# CORD-19 Analyse affiliation data

In general, this notebook is designated to analyse collected SCOPUS affiliation data.

First, relevant packages must be imported into the notebook.

In [1]:

```
import numpy as np
import pandas as pd
import csv
import ast
import collections
import matplotlib.pyplot as plt
import datetime
import re
import json
import pycountry
import time
import folium
from pybtex.database import parse_file, BibliographyData, Entry
from urllib.parse import urlparse
from collections import Counter
```

Thusly, the fetched affiliation data is read from the PKL-file and incorporated into a DataFrame for further processing.

In [2]:

```
read_affiliation = pd.read_pickle('extra_info_CS5099.pkl')
df_current_extra_info = pd.DataFrame()
df_current_extra_info['affiliation'] = read_affiliation['affiliation']
df_current_extra_info
```

Out[2]:

| | affiliation |
|---|---|
| **0** | [{'affiliation-country': 'United States'}, {'a... |
| **1** | [{'affiliation-country': 'United States'}, {'a... |
| **2** | [{'affiliation-country': 'United States'}, {'a... |
| **3** | [{'affiliation-country': 'United States'}, {'a... |
| **4** | [{'affiliation-country': 'United States'}, {'a... |
| **...** | ... |
| **74297** | {'affiliation-country': 'United States'} |
| **74298** | [{'affiliation-country': 'United States'}, {'a... |
| **74299** | {'affiliation-country': 'United States'} |
| **74300** | [{'affiliation-country': 'Turkey'}, {'affiliat... |
| **74301** | None |

74302 rows × 1 columns

In [3]:

```
df_current_extra_info.isnull().sum()
```

Out[3]:

```
affiliation    14700
dtype: int64
```

For expressiveness purposes, the number of None values must be considered by computing the ratio of None entries compared to the length of the DataFrame.

In [4]:

```
no_return_value = round(df_current_extra_info.isnull().sum() / len(df_current_extra_info) *
print(no_return_value)
```

```
affiliation    19.78
dtype: float64
```

Compared to the length of the dataset, ~19.8% of fetched SCOPUS data has no return value. Thusly, all rows which contain "None" values are dropped and the DataFrame is reindexed.

In [5]:

```
df_combined = df_current_extra_info.dropna()
df_combined = df_combined.reset_index(drop=True)
df_combined
```

Out[5]:

| | affiliation |
|---|---|
| **0** | [{'affiliation-country': 'United States'}, {'a... |
| **1** | [{'affiliation-country': 'United States'}, {'a... |
| **2** | [{'affiliation-country': 'United States'}, {'a... |
| **3** | [{'affiliation-country': 'United States'}, {'a... |
| **4** | [{'affiliation-country': 'United States'}, {'a... |
| **...** | ... |
| **59597** | [{'affiliation-country': 'United States'}, {'a... |
| **59598** | {'affiliation-country': 'United States'} |
| **59599** | [{'affiliation-country': 'United States'}, {'a... |
| **59600** | {'affiliation-country': 'United States'} |
| **59601** | [{'affiliation-country': 'Turkey'}, {'affiliat... |

59602 rows × 1 columns

The following functions support the creation of DataFrames based on the columns affiliation and core data.

In [6]:

```
def get_one_entry(dic):
    """
    This function receives a dictionary with one entry and returns it as transformed DataFr
    """
    df_affiliation_holder = pd.DataFrame(dic.items()).T
    df_affiliation_holder.columns = df_affiliation_holder.iloc[0]
    df_affiliation_holder = df_affiliation_holder.drop(df_affiliation_holder.index[0])
    return df_affiliation_holder
```

In [7]:

```
def get_various_entries(dic):
    """
    This function receives a dictionary with more than one entry and returns it as transfor
    """
    df_affiliation_holder = pd.DataFrame.from_dict(dic, orient='columns')
    return df_affiliation_holder
```

The next cell creates the DataFrame which focus on affiliation data and holds one country name per row.

In [8]:

```
%%time
df_affiliation = pd.DataFrame()
df_affiliation_holder = pd.DataFrame()

for i in df_combined['affiliation']:
    string_holder = str(i)
    if string_holder[0] == "[":
        df_affiliation_holder = get_various_entries(i)
    else:
        df_affiliation_holder = get_one_entry(i)
    df_affiliation = pd.concat([df_affiliation_holder, df_affiliation],ignore_index=True)
df_affiliation
```

| | affiliation-country |
|---|---|
| **0** | Turkey |
| **1** | Turkey |
| **2** | Turkey |
| **3** | United States |
| **4** | United States |
| **...** | ... |
| **196820** | United States |
| **196821** | United States |
| **196822** | United States |
| **196823** | United States |
| **196824** | United States |

Subsequently, one publication can have multiple affiliations.

In [9]:

```
ser_country = df_affiliation['affiliation-country']
ser_country
```

Out[9]:

```
0                Turkey
1                Turkey
2                Turkey
3         United States
4         United States
              ...
196820    United States
196821    United States
196822    United States
196823    United States
196824    United States
Name: affiliation-country, Length: 196825, dtype: object
```

Thusly, the object contains leading figures where the researchers came from.

In [10]:

```
ser_countries_counted = ser_country.value_counts()
ser countries counted
```

Out[10]:

```
United States     43083
China             22499
United Kingdom    14666
Italy             10936
France             8229
                  ...
Russia                1
Greenland             1
Kiribati              1
Cayman Islands        1
Dominica              1
Name: affiliation-country, Length: 200, dtype: int64
```

For visualisation purposes, the data is transformed to a DataFrame and stored in a CSV.

In [11]:

```
df_counted_geocode = pd.DataFrame()
df_counted_geocode['Country'] = ser_countries_counted.index
df_counted_geocode['Affiliations'] = ser_countries_counted.values
df_counted_geocode.to_csv("counted_affiliation_countries.csv")
df_counted_geocode
```

Out[11]:

| | Country | Affiliations |
|---|---|---|
| **0** | United States | 43083 |
| **1** | China | 22499 |
| **2** | United Kingdom | 14666 |
| **3** | Italy | 10936 |
| **4** | France | 8229 |
| **...** | ... | ... |
| **195** | Russia | 1 |
| **196** | Greenland | 1 |
| **197** | Kiribati | 1 |
| **198** | Cayman Islands | 1 |
| **199** | Dominica | 1 |

200 rows × 2 columns

Subsequently, the affiliated countries are plotted on a choropleth map.

In [12]:

```
#Adapted code from https://www.python-graph-gallery.com/292-choropleth-map-with-folium
#Adapted code from https://python-visualization.github.io/folium/quickstart.html

countries_geo = "countries.geojson"
country_affiliations = "counted_affiliation_countries.csv"
state_data = pd.read_csv(country_affiliations)

m = folium.Map(location=[25,0], zoom_start=2)

folium.Choropleth(
    geo_data=countries_geo,
    name="choropleth",
    data=state_data,
    columns=["Country", "Affiliations"],
    key_on="properties.ADMIN",
    fill_color="YlGn",
    fill_opacity=0.8,
    line_opacity=0.2,
    nan_fill_color='white',
    legend_name="Publication affilitions per country",
).add_to(m)

folium.LayerControl().add_to(m)

m
```

Out[12]: