/25/20  \
0      NaN      Afghanistan      0      0      0      0
1      NaN      Albania      0      0      0      0
2      NaN      Algeria      0      0      0      0
3      NaN      Andorra      0      0      0      0
4      NaN      Angola      0      0      0      0
5      NaN  Antigua and Barbuda      0      0      0      0
```

	1/26/20	1/27/20	1/28/20	1/29/20	...	4/21/20	4/22/20	4/23/20	\
0	0	0	0	0	...	1092	1176	1279	
1	0	0	0	0	...	609	634	663	
2	0	0	0	0	...	2811	2910	3007	
3	0	0	0	0	...	717	723	723	
4	0	0	0	0	...	24	25	25	
5	0	0	0	0	...	23	24	24	

	4/24/20	4/25/20	4/26/20	4/27/20	4/28/20	4/29/20	4/30/20
0	1351	1463	1531	1703	1828	1939	2171
1	678	712	726	736	750	766	773
2	3127	3256	3382	3517	3649	3848	4006
3	731	738	738	743	743	743	745
4	25	25	26	27	27	27	27
5	24	24	24	24	24	24	24

[6 rows x 102 columns]

0.2.3 Task 2.3: Aggregating the rows by the country

```
[6]: corona_dataset_aggregated = corona_dataset_csv.groupby("Country/Region").sum()
```

```
[7]: corona_dataset_aggregated.head()
```

```
[7]:
```

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	\
Country/Region								
Afghanistan	0	0	0	0	0	0	0	
Albania	0	0	0	0	0	0	0	
Algeria	0	0	0	0	0	0	0	
Andorra	0	0	0	0	0	0	0	
Angola	0	0	0	0	0	0	0	

	1/29/20	1/30/20	1/31/20	...	4/21/20	4/22/20	4/23/20	\
Country/Region				...				
Afghanistan	0	0	0	...	1092	1176	1279	
Albania	0	0	0	...	609	634	663	
Algeria	0	0	0	...	2811	2910	3007	
Andorra	0	0	0	...	717	723	723	
Angola	0	0	0	...	24	25	25	

	4/24/20	4/25/20	4/26/20	4/27/20	4/28/20	4/29/20	4/30/20
Country/Region							
Afghanistan	1351	1463	1531	1703	1828	1939	2171
Albania	678	712	726	736	750	766	773
Algeria	3127	3256	3382	3517	3649	3848	4006
Andorra	731	738	738	743	743	743	745
Angola	25	25	26	27	27	27	27

[5 rows x 100 columns]

```
[8]: corona_dataset_aggregated.shape
```

```
[8]: (187, 100)
```

0.2.4 Task 2.4: Visualizing data related to a country for example China

visualization always helps for better understanding of our data.

```
[9]: corona_dataset_aggregated.loc["China"]
```

```
[9]:
```

1/22/20	548
1/23/20	643
1/24/20	920
1/25/20	1406
1/26/20	2075

...

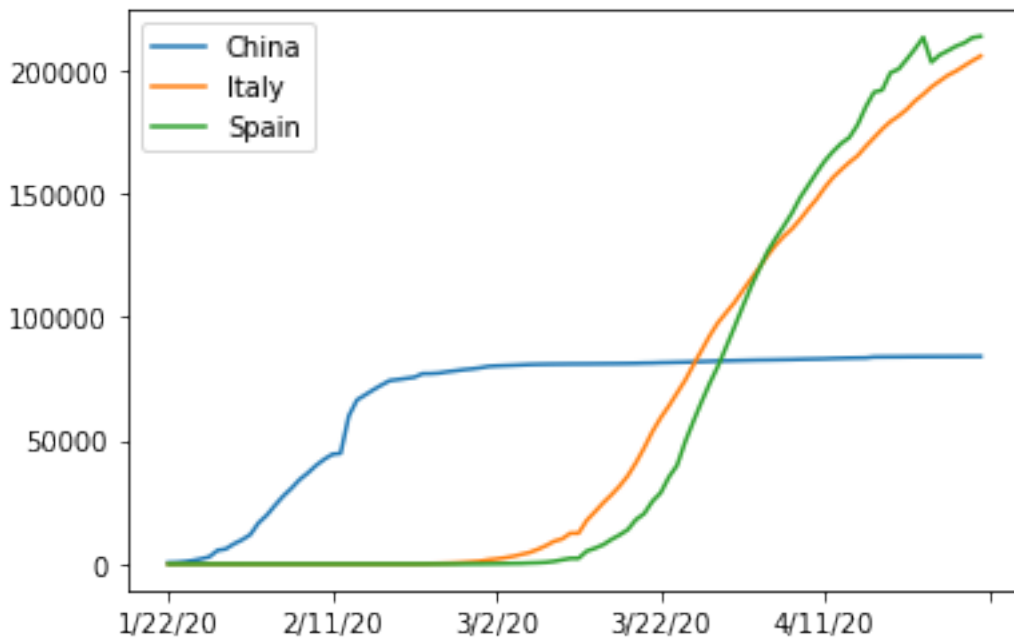
```
4/26/20    83912
4/27/20    83918
4/28/20    83940
4/29/20    83944
4/30/20    83956
Name: China, Length: 100, dtype: int64
```

0.2.5 Task3: Calculating a good measure

we need to find a good measure represented as a number, describing the spread of the virus in a country.

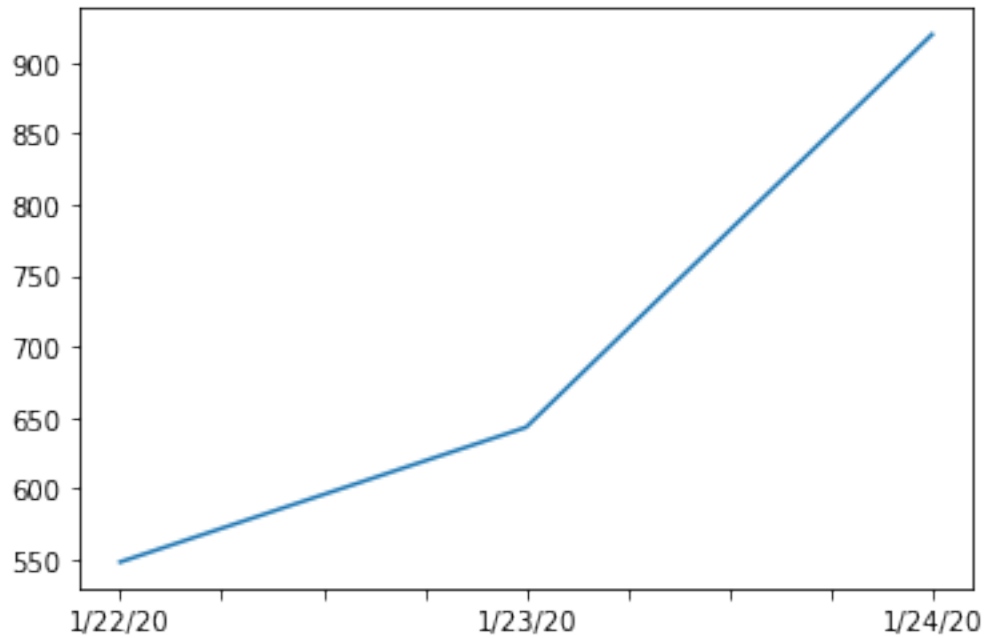
```
[10]: corona_dataset_aggregated.loc['China'].plot()
corona_dataset_aggregated.loc['Italy'].plot()
corona_dataset_aggregated.loc['Spain'].plot()
plt.legend()
```

```
[10]: <matplotlib.legend.Legend at 0x1d1e7f87b20>
```



```
[11]: corona_dataset_aggregated.loc["China"][:3].plot()
```

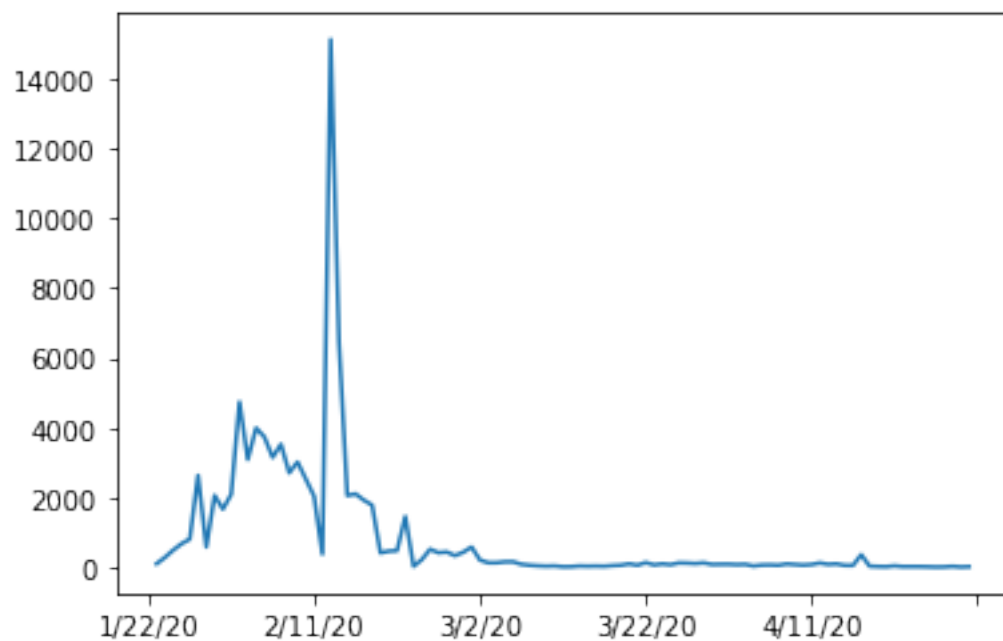
```
[11]: <AxesSubplot:>
```



0.2.6 task 3.1: caculating the first derivative of the curve

```
[12]: corona_dataset_aggregated.loc["China"].diff().plot()
```

```
[12]: <AxesSubplot:>
```



0.2.7 task 3.2: find maximum infection rate for China

```
[13]: corona_dataset_aggregated.loc["China"].diff().max()
```

```
[13]: 15136.0
```

```
[14]: corona_dataset_aggregated.loc["Italy"].diff().max()
```

```
[14]: 6557.0
```

```
[15]: corona_dataset_aggregated.loc["Spain"].diff().max()
```

```
[15]: 9630.0
```

0.2.8 Task 3.3: find maximum infection rate for all of the countries.

```
[16]: countries = list(corona_dataset_aggregated.index)
max_infection_rates = []
for c in countries :
    max_infection_rates.append(corona_dataset_aggregated.loc[c].diff().max())
corona_dataset_aggregated["max_infection_rate"] = max_infection_rates
```

```
[17]: corona_dataset_aggregated.head()
```

```
[17]:
```

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	\
Country/Region								
Afghanistan	0	0	0	0	0	0	0	
Albania	0	0	0	0	0	0	0	
Algeria	0	0	0	0	0	0	0	
Andorra	0	0	0	0	0	0	0	
Angola	0	0	0	0	0	0	0	

	1/29/20	1/30/20	1/31/20	...	4/22/20	4/23/20	4/24/20	\
Country/Region				...				
Afghanistan	0	0	0	...	1176	1279	1351	
Albania	0	0	0	...	634	663	678	
Algeria	0	0	0	...	2910	3007	3127	
Andorra	0	0	0	...	723	723	731	
Angola	0	0	0	...	25	25	25	

	4/25/20	4/26/20	4/27/20	4/28/20	4/29/20	4/30/20	\
Country/Region							
Afghanistan	1463	1531	1703	1828	1939	2171	
Albania	712	726	736	750	766	773	
Algeria	3256	3382	3517	3649	3848	4006	
Andorra	738	738	743	743	743	745	

Angola	25	26	27	27	27	27
--------	----	----	----	----	----	----

	max_infection_rate
Country/Region	
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

[5 rows x 101 columns]

0.2.9 Task 3.4: create a new dataframe with only needed column

```
[18]: corona_data = pd.DataFrame(corona_dataset_aggregated["max_infection_rate"])
```

```
[19]: corona_data.head()
```

```
[19]:
```

	max_infection_rate
Country/Region	
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

0.2.10 Task4:

- Importing the WorldHappinessReport.csv dataset
- selecting needed columns for our analysis
- join the datasets
- calculate the correlations as the result of our analysis

0.2.11 Task 4.1 : importing the dataset

```
[20]: happiness_report_csv = pd.read_csv("Datasets/worldwide_happiness_report.csv")
```

```
[21]: happiness_report_csv.head()
```

```
[21]:
```

	Overall rank	Country or region	Score	GDP per capita	Social support \
0	1	Finland	7.769	1.340	1.587
1	2	Denmark	7.600	1.383	1.573
2	3	Norway	7.554	1.488	1.582
3	4	Iceland	7.494	1.380	1.624
4	5	Netherlands	7.488	1.396	1.522

	Healthy life expectancy	Freedom to make life choices	Generosity \
0	0.986	0.596	0.153

1	0.996	0.592	0.252
2	1.028	0.603	0.271
3	1.026	0.591	0.354
4	0.999	0.557	0.322

Perceptions of corruption	
0	0.393
1	0.410
2	0.341
3	0.118
4	0.298

0.2.12 Task 4.2: let's drop the useless columns

