Course9 Assignment

which countries can we avoid the most tourists

1. Introduction

COVID-19 have been one of the most biggest interest in 2020. Although people want to travel all over the world, our transportation is limitted pretty much. Tourism/transportation is one of the basis of our pleasure, to watch some sports game, to travel in natures, go for a shopping, seeing families or so. Therefore many people is waiting the border get open again, and we are able to travel freely as we used to do, and so do I. Now my concern is which places/countries should I avoid during this pandemic. On the otherhand, budget is a big factor of the traveling, which is obviously the cheaper the better. So, in this assignment, I want to find the countries which I should avoid from 2 perspectives, COVID number and cost of traveling. The main idea here is not ignoring any rules or violating regulations. This result is just showing the idea of holiday planning.

2. Data and Methodology

in this section, i will collect all the necessary information for this project and process the collected data into a easier form.

2.1 preparation

[1]:

```
#import all necessary libraries for this project
import numpy as np
import pandas as pd
import json # library to handle JSON files

!conda install -c conda-forge geopy --yes # uncomment this line if you haven't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans
!conda install -c conda-forge folium=0.5.0 --yes
import folium

print('Folium installed and imported!')
print('done')
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    certifi-2020.12.5          |   py36h5fab9bb_1         143
KB  conda-forge
    geographiclib-1.50         |             py_0          34
KB  conda-forge
    geopy-2.1.0                |     pyhd3deb0d_0          64
KB  conda-forge

------------------------------------------------------------
                                           Total:         240
KB

The following NEW packages will be INSTALLED:

  geographiclib      conda-forge/noarch::geographiclib-1.50-
py_0
  geopy              conda-forge/noarch::geopy-2.1.0-
pyhd3deb0d_0

The following packages will be UPDATED:

  certifi                          2020.12.5-py36h5fab9bb_0
--> 2020.12.5-py36h5fab9bb_1



Downloading and Extracting Packages
geopy-2.1.0          | 64 KB     |
##################################### | 100%
certifi-2020.12.5    | 143 KB    |
##################################### | 100%
geographiclib-1.50   | 34 KB     |
##################################### | 100%
```

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve.
Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - folium=0.5.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    altair-4.1.0               |             py_1         614
KB  conda-forge
    branca-0.4.2               |     pyhd8ed1ab_0          26
KB  conda-forge
    folium-0.5.0               |             py_0          45
KB  conda-forge
    pandas-1.1.5               |   py36h284efc9_0        11.3
MB  conda-forge
    pytz-2021.1                |     pyhd8ed1ab_0         239
KB  conda-forge
    toolz-0.11.1               |             py_0          46
KB  conda-forge
    vincent-0.4.4              |             py_1          28
KB  conda-forge

------------------------------------------------------------
                                           Total:        12.3
MB

The following NEW packages will be INSTALLED:

  altair             conda-forge/noarch::altair-4.1.0-py_1
  branca             conda-forge/noarch::branca-0.4.2-
pyhd8ed1ab_0
  folium             conda-forge/noarch::folium-0.5.0-py_0
  pandas             conda-forge/linux-64::pandas-1.1.5-
py36h284efc9_0
```

```
  pytz                  conda-forge/noarch::pytz-2021.1-
pyhd8ed1ab_0
  toolz                 conda-forge/noarch::toolz-0.11.1-py_0
  vincent               conda-forge/noarch::vincent-0.4.4-py_1
```

```
Downloading and Extracting Packages
folium-0.5.0         | 45 KB     |
##################################### | 100%
branca-0.4.2         | 26 KB     |
##################################### | 100%
altair-4.1.0         | 614 KB    |
##################################### | 100%
pandas-1.1.5         | 11.3 MB   |
##################################### | 100%
pytz-2021.1          | 239 KB    |
##################################### | 100%
toolz-0.11.1         | 46 KB     |
##################################### | 100%
vincent-0.4.4        | 28 KB     |
##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Folium installed and imported!
done
```

[2]:

```
pip install pycountry
```

```
Collecting pycountry
  Downloading https://files.pythonhosted.org/packages/
76/73/6f1a412f14f68c273feea29a6ea9b9f1e268177d32e0e69ad6790d3
06312/pycountry-20.7.3.tar.gz (10.1MB)
     |████████████████████████████████| 10.1MB 2.8MB/s eta
0:00:01    |███████████                     | 3.6MB 927kB/s
eta 0:00:08     |█████████████████████           | 6.6MB
2.8MB/s eta 0:00:02     |████████████████████████▍       |
7.7MB 2.8MB/s eta 0:00:01████████      | 8.2MB 2.8MB/s eta
0:00:01     |█████████████████████████████    | 9.1MB 2.8MB/s
eta 0:00:01
Building wheels for collected packages: pycountry
  Building wheel for pycountry (setup.py) ... done
  Stored in directory: /home/jupyterlab/.cache/pip/wheels/
33/4e/a6/be297e6b83567e537bed9df4a93f8590ec01c1acfbcd405348
Successfully built pycountry
Installing collected packages: pycountry
Successfully installed pycountry-20.7.3
Note: you may need to restart the kernel to use updated
packages.
```

[3]:

```
import pycountry
```

[4]:

```
import matplotlib.pyplot as plt
```

2.2 Price index

in this section, i collect all the necessary information to know the price index. the reason to know this factor is to compare the price value between each countries. for this reason i use BigMc index, which compares the price of BigMc in each countries. from this factor, you can estimate how much your travel budget will be. the more expensive bigmc the more money you need.

