Data cleaning

Never try to use a machine learning method on some new data before elementary data visualising and cleaning.

Content mostly inspired from https://learning.oreilly.com/library/view/data-cleaning-and/9781803241678/B17978_01_ePub.xhtml#_idParaDest-22 (https://learning.oreilly.com/library/view/data-cleaning-and/9781803241678/B17978_01_ePub.xhtml#_idParaDest-22)

Dataset coming from https://ourworldindata.org/covid-deaths (https://ourworldindata.org/covid-deaths)

Objectives:

- Basics of Numpy (to treat `arrays`) and Pandas (to treat `DataFrame`) and matplotlib.pyplot (to plot data)
- Subsetting data
- Generating summary statistics for continuous features
- Identifying extreme values and outliers
- Using histograms, boxplots, and violin plots to examine the distribution of continuous features

A - Import data and first contact with DataFrame 数据集的读取与基本操作

1. 读取数据：pd.read.csv()

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

covid19 = pd.read_csv("./owid-covid-data.csv")
covid19
```

Out[1]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers	handwashing_facilities	ho
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
429430	ZWE	Africa	Zimbabwe	2024-07-31	266386.0	0.0	0.0	5740.0	0.0	0.0	...	30.7	36.791	
429431	ZWE	Africa	Zimbabwe	2024-08-01	266386.0	0.0	0.0	5740.0	0.0	0.0	...	30.7	36.791	
429432	ZWE	Africa	Zimbabwe	2024-08-02	266386.0	0.0	0.0	5740.0	0.0	0.0	...	30.7	36.791	
429433	ZWE	Africa	Zimbabwe	2024-08-03	266386.0	0.0	0.0	5740.0	0.0	0.0	...	30.7	36.791	
429434	ZWE	Africa	Zimbabwe	2024-08-04	266386.0	0.0	0.0	5740.0	0.0	0.0	...	30.7	36.791	

429435 rows × 67 columns

2. 检索列名：.columns

In [2]:

```
covid19.columns
```

Out[2]:

```
Index(['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
       'new_cases_smoothed', 'total_deaths', 'new_deaths',
       'new_deaths_smoothed', 'total_cases_per_million',
       'new_cases_per_million', 'new_cases_smoothed_per_million',
       'total_deaths_per_million', 'new_deaths_per_million',
       'new_deaths_smoothed_per_million', 'reproduction_rate', 'icu_patients',
       'icu_patients_per_million', 'hosp_patients',
       'hosp_patients_per_million', 'weekly_icu_admissions',
       'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
       'weekly_hosp_admissions_per_million', 'total_tests', 'new_tests',
       'total_tests_per_thousand', 'new_tests_per_thousand',
       'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
       'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated', 'total_boosters',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred', 'stringency_index',
       'population_density', 'median_age', 'aged_65_older', 'aged_70_older',
       'gdp_per_capita', 'extreme_poverty', 'cardiovasc_death_rate',
       'diabetes_prevalence', 'female_smokers', 'male_smokers',
       'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index', 'population',
       'excess_mortality_cumulative_absolute', 'excess_mortality_cumulative',
       'excess_mortality', 'excess_mortality_cumulative_per_million'],
      dtype='object')
```

3. 获取每列的数据类型: .dtypes

In [3]:
```
covid19.dtypes
```

Out[3]:
```
iso_code                                    object
continent                                   object
location                                    object
date                                        object
total_cases                                float64
                                            ...   
population                                   int64
excess_mortality_cumulative_absolute       float64
excess_mortality_cumulative                float64
excess_mortality                           float64
excess_mortality_cumulative_per_million    float64
Length: 67, dtype: object
```

In [4]:
```
covid19.dtypes == 'float64'
```

Out[4]:
```
iso_code                                   False
continent                                  False
location                                   False
date                                       False
total_cases                                 True
                                           ...  
population                                 False
excess_mortality_cumulative_absolute        True
excess_mortality_cumulative                 True
excess_mortality                            True
excess_mortality_cumulative_per_million     True
Length: 67, dtype: bool
```

4. 索引数据: .loc / .iloc

Two modes to access a particular data:

- .loc[`index_name` , `column_name`] （使用name，也可同时使用index）
- .iloc[`index_number` , `column_number`] （使用index）

NB: the indexes can be replaced by lists

In [5]:
```
covid19.iloc[0,0]
```

Out[5]:
```
'AFG'
```

In [6]:
```
covid19.iloc[[0,1],[0,1,2] ]
```

Out[6]:

	iso_code	continent	location
0	AFG	Asia	Afghanistan
1	AFG	Asia	Afghanistan

In [7]:
```
covid19.loc[0,'iso_code'], covid19.loc[[0,1],['iso_code', 'continent', 'location']]
```

Out[7]:
```
('AFG',
   iso_code continent     location
 0      AFG      Asia  Afghanistan
 1      AFG      Asia  Afghanistan)
```

In [8]:
```
covid19.loc[:, "iso_code"]
```

Out[8]:
```
0         AFG
1         AFG
2         AFG
3         AFG
4         AFG
         ... 
429430    ZWE
429431    ZWE
429432    ZWE
429433    ZWE
429434    ZWE
Name: iso_code, Length: 429435, dtype: object
```

5. 作图：plt.plot()

Let us look at the data from Afghanistan
首先筛选出 iso_code 为 AFG 的信息

In [9]:
```
afg = covid19[covid19['iso_code'] == 'AFG']
afg
```

Out[9]:

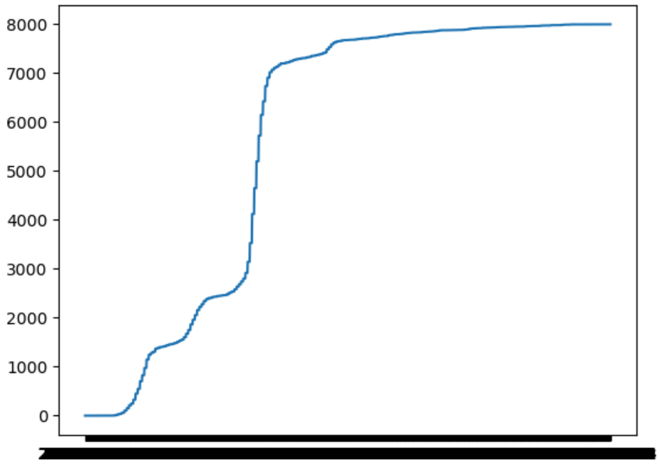
	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers	handwashing_facilities	hosp
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1669	AFG	Asia	Afghanistan	2024-07-31	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1670	AFG	Asia	Afghanistan	2024-08-01	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1671	AFG	Asia	Afghanistan	2024-08-02	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1672	AFG	Asia	Afghanistan	2024-08-03	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1673	AFG	Asia	Afghanistan	2024-08-04	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	

1674 rows × 67 columns

我们希望 plot 总死亡人数关于时间的折线图，如果直接用以下两种方法作图，x轴的label会出问题

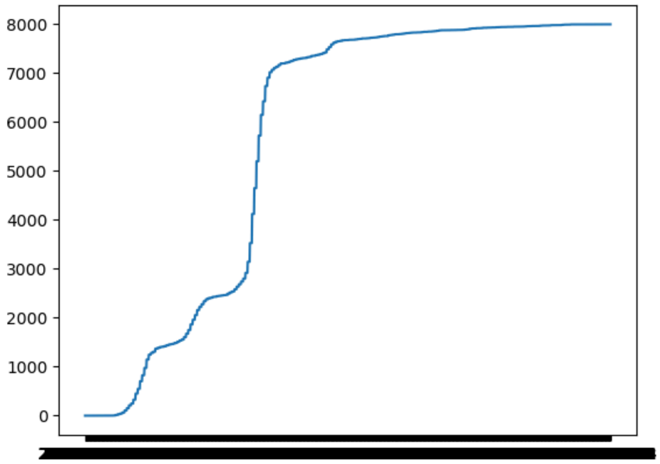
In [10]:
```
# 方法一：先把index改成date，再plot(total_deaths)
afg.index = afg.date
plt.plot(afg.total_deaths)
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x11f6cc130>]
```



In [11]:
```
# 方法二：plot(date, total_deaths)
plt.plot(afg.date, afg.total_deaths)
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x11ef6b400>]
```



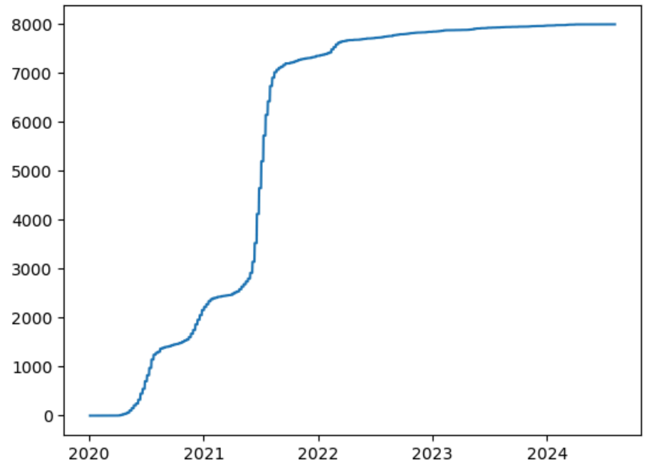
正确的做法是先将date数据的datatype转换为datetime，再进行plot操作

In [12]:

```
afg.index = pd.to_datetime(afg.date) # 把 date 这一列的 datatype 从 string 转换成 datetime，再设置成 index
plt.plot(afg.total_deaths)
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x169df66e0>]
```

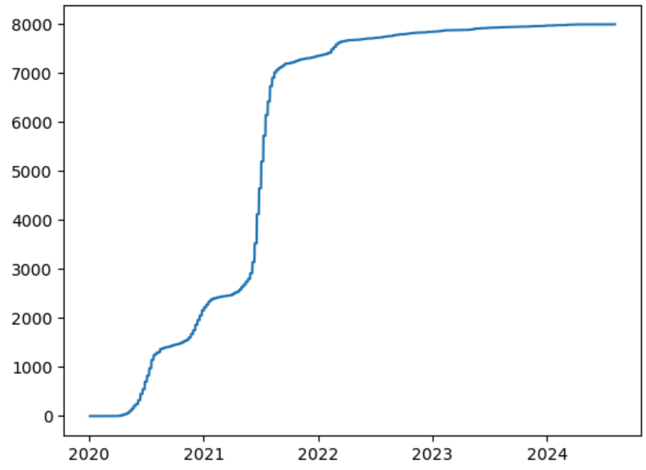


In [13]:

```
plt.plot(pd.to_datetime(afg.date), afg.total_deaths)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x1757fe410>]
```



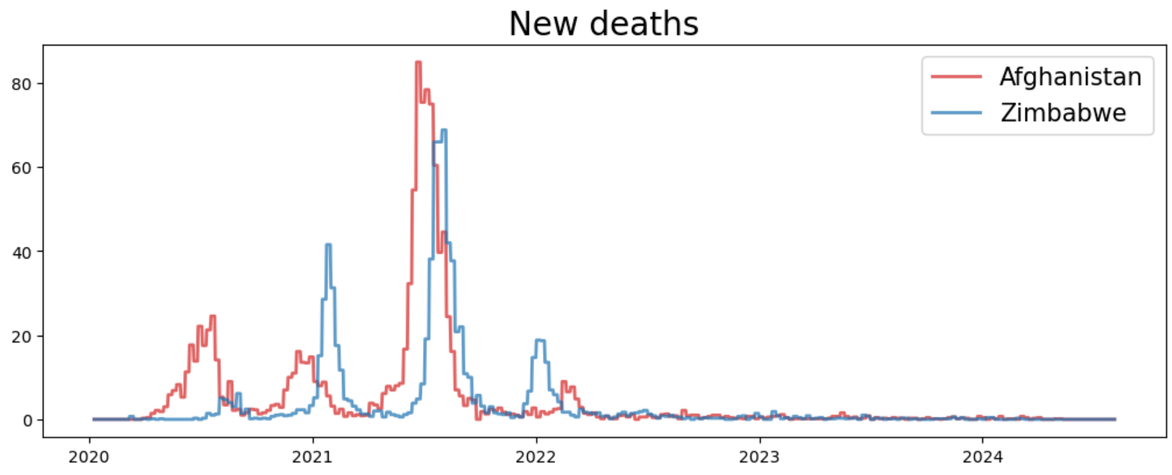
More beautiful graphs are available with `subplot` :

In [14]:

```
axesNb = 1
fig, axs = plt.subplots(axesNb, 1, constrained_layout=True, figsize= (10, 4*axesNb))
axs.set_title('New deaths', fontsize = 20)
afg = covid19[covid19['iso_code'] == 'AFG']
zwe = covid19[covid19['iso_code'] == 'ZWE']
afg.index = pd.to_datetime(afg.date)
zwe.index = pd.to_datetime(zwe.date)
axs.plot(afg.new_deaths_smoothed, 'tab:red', linewidth=2, alpha = 0.7, label = 'Afghanistan')
axs.plot(zwe.new_deaths_smoothed, 'tab:blue', linewidth=2, alpha = 0.7, label = 'Zimbabwe')
axs.legend(loc = 'upper right', fontsize = 15)
```

Out[14]:

```
<matplotlib.legend.Legend at 0x1757e3e80>
```



In [15]:
```
covid19[covid19['iso_code'] == 'AFG']
```

Out[15]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers	handwashing_facilities	hosp
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	NaN	0.0	0.0	NaN	...	NaN	37.746	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1669	AFG	Asia	Afghanistan	2024-07-31	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1670	AFG	Asia	Afghanistan	2024-08-01	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1671	AFG	Asia	Afghanistan	2024-08-02	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1672	AFG	Asia	Afghanistan	2024-08-03	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	
1673	AFG	Asia	Afghanistan	2024-08-04	235214.0	0.0	0.0	7998.0	0.0	0.0	...	NaN	37.746	

1674 rows × 67 columns

6. 对含NaN数据的处理

afg.new_deaths_smoothed

6.1 Pandas

`Pandas` deals with Nans more smoothly than `Numpy` .

使用 .max(), .mean(), sum() 时， `Pandas` 会自动忽略 NaN

In [16]:
```
afg.new_deaths_smoothed.max(),afg.new_deaths_smoothed.mean(),afg.new_deaths_smoothed.sum()
```

Out[16]:
```
(85.0, 4.792118633912522, 7998.045999999999)
```

6.2 Numpy

使用 .max(), .mean(), sum() 时， `Numpy` 不会自动忽略 NaN

In [17]:
```
afg_arr = afg.new_deaths_smoothed.to_numpy() # 转换为 nparray
afg_arr
```

Out[17]:
```
array([nan, nan, nan, ...,  0.,  0.,  0.])
```

In [18]:
```
afg_arr.max(),afg_arr.mean(),afg_arr.sum()
```

Out[18]:
```
(nan, nan, nan)
```

B - Create a new DataFrame with important and usable information 数据清洗

We want to do a country-wise statistical study of covid, do not need anymore the history of death.

先查看每行数据的location

In [19]:
```
covid19.location
```

Out[19]:
```
0         Afghanistan
1         Afghanistan
2         Afghanistan
3         Afghanistan
4         Afghanistan
             ...
429430       Zimbabwe
429431       Zimbabwe
429432       Zimbabwe
429433       Zimbabwe
429434       Zimbabwe
Name: location, Length: 429435, dtype: object
```

1. 统计各个value的数量：value_counts()

In [20]:
```
prd_len = covid19.location.value_counts()
prd_len
```

Out[20]:
```
location
High-income countries           3026
European Union (27)             3024
Upper-middle-income countries   3013
Lower-middle-income countries   2983
Low-income countries            2724
                                ...
Scotland                        1305
Wales                           1198
Macao                            795
Northern Cyprus                  691
Western Sahara                     1
Name: count, Length: 255, dtype: int64
```

Remarks:

1. `location` can be a group of countries
2. survey periods not the same for every location! --> check that data is consistent between locations

2. 分组操作：group_by()

Can use `group_by` do do internal operation in the dataframe

In [21]:
```python
cumm_col = ['total_cases','total_deaths']
loc_col = ['new_cases','new_deaths']
covid19[['iso_code'] + loc_col] # append list 的快捷方法
```

Out[21]:

	iso_code	new_cases	new_deaths
0	AFG	0.0	0.0
1	AFG	0.0	0.0
2	AFG	0.0	0.0
3	AFG	0.0	0.0
4	AFG	0.0	0.0
...	...	...	...
429430	ZWE	0.0	0.0
429431	ZWE	0.0	0.0
429432	ZWE	0.0	0.0
429433	ZWE	0.0	0.0
429434	ZWE	0.0	0.0

429435 rows × 3 columns

利用 daily data 和 community data 验证数据集的准确性

In [22]:
```python
# 将 daily data 相加，得到总 cases 数和总 deaths 数
covid19[loc_col].groupby(covid19.iso_code).sum()
```

Out[22]:

	new_cases	new_deaths
iso_code		
ABW	44224.0	292.0
AFG	235214.0	7998.0
AGO	107481.0	1937.0
AIA	3904.0	12.0
ALB	335047.0	3605.0
...	...	...
WSM	17077.0	31.0
YEM	11945.0	2159.0
ZAF	4072765.0	102595.0
ZMB	349842.0	4077.0
ZWE	266387.0	5740.0

255 rows × 2 columns

In [23]:
```python
# 将 community data 取最大值，得到的即是最新的，即总 cases 数和总 deaths 数
covid19[cumm_col].groupby(covid19.iso_code).max()
```

Out[23]:

	total_cases	total_deaths
iso_code		
ABW	44224.0	292.0
AFG	235214.0	7998.0
AGO	107481.0	1937.0
AIA	3904.0	12.0
ALB	335047.0	3605.0
...	...	...
WSM	17057.0	31.0
YEM	11945.0	2159.0
ZAF	4072765.0	102595.0
ZMB	349842.0	4077.0
ZWE	266386.0	5740.0

255 rows × 2 columns

3. 合并dataframes：concat()

axis = 0: 竖着合并
axis = 1: 并排合并

```
In [24]: # 合并上述两个 df，进行比较，判断数据集的准确性
         compare_df = pd.concat([covid19[loc_col].groupby(covid19.iso_code).sum(), covid19[cumm_col].groupby(covid19.iso_code).max()], axis=1)
         compare_df
```

Out[24]:

	new_cases	new_deaths	total_cases	total_deaths
iso_code				
ABW	44224.0	292.0	44224.0	292.0
AFG	235214.0	7998.0	235214.0	7998.0
AGO	107481.0	1937.0	107481.0	1937.0
AIA	3904.0	12.0	3904.0	12.0
ALB	335047.0	3605.0	335047.0	3605.0
...	...	...	...	...
WSM	17077.0	31.0	17057.0	31.0
YEM	11945.0	2159.0	11945.0	2159.0
ZAF	4072765.0	102595.0	4072765.0	102595.0
ZMB	349842.0	4077.0	349842.0	4077.0
ZWE	266387.0	5740.0	266386.0	5740.0

255 rows × 4 columns

```
In [25]: compare_df[compare_df['new_cases']!=compare_df['total_cases']]
```

Out[25]:

	new_cases	new_deaths	total_cases	total_deaths
iso_code				
BDI	54674.0	15.0	54569.0	15.0
BLZ	71416.0	688.0	71414.0	688.0
ECU	1079039.0	36050.0	1077445.0	36050.0
ESH	0.0	0.0	NaN	NaN
FJI	69054.0	885.0	69047.0	885.0
FSM	31852.0	65.0	31765.0	65.0
GNQ	17228.0	183.0	17130.0	183.0
GTM	1250398.0	20203.0	1250371.0	20203.0
HKG	0.0	0.0	NaN	NaN
MAC	0.0	0.0	NaN	NaN
MDV	186695.0	316.0	186694.0	316.0
MRT	63996.0	997.0	63872.0	997.0
MWI	89256.0	2686.0	89168.0	2686.0
MYT	42902.0	187.0	42027.0	187.0
OWID_AFR	13146831.0	259121.0	13145380.0	259117.0
OWID_ASI	301564180.0	1637335.0	301499099.0	1637249.0
OWID_CYN	0.0	0.0	NaN	NaN
OWID_ENG	0.0	0.0	NaN	NaN
OWID_HIC	429044052.0	3001093.0	429044049.0	2997359.0
OWID_LIC	1944687.0	43530.0	1944334.0	43529.0
OWID_LMC	92019711.0	1188056.0	91954400.0	1188026.0
OWID_NAM	124492698.0	1671512.0	124492666.0	1671178.0
OWID_NIR	0.0	0.0	NaN	NaN
OWID_OCE	15003468.0	33024.0	15003352.0	32918.0
OWID_SAM	68811012.0	1357619.0	68809418.0	1354187.0
OWID_SCT	0.0	0.0	NaN	NaN
OWID_UMC	251756125.0	2824538.0	251753518.0	2824452.0
OWID_WLS	0.0	0.0	NaN	NaN
OWID_WRL	775935057.0	7060988.0	775866783.0	7057132.0
PHL	4205462.0	66864.0	4173631.0	66864.0
PLW	6374.0	10.0	6372.0	10.0
SXM	11052.0	92.0	11051.0	92.0
THA	4799181.0	34775.0	4799180.0	34715.0
TWN	0.0	0.0	NaN	NaN
VIR	25391.0	132.0	25389.0	132.0
WSM	17077.0	31.0	17057.0	31.0
ZWE	266387.0	5740.0	266386.0	5740.0

4. 筛选出特定的行和列

We will remove those location with incoherences. Let us first chose the columns of interest

In [26]:
```
dem_cols =['location','population_density', 'median_age', 'aged_65_older', 'aged_70_older',
       'gdp_per_capita', 'extreme_poverty', 'cardiovasc_death_rate',
       'diabetes_prevalence',
       'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index']
tot_cols = ['total_cases_per_million',
       'total_deaths_per_million',
       'icu_patients_per_million',
       'hosp_patients_per_million',
       'weekly_icu_admissions_per_million',
       'weekly_hosp_admissions_per_million',
       'total_tests_per_thousand',
       'total_vaccinations_per_hundred',
       'people_fully_vaccinated_per_hundred']
```

In [27]:
```
covid19[dem_cols].groupby(covid19.iso_code).first() # 选出每组的第一行数据
```

Out[27]:

	location	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	extreme_poverty	cardiovasc_death_rate	diabetes_prevalence	handwashing_facilities	hos
iso_code											
ABW	Aruba	584.800	41.2	13.085	7.452	35973.781	NaN	NaN	11.62	NaN	
AFG	Afghanistan	54.422	18.6	2.581	1.337	1803.987	NaN	597.029	9.59	37.746	
AGO	Angola	23.890	16.8	2.405	1.362	5819.495	NaN	276.045	3.94	26.664	
AIA	Anguilla	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
ALB	Albania	104.871	38.0	13.188	8.643	11803.431	1.1	304.195	10.08	NaN	
...	...	...	...	...	...	...	...	...	...	...	
WSM	Samoa	69.413	22.0	5.606	3.564	6021.557	NaN	348.977	9.21	NaN	
YEM	Yemen	53.508	20.3	2.922	1.583	1479.147	18.8	495.003	5.35	49.542	
ZAF	South Africa	46.754	27.3	5.344	3.053	12294.876	18.9	200.380	5.52	43.993	
ZMB	Zambia	22.995	17.7	2.480	1.542	3689.251	57.5	234.499	3.94	13.938	
ZWE	Zimbabwe	42.729	19.6	2.822	1.882	1899.775	21.4	307.846	1.82	36.791	

255 rows × 13 columns

In [28]:
```
country_stats_df = pd.concat([covid19[dem_cols].groupby(covid19.iso_code).first(), # 使用 first 筛选出一行（每行都一样）
                              covid19[tot_cols].groupby(covid19.iso_code).max()], axis=1) # 使用 max 筛选出最新的数据
country_stats_df = country_stats_df[compare_df['new_cases']==compare_df['total_cases']] # 筛选出 compatible 的行
country_stats_df
```

Out[28]:

	location	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	extreme_poverty	cardiovasc_death_rate	diabetes_prevalence	handwashing_facilities	...
iso_code											
ABW	Aruba	584.800	41.2	13.085	7.452	35973.781	NaN	NaN	11.62	NaN	...
AFG	Afghanistan	54.422	18.6	2.581	1.337	1803.987	NaN	597.029	9.59	37.746	...
AGO	Angola	23.890	16.8	2.405	1.362	5819.495	NaN	276.045	3.94	26.664	...
AIA	Anguilla	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
ALB	Albania	104.871	38.0	13.188	8.643	11803.431	1.1	304.195	10.08	NaN	...
...	...	...	...	...	...	...	...	...	...	...	...
VUT	Vanuatu	22.662	23.1	4.394	2.620	2921.909	13.2	546.300	12.02	25.209	...
WLF	Wallis and Futuna	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
YEM	Yemen	53.508	20.3	2.922	1.583	1479.147	18.8	495.003	5.35	49.542	...
ZAF	South Africa	46.754	27.3	5.344	3.053	12294.876	18.9	200.380	5.52	43.993	...
ZMB	Zambia	22.995	17.7	2.480	1.542	3689.251	57.5	234.499	3.94	13.938	...

218 rows × 22 columns

5. 统计每列的非 NaN 数：.notna().sum(axis=0)

Let us remove rows with NaN values

In [29]:
```
country_stats_df.notna().sum(axis=0)
```

Out[29]:
```
location                               218
population_density                     195
median_age                             180
aged_65_older                          175
aged_70_older                          178
gdp_per_capita                         177
extreme_poverty                        116
cardiovasc_death_rate                  179
diabetes_prevalence                    188
handwashing_facilities                  84
hospital_beds_per_thousand             161
life_expectancy                        211
human_development_index                173
total_cases_per_million                218
total_deaths_per_million               218
icu_patients_per_million                38
hosp_patients_per_million               36
weekly_icu_admissions_per_million       22
weekly_hosp_admissions_per_million      31
total_tests_per_thousand               171
total_vaccinations_per_hundred         203
people_fully_vaccinated_per_hundred    201
dtype: int64
```

6. 统计不含 NaN 的行数：.notna().all(axis = 1).sum()

`.notna().all(axis = 1)`：判断每一行是否不含NaN

```
In [30]: country_stats_df.notna().all(axis = 1).sum()
```

```
Out[30]: 0
```

Need to be more tolerant otherwise not much data left

```
In [31]: country_stats_df[dem_cols+tot_cols].notna().sum(axis = 0)
```

```
Out[31]: location                              218
         population_density                    195
         median_age                            180
         aged_65_older                         175
         aged_70_older                         178
         gdp_per_capita                        177
         extreme_poverty                       116
         cardiovasc_death_rate                 179
         diabetes_prevalence                   188
         handwashing_facilities                 84
         hospital_beds_per_thousand            161
         life_expectancy                       211
         human_development_index               173
         total_cases_per_million               218
         total_deaths_per_million              218
         icu_patients_per_million               38
         hosp_patients_per_million              36
         weekly_icu_admissions_per_million      22
         weekly_hosp_admissions_per_million     31
         total_tests_per_thousand              171
         total_vaccinations_per_hundred        203
         people_fully_vaccinated_per_hundred   201
         dtype: int64
```

```
In [32]: [l for l in dem_cols+tot_cols]
```

```
Out[32]: ['location',
          'population_density',
          'median_age',
          'aged_65_older',
          'aged_70_older',
          'gdp_per_capita',
          'extreme_poverty',
          'cardiovasc_death_rate',
          'diabetes_prevalence',
          'handwashing_facilities',
          'hospital_beds_per_thousand',
          'life_expectancy',
          'human_development_index',
          'total_cases_per_million',
          'total_deaths_per_million',
          'icu_patients_per_million',
          'hosp_patients_per_million',
          'weekly_icu_admissions_per_million',
          'weekly_hosp_admissions_per_million',
          'total_tests_per_thousand',
          'total_vaccinations_per_hundred',
          'people_fully_vaccinated_per_hundred']
```

```
In [33]: kept_cols = [l for l in dem_cols+tot_cols if country_stats_df[l].notna().sum()>160]
         kept_cols
```

```
Out[33]: ['location',
          'population_density',
          'median_age',
          'aged_65_older',
          'aged_70_older',
          'gdp_per_capita',
          'cardiovasc_death_rate',
          'diabetes_prevalence',
          'hospital_beds_per_thousand',
          'life_expectancy',
          'human_development_index',
          'total_cases_per_million',
          'total_deaths_per_million',
          'total_tests_per_thousand',
          'total_vaccinations_per_hundred',
          'people_fully_vaccinated_per_hundred']
```

```
In [34]: country_stats_df[kept_cols].notna().all(axis = 1).sum()
```

```
Out[34]: 132
```

In [35]:
```
country_stats_df = country_stats_df[kept_cols][country_stats_df[kept_cols].notna().all(axis = 1)]
country_stats_df
```

Out[35]:

	location	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	cardiovasc_death_rate	diabetes_prevalence	hospital_beds_per_thousand	life_expectancy
iso_code										
AFG	Afghanistan	54.422	18.6	2.581	1.337	1803.987	597.029	9.59	0.50	64.83
ALB	Albania	104.871	38.0	13.188	8.643	11803.431	304.195	10.08	2.89	78.57
ARE	United Arab Emirates	112.442	34.0	1.144	0.526	67293.483	317.840	17.26	1.20	77.97
ARG	Argentina	16.177	31.9	11.198	7.441	18933.907	191.032	5.50	5.00	76.67
ARM	Armenia	102.931	35.7	11.232	7.571	8787.580	341.010	7.11	4.20	75.09
...	...	...	...	...	...	...	...	...	...	...
VCT	Saint Vincent and the Grenadines	281.787	31.8	7.724	4.832	10727.146	252.675	11.62	2.60	72.53
VNM	Vietnam	308.127	32.6	7.150	4.718	6171.884	245.465	6.00	2.60	75.40
YEM	Yemen	53.508	20.3	2.922	1.583	1479.147	495.003	5.35	0.70	66.12
ZAF	South Africa	46.754	27.3	5.344	3.053	12294.876	200.380	5.52	2.32	64.13
ZMB	Zambia	22.995	17.7	2.480	1.542	3689.251	234.499	3.94	2.00	63.89

132 rows × 16 columns

C - Elementary statistic's sumary of our data 数据分析

1. 查看dataframe的基本信息：.info()

In [36]:
```
country_stats_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 132 entries, AFG to ZMB
Data columns (total 16 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   location                             132 non-null    object 
 1   population_density                   132 non-null    float64
 2   median_age                           132 non-null    float64
 3   aged_65_older                        132 non-null    float64
 4   aged_70_older                        132 non-null    float64
 5   gdp_per_capita                       132 non-null    float64
 6   cardiovasc_death_rate                132 non-null    float64
 7   diabetes_prevalence                  132 non-null    float64
 8   hospital_beds_per_thousand           132 non-null    float64
 9   life_expectancy                      132 non-null    float64
 10  human_development_index              132 non-null    float64
 11  total_cases_per_million              132 non-null    float64
 12  total_deaths_per_million             132 non-null    float64
 13  total_tests_per_thousand             132 non-null    float64
 14  total_vaccinations_per_hundred       132 non-null    float64
 15  people_fully_vaccinated_per_hundred  132 non-null    float64
dtypes: float64(15), object(1)
memory usage: 17.5+ KB
```

2. 计算dataframe的基本数据：.describe()

Concentrate on some important quantities

In [37]:
```
imp_cols = ['population_density',   'median_age','diabetes_prevalence','hospital_beds_per_thousand', 'life_expectancy', 'total_cases_per_mil
'total_deaths_per_million', 'total_tests_per_thousand', 'total_vaccinations_per_hundred','people_fully_vaccinated_per_hundred'
]
country_stats_df[imp_cols].describe()
```

Out[37]:

	population_density	median_age	diabetes_prevalence	hospital_beds_per_thousand	life_expectancy	total_cases_per_million	total_deaths_per_million	total_tests_per_thousand	total_vaccir
count	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	132.000000	
mean	160.968386	31.856818	7.863712	3.019553	73.952576	187178.168788	1528.914970	1980.795235	
std	257.086231	8.908716	3.852164	2.458776	6.816569	193802.856393	1489.635439	4033.809118	
min	1.980000	15.100000	0.990000	0.100000	53.280000	312.509000	11.846000	5.219000	
25%	34.509000	25.375000	5.262500	1.300000	69.780000	23414.595500	201.001750	179.406250	
50%	84.304000	31.900000	7.110000	2.500000	75.210000	123838.865000	1118.925000	698.804500	
75%	206.285750	39.700000	9.907500	3.930000	78.770000	280978.750000	2501.357000	2004.198750	
max	1935.907000	48.200000	22.020000	13.050000	84.630000	763598.600000	6601.110000	32925.826000	

3. 计算各列的quantile：.quantile()

注意到第一列的最后一行远大于倒数第二行，因此这是一个outlier。其他列可用同样的方法判断是否存在outlier

In [38]:
```
country_stats_df[imp_cols].quantile([0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0])
```

Out[38]:

	population_density	median_age	diabetes_prevalence	hospital_beds_per_thousand	life_expectancy	total_cases_per_million	total_deaths_per_million	total_tests_per_thousand	total_vaccinati
0.0	1.9800	15.10	0.990	0.100	53.280	312.5090	11.8460	5.2190	
0.1	15.0016	18.82	3.940	0.700	64.007	3674.5024	56.8762	33.2708	
0.2	24.3692	23.12	4.794	1.100	67.114	11127.4674	143.1396	122.2254	
0.3	45.6207	27.30	5.629	1.400	71.561	32636.4686	366.1514	287.4118	
0.4	65.6856	29.48	6.374	1.900	73.668	84749.4090	681.6962	445.6362	
0.5	84.3040	31.90	7.110	2.500	75.210	123838.8650	1118.9250	698.8045	
0.6	104.8968	34.06	8.098	2.900	76.692	168988.8120	1617.9724	1147.3790	
0.7	133.8611	38.21	9.129	3.400	77.895	241302.1770	2125.0556	1771.1517	
0.8	231.7654	41.36	10.774	4.508	80.270	342802.5240	2764.1752	2479.5118	
0.9	347.1957	43.39	12.782	6.614	82.289	511944.4620	3492.4439	4101.6923	
1.0	1935.9070	48.20	22.020	13.050	84.630	763598.6000	6601.1100	32925.8260	

4. 计算各列的skewness：.skew()

若存在outlier，则该列的skewness会较大

In [39]:
```
country_stats_df[imp_cols].skew()
```

Out[39]:
```
population_density                    4.320390
median_age                           -0.101254
diabetes_prevalence                   0.964258
hospital_beds_per_thousand            1.571958
life_expectancy                      -0.618743
total_cases_per_million               1.048853
total_deaths_per_million              1.060179
total_tests_per_thousand              5.022126
total_vaccinations_per_hundred        0.037235
people_fully_vaccinated_per_hundred  -0.398983
dtype: float64
```

5. 后续分析：outlier可能是哪些国家

There is a factor 10 between the 9th and the 10th decile, which country is concerned?

Is it China?

In [40]:
```
country_stats_df.loc['CHN']
```

Out[40]:
```
location                                  China
population_density                      147.674
median_age                                 38.7
aged_65_older                            10.641
aged_70_older                             5.929
gdp_per_capita                        15308.712
cardiovasc_death_rate                   261.899
diabetes_prevalence                        9.74
hospital_beds_per_thousand                 4.34
life_expectancy                           76.91
human_development_index                   0.761
total_cases_per_million               69726.805
total_deaths_per_million                 85.817
total_tests_per_thousand               6461.913
total_vaccinations_per_hundred           244.84
people_fully_vaccinated_per_hundred       89.54
Name: CHN, dtype: object
```

根据total_tests_per_thousand排序

In [41]:
```
country_stats_df[country_stats_df.total_tests_per_thousand >5000].sort_values('total_tests_per_thousand')
```

Out[41]:

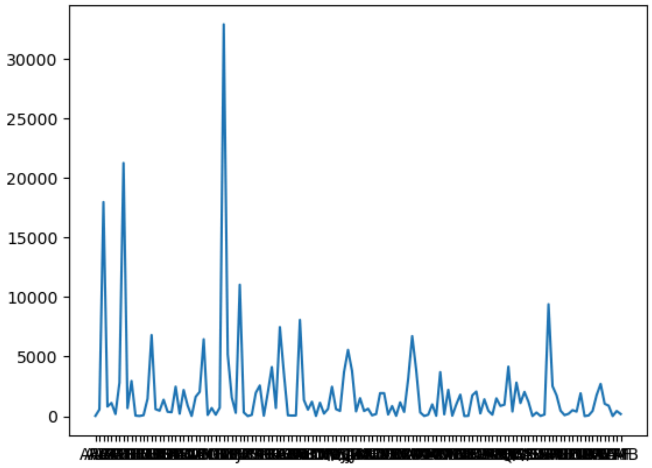
	location	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	cardiovasc_death_rate	diabetes_prevalence	hospital_beds_per_thousand	life_expectancy
iso_code										
CZE	Czechia	137.176	43.3	19.027	11.580	32605.906	227.485	6.82	6.63	79.38
ISR	Israel	402.606	30.6	11.733	7.359	33132.320	93.320	6.74	2.99	82.97
CHN	China	147.674	38.7	10.641	5.929	15308.712	261.899	9.74	4.34	76.91
LUX	Luxembourg	231.447	39.7	14.312	9.842	94277.965	128.275	4.42	4.51	82.25
BHR	Bahrain	1935.907	32.4	2.372	1.387	43290.705	151.689	16.52	2.00	77.29
GBR	United Kingdom	272.898	40.8	18.517	12.527	39753.244	122.137	4.28	2.54	81.32
GRC	Greece	83.479	45.3	20.396	14.524	24574.382	175.695	4.55	4.21	82.24
SVK	Slovakia	113.128	41.2	15.070	9.167	30155.152	287.959	7.29	5.82	77.54
DNK	Denmark	136.520	42.3	19.677	12.325	46682.515	114.767	6.41	2.50	80.90
ARE	United Arab Emirates	112.442	34.0	1.144	0.526	67293.483	317.840	17.26	1.20	77.97
AUT	Austria	106.749	44.4	19.202	13.748	45436.686	145.183	6.35	7.37	81.54
CYP	Cyprus	127.657	37.3	13.416	8.563	32415.132	141.171	9.24	3.40	80.98

In [42]:

```
# plt.plot(country_stats_df.total_vaccinations)
plt.plot(country_stats_df.total_tests_per_thousand)
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x1767118d0>]
```



D - Identify outliers 查找异常值

1. 查找outliers

Can build a function to detetct outliers for each variable

In [43]:

```
def getextremevalues(dfin):
    dfout = pd.DataFrame(columns=dfin.columns, data=None)
    for col in dfin.columns[1:10]:
        thirdq, firstq = dfin[col].quantile(0.75), dfin[col].quantile(0.25)
        interquartilerange = 1.5*(thirdq-firstq)
        extvalhigh, extvallow = interquartilerange+thirdq, firstq-interquartilerange
        df = dfin.loc[(dfin[col]>extvalhigh) | (dfin[col]<extvallow)]
        df = df.assign(varname = col,threshlow = extvallow,threshhigh = extvalhigh)
        dfout = pd.concat([dfout, df])
    return dfout
```

In [44]:

```
getextremevalues(country_stats_df)
```

```
/var/folders/65/bgzbm5n50g33w2jxwhjx9q7c0000gn/T/ipykernel_28461/2077076489.py:9: FutureWarning: The behavior of DataFrame concatenation wi
th empty or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA columns when determining the res
ult dtypes. To retain the old behavior, exclude the relevant entries before the concat operation.
  dfout = pd.concat([dfout, df])
```

Out[44]:

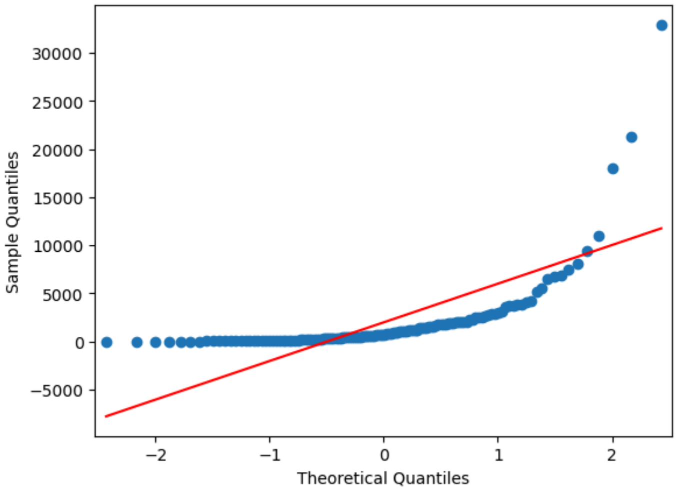
	location	population_density	median_age	aged_65_older	aged_70_older	gdp_per_capita	cardiovasc_death_rate	diabetes_prevalence	hospital_beds_per_thousand	life_expectancy	hu
BGD	Bangladesh	1265.036	27.5	5.098	3.262	3523.984	298.003	8.38	0.800	72.59	
BHR	Bahrain	1935.907	32.4	2.372	1.387	43290.705	151.689	16.52	2.000	77.29	
BRB	Barbados	664.463	39.8	14.952	9.473	16978.068	170.050	13.57	5.800	79.19	
KOR	South Korea	527.967	43.4	13.914	8.622	35938.374	85.998	6.80	12.270	83.03	
LBN	Lebanon	594.561	31.1	8.514	5.430	13367.565	266.591	12.71	2.900	78.93	
MLT	Malta	1454.037	42.4	19.426	11.324	36513.323	168.711	8.83	4.485	82.53	
MUS	Mauritius	622.962	37.4	10.945	5.884	20292.745	224.644	22.02	3.400	74.99	
NLD	Netherlands	508.544	43.2	18.779	11.881	48472.545	109.361	5.29	3.320	82.28	
ARE	United Arab Emirates	112.442	34.0	1.144	0.526	67293.483	317.840	17.26	1.200	77.97	
BRN	Brunei	81.347	32.4	4.591	2.382	71809.251	201.285	12.79	2.700	75.86	
IRL	Ireland	69.874	38.7	13.928	8.678	67335.293	126.459	3.28	2.960	82.30	
LUX	Luxembourg	231.447	39.7	14.312	9.842	94277.965	128.275	4.42	4.510	82.25	
QAT	Qatar	227.322	31.9	1.307	0.617	116935.600	176.690	16.52	1.200	80.23	
AFG	Afghanistan	54.422	18.6	2.581	1.337	1803.987	597.029	9.59	0.500	64.83	
AZE	Azerbaijan	119.309	32.4	6.018	3.871	15847.419	559.812	7.11	4.700	73.00	
UKR	Ukraine	77.390	41.4	16.462	11.133	7894.393	539.849	7.11	8.800	72.06	
ARE	United Arab Emirates	112.442	34.0	1.144	0.526	67293.483	317.840	17.26	1.200	77.97	
EGY	Egypt	97.999	25.3	5.159	2.891	10550.206	525.432	17.31	1.600	71.99	
MUS	Mauritius	622.962	37.4	10.945	5.884	20292.745	224.644	22.02	3.400	74.99	
SAU	Saudi Arabia	15.322	31.9	3.295	1.845	49045.411	259.538	17.72	2.700	75.13	
BLR	Belarus	46.858	40.3	14.799	9.788	17167.967	443.129	5.18	11.000	74.79	
DEU	Germany	237.016	46.6	21.453	15.957	45229.245	156.139	8.31	8.000	81.33	
JPN	Japan	347.778	48.2	27.049	18.493	39002.223	79.370	5.72	13.050	84.63	
KOR	South Korea	527.967	43.4	13.914	8.622	35938.374	85.998	6.80	12.270	83.03	
RUS	Russia	8.823	39.6	14.178	9.393	24765.954	431.297	6.18	8.050	72.58	
UKR	Ukraine	77.390	41.4	16.462	11.133	7894.393	539.849	7.11	8.800	72.06	
CAF	Central African Republic	7.479	18.3	3.655	2.251	661.240	435.727	6.10	1.000	53.28	

2. 和标准分布比较：Q-Q plots

Useful representation: **quantile-quantile (Q-Q)** plots (默认和normal distribution比较)

```
In [45]: import statsmodels.api as sm
         import scipy

         sm.qqplot(country_stats_df[['total_tests_per_thousand']].sort_values(['total_tests_per_thousand']),line='s')
         plt.show()
```

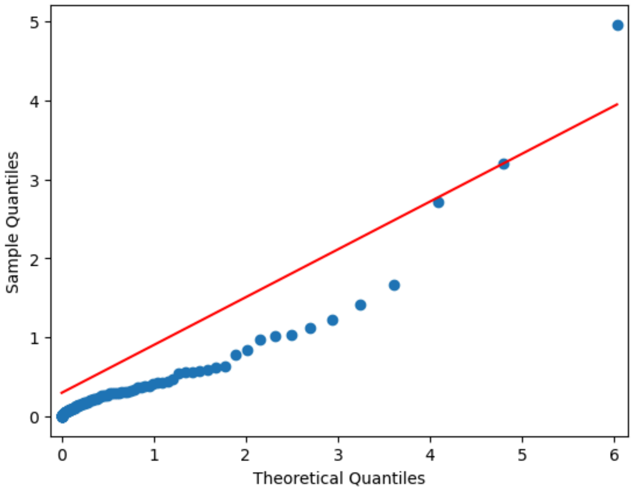


可以使用？来查询函数的用法

```
In [46]: #sm.qqplot?
```

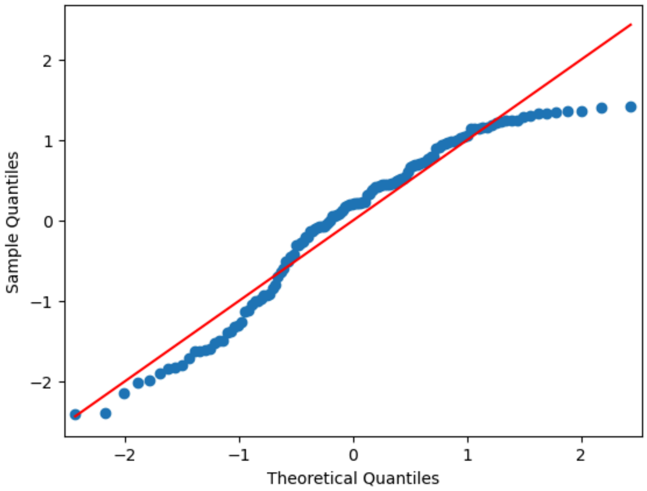
将normal distribution替换成exponential distribution

```
In [47]: sm.qqplot(country_stats_df[['total_tests_per_thousand']].sort_values(['total_tests_per_thousand']),line='s', dist = scipy.stats.distribution
         plt.show()
```



将 human development index 和 normal distribution 比较

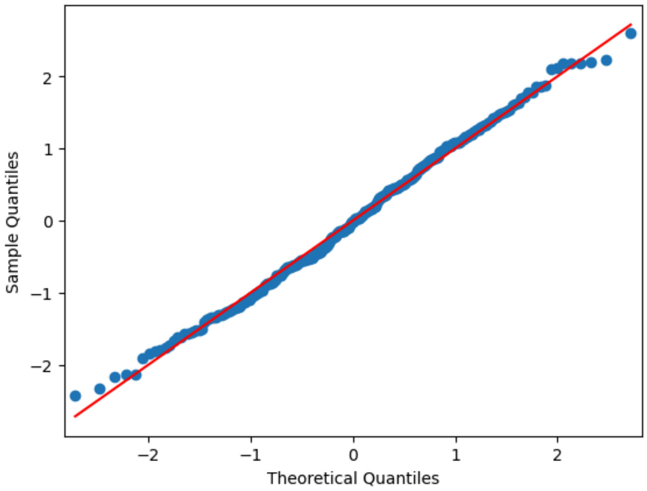
```
In [48]: sm.qqplot(country_stats_df[['human_development_index']].sort_values(['human_development_index']),line='s', fit=True)
         plt.show()
```



将从 normal distribution 中 sample 的样本和 normal distribution 作比较

In [49]:

```
norm_drawings = np.random.randn(300)
sm.qqplot(norm_drawings,line='s', fit=True)
plt.show()
```

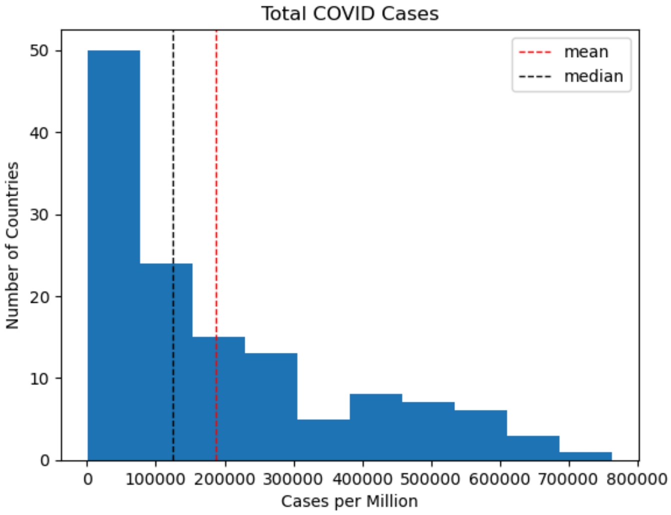


E - Histograms, boxplots, and violin plots 数据可视化

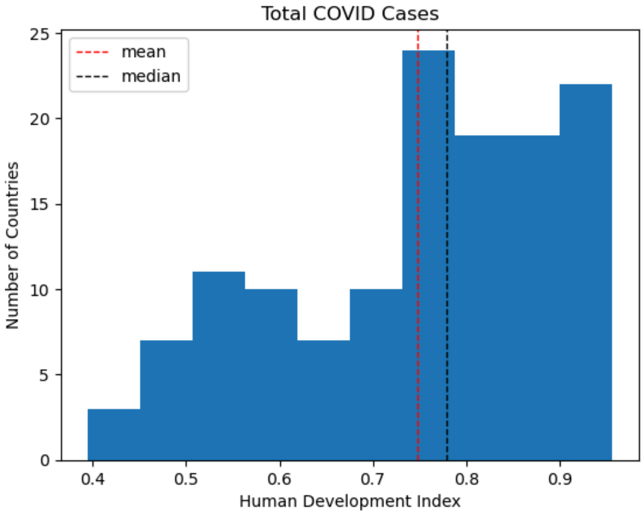
1. 柱状图：plt.hist()

In [50]:

```
plt.hist(country_stats_df['total_cases_per_million'], bins=10)
plt.axvline(country_stats_df.total_cases_per_million.mean(), color='red',
   linestyle='dashed', linewidth=1, label='mean')
plt.axvline(country_stats_df.total_cases_per_million.median(), color='black',
   linestyle='dashed', linewidth=1, label='median')
plt.title("Total COVID Cases")
plt.xlabel('Cases per Million')
plt.ylabel("Number of Countries")
plt.legend()
plt.show()
```

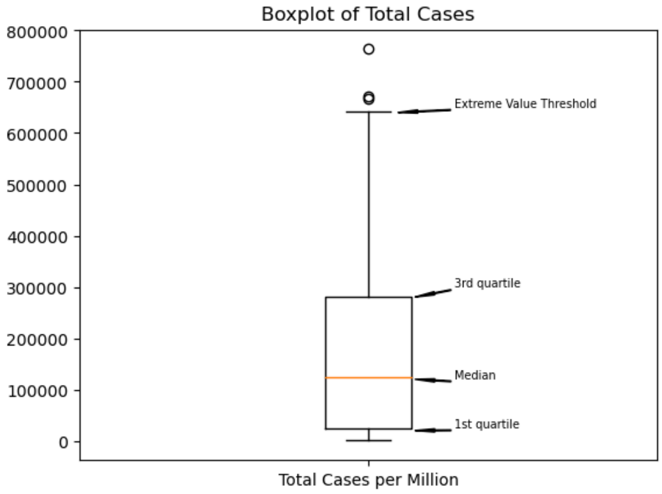


In [51]:
```
plt.hist(country_stats_df['human_development_index'], bins=10)
plt.axvline(country_stats_df.human_development_index.mean(), color='red',
   linestyle='dashed', linewidth=1, label='mean')
plt.axvline(country_stats_df.human_development_index.median(), color='black',
   linestyle='dashed', linewidth=1, label='median')
plt.title("Total COVID Cases")
plt.xlabel('Human Development Index')
plt.ylabel("Number of Countries")
plt.legend()
plt.show()
```

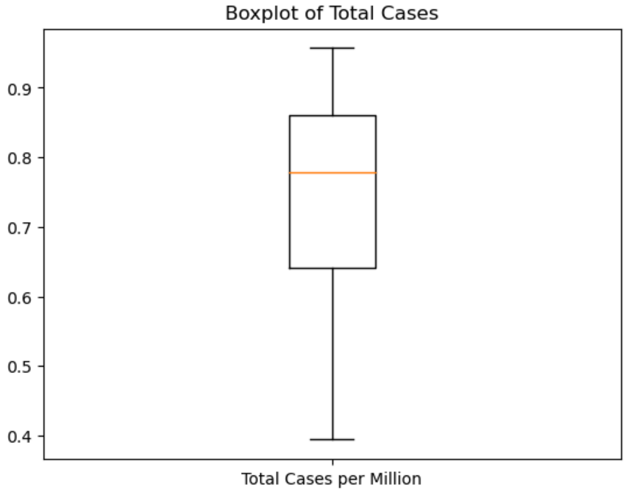


2. 箱型图：plt.boxplot()

In [52]:
```
plt.boxplot(country_stats_df.total_cases_per_million.dropna(), labels=['Total Cases per Million'])
plt.annotate('Extreme Value Threshold', xy=(1.05,640000), xytext=(1.15,650000), size=7, arrowprops=dict(facecolor='black', headwidth=2, widt
plt.annotate('3rd quartile', xy=(1.08,280000), xytext=(1.15,300000), size=7, arrowprops=dict(facecolor='black', headwidth=2, width=0.5, shri
plt.annotate('Median', xy=(1.08,120000), xytext=(1.15,120000), size=7, arrowprops=dict(facecolor='black', headwidth=2, width=0.5, shrink=0.0
plt.annotate('1st quartile', xy=(1.08,20000), xytext=(1.15,25000), size=7, arrowprops=dict(facecolor='black', headwidth=2, width=0.5, shrink
plt.title("Boxplot of Total Cases")
plt.show()
```



In [53]:
```
plt.boxplot(country_stats_df.human_development_index.dropna(), labels=['Total Cases per Million'])
plt.title("Boxplot of Total Cases")
plt.show()
```

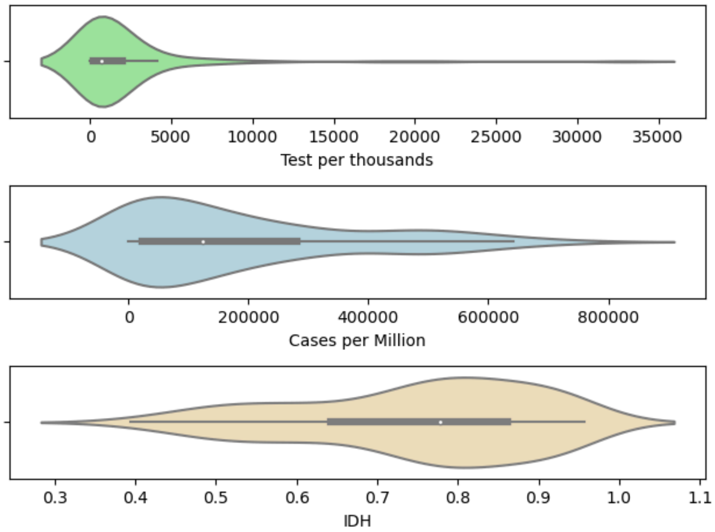


3. 小提琴图：sns.violinplot()

In [54]:
```
import seaborn as sns
fig = plt.figure()

ax0 = plt.subplot(3,1,1)
ax0.set_xlabel("Test per thousands")
sns.violinplot(data=country_stats_df.total_tests_per_thousand, color="lightgreen", orient="h")
ax0.set_yticklabels([])
ax1 = plt.subplot(3,1,2)
ax1.set_xlabel("Cases per Million")
sns.violinplot(data=country_stats_df.total_cases_per_million, color="lightblue", orient="h")
ax1.set_yticklabels([])
ax2 = plt.subplot(3,1,3)
ax2.set_xlabel("IDH")
sns.violinplot(data=country_stats_df.human_development_index, color="wheat",  orient="h")
ax2.set_yticklabels([])
plt.tight_layout()
plt.show()
```

```
/Users/ren/anaconda3/lib/python3.10/site-packages/seaborn/categorical.py:486: FutureWarning: Series.__getitem__ treating keys as positions
is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value b
y position, use `ser.iloc[pos]`
  if np.isscalar(data[0]):
```



F - Examining multivariate relationships between Features and targets 相关性和拟合

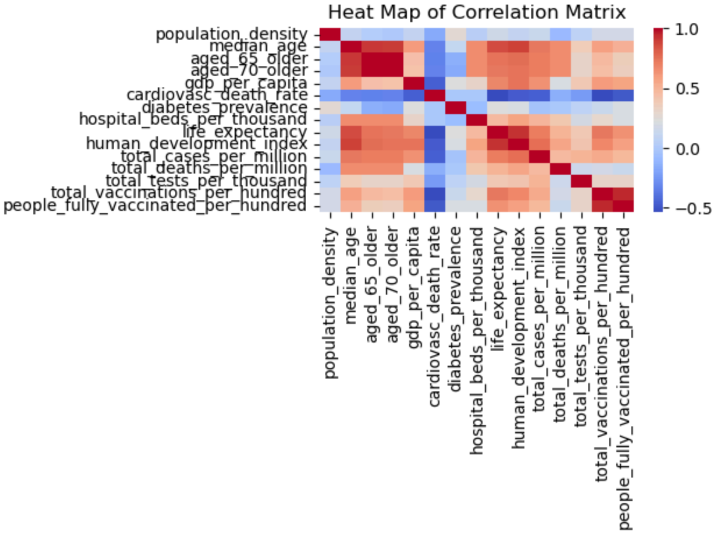
In [55]:
```
kept_cols[1:]
```

Out[55]:
```
['population_density',
 'median_age',
 'aged_65_older',
 'aged_70_older',
 'gdp_per_capita',
 'cardiovasc_death_rate',
 'diabetes_prevalence',
 'hospital_beds_per_thousand',
 'life_expectancy',
 'human_development_index',
 'total_cases_per_million',
 'total_deaths_per_million',
 'total_tests_per_thousand',
 'total_vaccinations_per_hundred',
 'people_fully_vaccinated_per_hundred']
```

1. 相关性热图：df.corr() 和 sns.heatmap

In [56]:

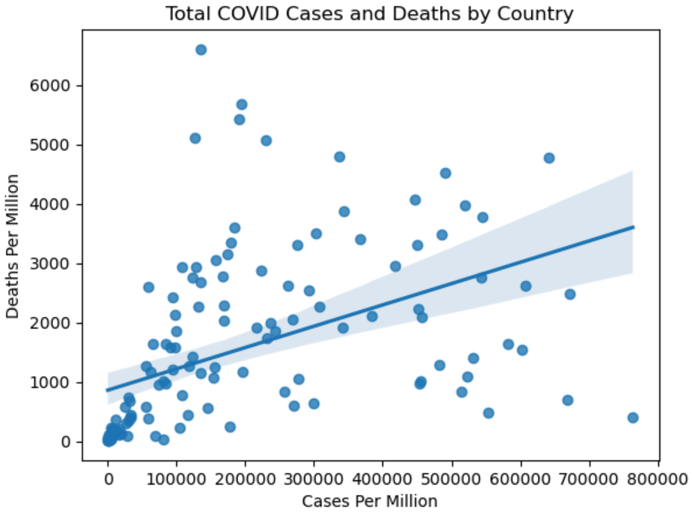
```
eval_col = kept_cols[1:]
corrmatrix = country_stats_df[eval_col].corr(method="pearson")
sns.heatmap(corrmatrix, xticklabels =
  eval_col, yticklabels=eval_col,
  cmap="coolwarm")
plt.title('Heat Map of Correlation Matrix')
plt.tight_layout()
plt.show()
```



2. 线性回归：sns.regplot()

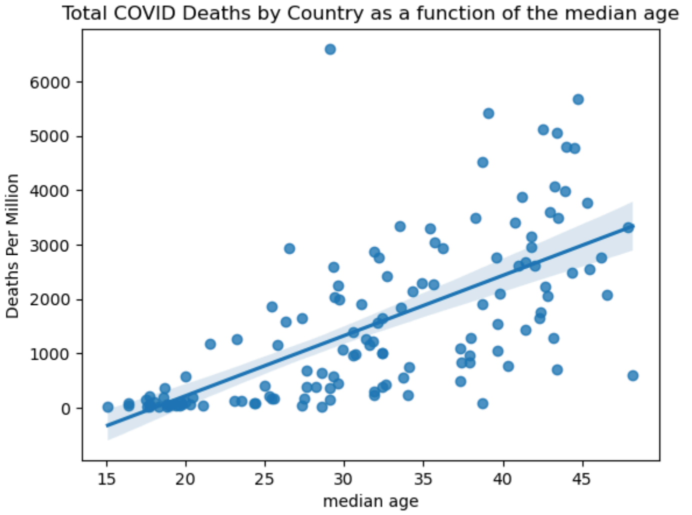
In [57]:

```
ax = sns.regplot(x="total_cases_per_million",  y="total_deaths_per_million", data=country_stats_df)
ax.set(xlabel="Cases Per Million", ylabel="Deaths Per Million", title="Total COVID Cases and Deaths by Country")
plt.show()
```



In [58]:

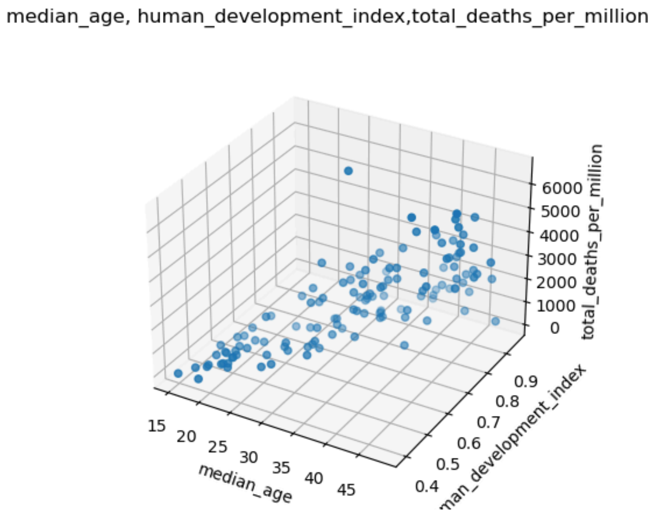
```
ax = sns.regplot(x="median_age",  y="total_deaths_per_million", data=country_stats_df)
ax.set(xlabel="median age", ylabel="Deaths Per Million", title="Total COVID Deaths by Country as a function of the median age")
plt.show()
```



3. 3D可视化：ax.scatter3D()

In [59]:

```
fig = plt.figure()
plt.suptitle("median_age, human_development_index,total_deaths_per_million")
ax = plt.axes(projection='3d')
ax.set_xlabel("median_age")
ax.set_ylabel("human_development_index")
ax.set_zlabel("total_deaths_per_million")
ax.scatter3D(country_stats_df.median_age, country_stats_df.human_development_index,  country_stats_df.total_deaths_per_million)
plt.show()
```


median_age, human_development_index,total_deaths_per_million
median_age
man_development_index
total_deaths_per_million