```
[22]: useless_cols = ["Overall rank", "Score", "Generosity", "Perceptions of_
↳corruption"]
```

```
[23]: happiness_report_csv.drop(useless_cols, axis=1, inplace=True)
```

```
[24]: happiness_report_csv.head()
```

```
[24]: Country or region  GDP per capita  Social support  Healthy life expectancy \
0      Finland          1.340          1.587          0.986
1      Denmark          1.383          1.573          0.996
2      Norway           1.488          1.582          1.028
3      Iceland          1.380          1.624          1.026
4      Netherlands      1.396          1.522          0.999
```

Freedom to make life choices	
0	0.596
1	0.592
2	0.603
3	0.591
4	0.557

0.2.13 Task 4.3: changing the indices of the dataframe

```
[25]: happiness_report_csv.set_index("Country or region", inplace=True)
```

```
[26]: happiness_report_csv.head()
```

```
[26]: Country or region  GDP per capita  Social support  Healthy life expectancy \
Country or region
Finland          1.340          1.587          0.986
Denmark          1.383          1.573          0.996
Norway           1.488          1.582          1.028
Iceland          1.380          1.624          1.026
```


Netherlands	1.396	1.522	0.999
-------------	-------	-------	-------

Freedom to make life choices	
Country or region	
Finland	0.596
Denmark	0.592
Norway	0.603
Iceland	0.591
Netherlands	0.557

0.2.14 Task4.4: now let's join two dataset we have prepared

Corona Dataset :

```
[27]: corona_data.head()
```

```
[27]:
```

	max_infection_rate
Country/Region	
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

```
[28]: corona_data.shape
```

```
[28]: (187, 1)
```

wolrd happiness report Dataset :

```
[29]: happiness_report_csv.head()
```

```
[29]:
```

	GDP per capita	Social support	Healthy life expectancy \
Country or region			
Finland	1.340	1.587	0.986
Denmark	1.383	1.573	0.996
Norway	1.488	1.582	1.028
Iceland	1.380	1.624	1.026
Netherlands	1.396	1.522	0.999

Freedom to make life choices	
Country or region	
Finland	0.596
Denmark	0.592
Norway	0.603
Iceland	0.591
Netherlands	0.557

```
[30]: happiness_report_csv.shape
```

[30]: (156, 4)

```
[31]: data = corona_data.join(happiness_report_csv, how="inner")
data.head()
```

```
[31]:
```

	max_infection_rate	GDP per capita	Social support \
Afghanistan	232.0	0.350	0.517
Albania	34.0	0.947	0.848
Algeria	199.0	1.002	1.160
Argentina	291.0	1.092	1.432
Armenia	134.0	0.850	1.055

	Healthy life expectancy	Freedom to make life choices
Afghanistan	0.361	0.000
Albania	0.874	0.383
Algeria	0.785	0.086
Argentina	0.881	0.471
Armenia	0.815	0.283

0.2.15 Task 4.5: correlation matrix

```
[32]: data.corr()
```

```
[32]:
```

	max_infection_rate	GDP per capita \
max_infection_rate	1.000000	0.250118
GDP per capita	0.250118	1.000000
Social support	0.191958	0.759468
Healthy life expectancy	0.289263	0.863062
Freedom to make life choices	0.078196	0.394603

	Social support	Healthy life expectancy \
max_infection_rate	0.191958	0.289263
GDP per capita	0.759468	0.863062
Social support	1.000000	0.765286
Healthy life expectancy	0.765286	1.000000
Freedom to make life choices	0.456246	0.427892

	Freedom to make life choices
max_infection_rate	0.078196
GDP per capita	0.394603
Social support	0.456246
Healthy life expectancy	0.427892
Freedom to make life choices	1.000000

0.2.16 Task 5: Visualization of the results

our Analysis is not finished unless we visualize the results in terms figures and graphs so that everyone can understand what you get out of our analysis

```
[33]: data.head()
```

```
[33]:
```

	max_infection_rate	GDP per capita	Social support \
Afghanistan	232.0	0.350	0.517
Albania	34.0	0.947	0.848
Algeria	199.0	1.002	1.160
Argentina	291.0	1.092	1.432
Armenia	134.0	0.850	1.055

	Healthy life expectancy	Freedom to make life choices
Afghanistan	0.361	0.000
Albania	0.874	0.383
Algeria	0.785	0.086
Argentina	0.881	0.471
Armenia	0.815	0.283

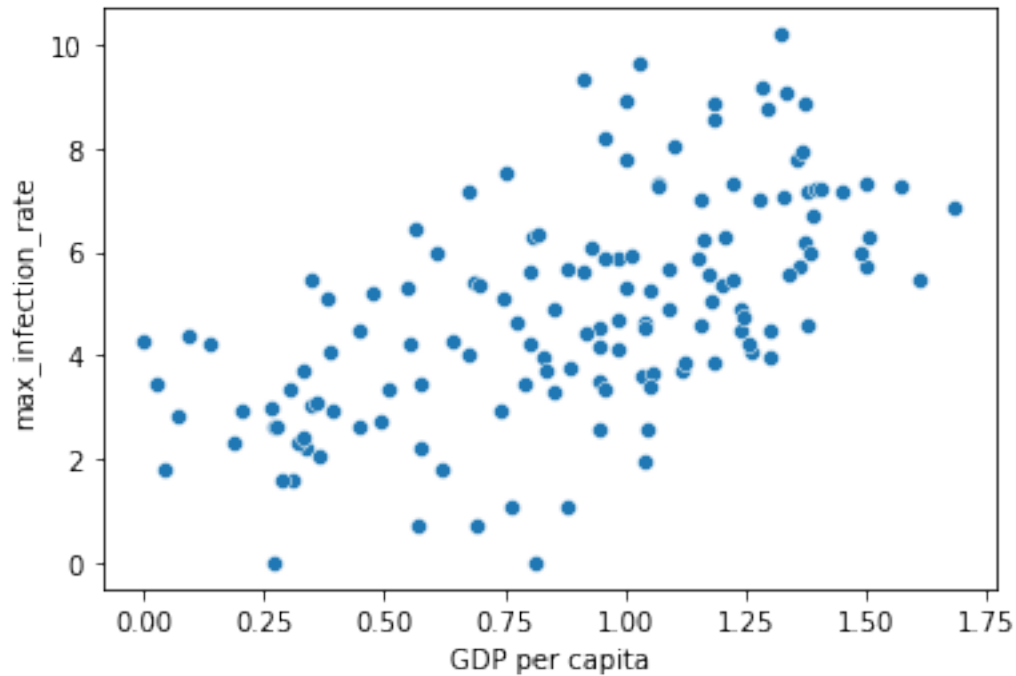
0.2.17 Task 5.1: Plotting GDP vs maximum Infection rate

```
[35]: x = data["GDP per capita"]
      y = data["max_infection_rate"]
      sns.scatterplot(x,np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[35]: <AxesSubplot:xlabel='GDP per capita', ylabel='max_infection_rate'>
```

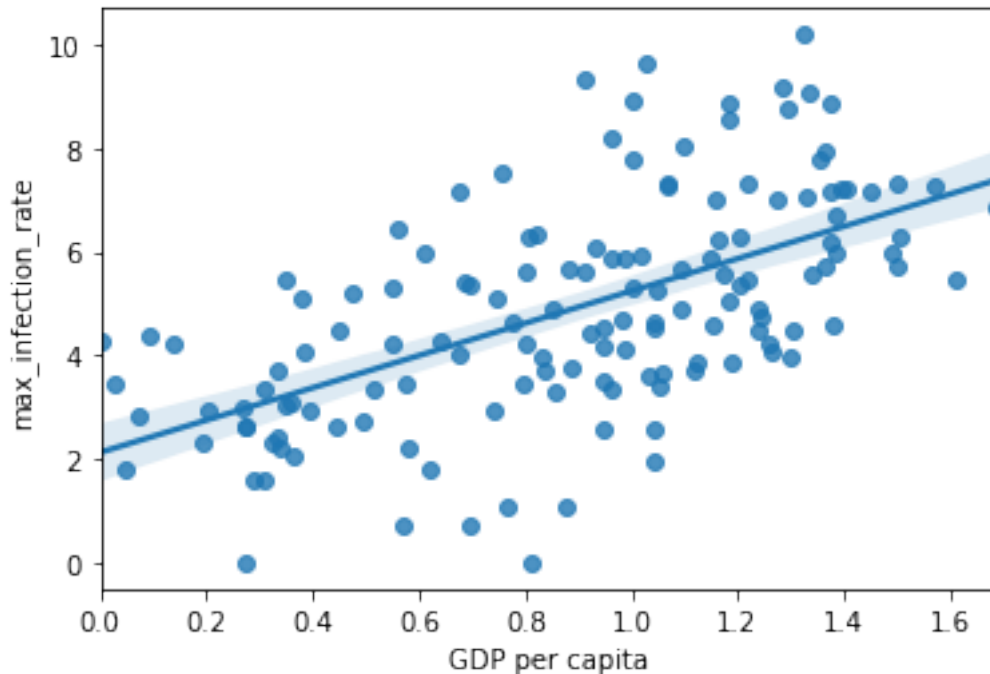


```
[36]: sns.regplot(x, np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[36]: <AxesSubplot:xlabel='GDP per capita', ylabel='max_infection_rate'>
```



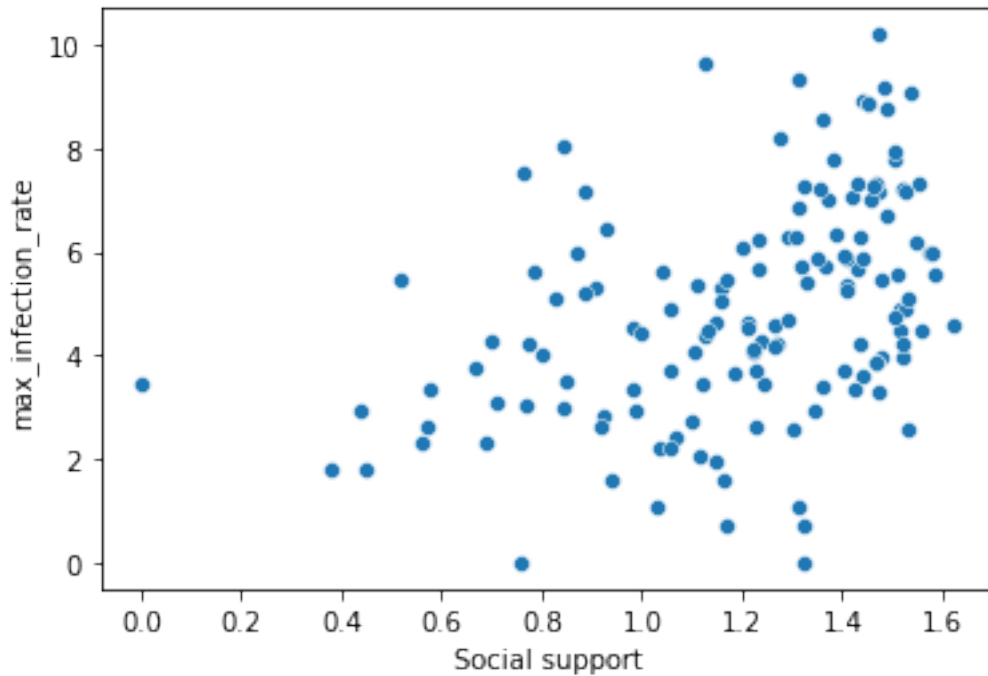
0.2.18 Task 5.2: Plotting Social support vs maximum Infection rate

```
[37]: x = data["Social support"]
      y = data["max_infection_rate"]
      sns.scatterplot(x,np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[37]: <AxesSubplot:xlabel='Social support', ylabel='max_infection_rate'>
```

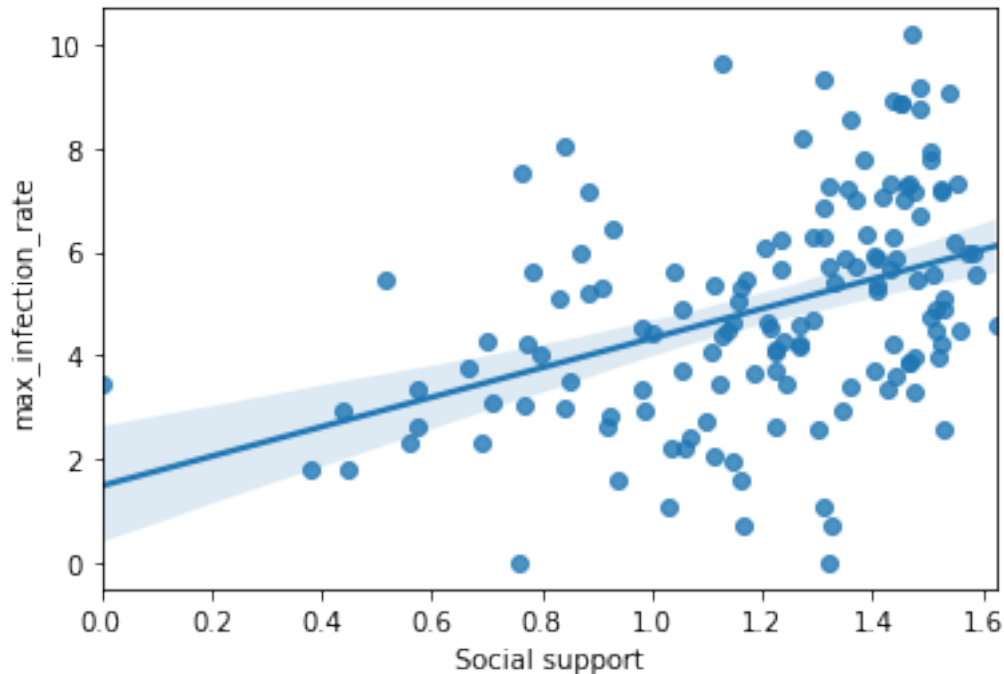


```
[38]: sns.regplot(x, np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[38]: <AxesSubplot:xlabel='Social support', ylabel='max_infection_rate'>
```



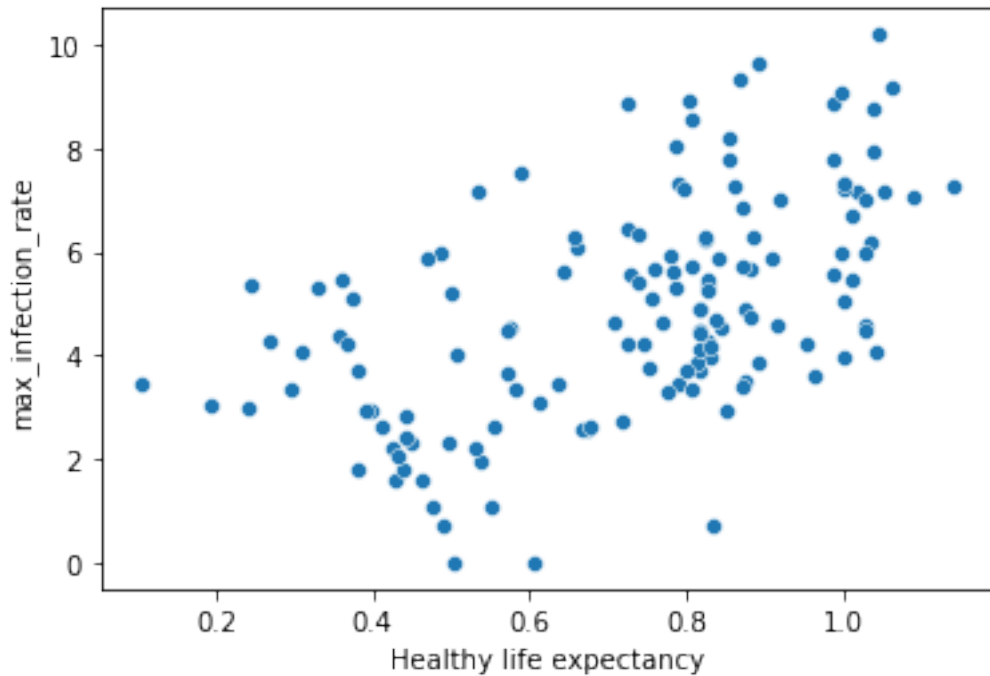
0.2.19 Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

```
[39]: x = data["Healthy life expectancy"]
      y = data["max_infection_rate"]
      sns.scatterplot(x,np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[39]: <AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rate'>
```

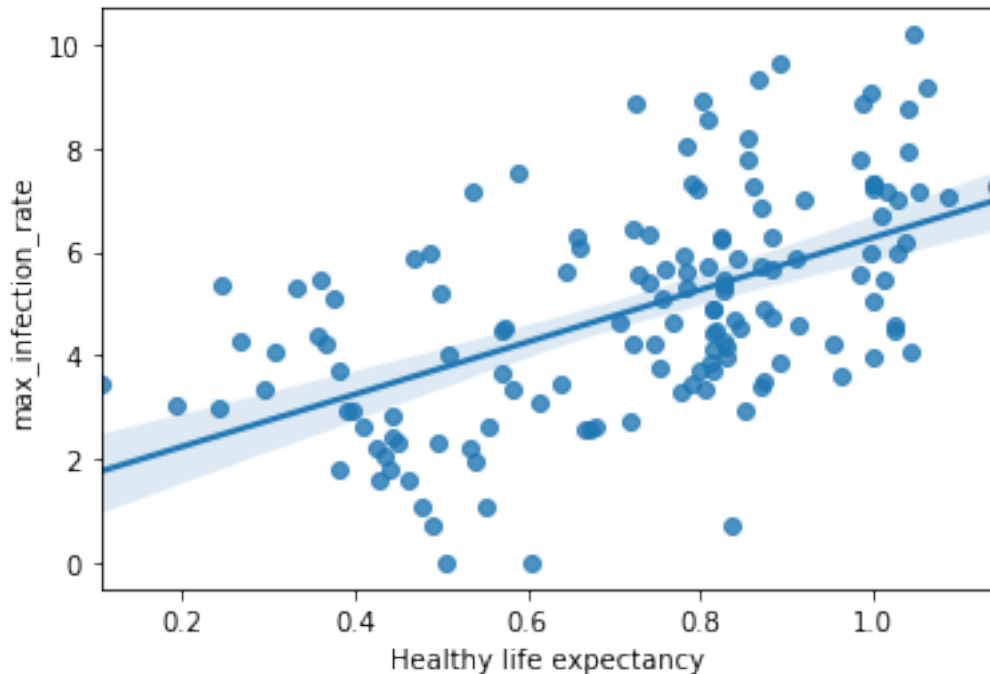


```
[40]: sns.regplot(x, np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[40]: <AxesSubplot:xlabel='Healthy life expectancy', ylabel='max_infection_rate'>
```

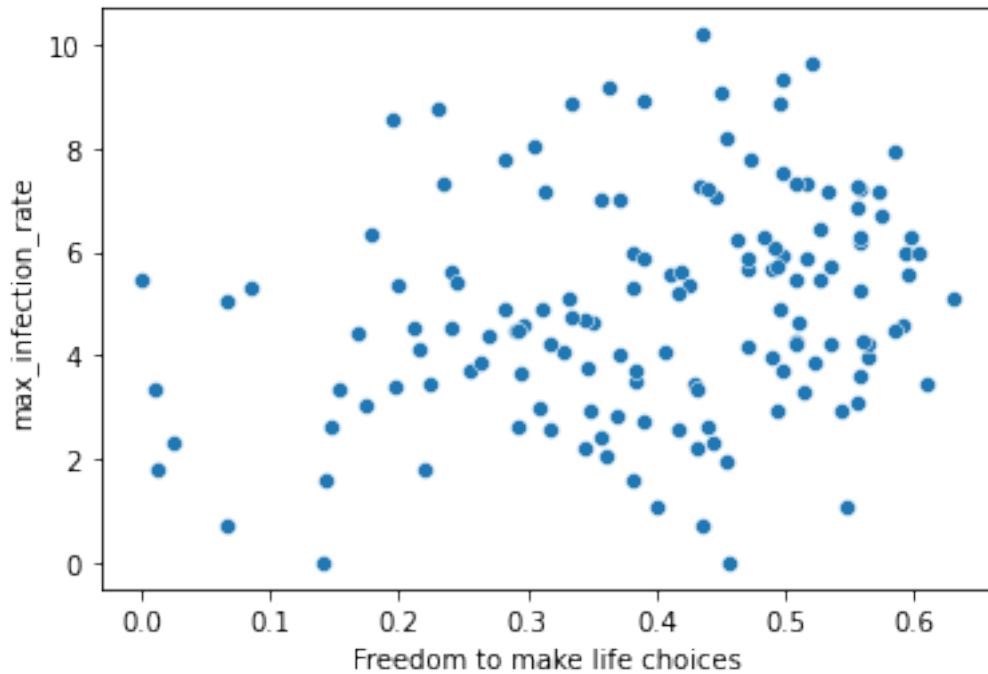
0.2.20 Task 5.4: Plotting Freedom to make life choices vs maximum Infection rate

```
[41]: x = data["Freedom to make life choices"]
      y = data["max_infection_rate"]
      sns.scatterplot(x,np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[41]: <AxesSubplot:xlabel='Freedom to make life choices', ylabel='max_infection_rate'>
```



```
[42]: sns.regplot(x, np.log(y))
```

d:\soft\python\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
[42]: <AxesSubplot:xlabel='Freedom to make life choices', ylabel='max_infection_rate'>
```