```
[5]:
#collect data
```

```
#collect data
price_index = pd.read_csv('price_index.csv')
price_index.head()
```

```
[5]:
```

	date	iso_a3	currency_code	name	local_price	dollar_ex	dollar_price	GDP_dollar	adj_price	USD	EUR	GBP	JPY	CNY
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1/07/2011	ARG	ARS	Argentina	20.00	4.13	4.839	9138.	3.14	1.01	0.47	0.84	0.90	0.94
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1/07/2011	AUS	AUD	Australia	4.56	0.92	4.943	5558	5.79	0.11	-0.1	0.02	0.05	0.08
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[6]:
#drop unnecessary data

```
#drop unnecessary data
price_index.dropna(how='all', inplace=True)
price_index
```

[6]:

	date	iso_a3	currency_code	name	local_price	dollar_ex	dollar_price	GDP_dollar	adj_price	USD	EUR	GBP	JPY	CNY
1	1/07/2011	ARG	ARS	Argentina	20.00	4.1325	4.839	9138.	3.14	1.01	0.47	0.84	0.90	0.94
3	1/07/2011	AUS	AUD	Australia	4.56	0.9223	4.943	5558	5.79	0.11	-0.1	0.02	0.05	0.08
5	1/07/2011	BRA	BRL	Brazil	9.50	1.5416	6.162	1081	3.24	1.48	0.82	1.28	1.35	1.40
7	1/07/2011	GBR	GBP	Britain	2.39	0.6141	3.891	3611	4.68	0.08	-0.2	0.00	0.03	0.05
9	1/07/2011	CAN	CAD	Canada	4.73	0.9458	5.000	4621	5.25	0.24	-0.0	0.14	0.17	0.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

15	1/01/2021	UKR	UAH	Ukraine	62.00	28.140	2.203	3706.	3.01	-0.2	-0.3	-0.2	-0.1	-0.3
15	1/01/2021	ARE	AED	United Arab Emirates	14.75	3.6731	4.015	3917	4.44	-0.1	-0.2	-0.0	0.08	-0.1
15	1/01/2021	USA	USD	United States	5.66	1.0000	5.660	6525	5.49	0.00	-0.1	0.06	0.23	-0.0
15	1/01/2021	URY	UYU	Uruguay	204.0	42.495	4.800	1611	3.51	0.32	0.17	0.40	0.63	0.29
15	1/01/2021	VNM	VND	Vietnam	6600	23064.	2.861	3416.	3.00	-0.0	-0.1	-0.0	0.14	-0.0

757 rows × 14 columns

[7]:

```
price_index = price_index.drop_duplicates(['iso_a3'],
keep='last')
```

2.4 COVID information

in this section, i collect all the necessary information for COVID risk. this is absolutely one of the most hottest topic recently. from this data, i will find which country have more risk or not.

[8]:

```
!git clone https://github.com/CSSEGISandData/COVID-19.git
```

```
fatal: destination path 'COVID-19' already exists and is not
an empty directory.
```

[9]:
#read data

```
#read data
df_time_confirmed =
pd.read_csv('time_series_covid19_confirmed_global.csv')
```

[10]:
#remove unnecessary information

```
#remove unnecessary information
```

```
df_time_confirmed.dropna(how='all', inplace=True)
df_time_confirmed
```

[10]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	1/31/21	2/01/2021	2/02/2021	2/03/2021	2/04/2021	2/05/2021	2/06/2021	2/07/2021	2/08/2021	2/09/2021
1	NaN	Afghanistan	33.	67.	0.0	0.0	0.0	0.0	0.0	0.0	...	55	550	551	551	552	552	553	553	553	553
3	NaN	Albania	41.	20.	0.0	0.0	0.0	0.0	0.0	0.0	...	78	789	799	809	819	830	842	853	862	875
5	NaN	Algeria	28.	1.6	0.0	0.0	0.0	0.0	0.0	0.0	...	10	107	107	108	108	108	108	109	109	109
7	NaN	Andorra	42.	1.5	0.0	0.0	0.0	0.0	0.0	0.0	...	99	997	100	100	101	101	102	102	102	103
9	NaN	Angola	-11	17.	0.0	0.0	0.0	0.0	0.0	0.0	...	19	198	199	199	199	200	200	200	201	201
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5	NaN	Vietnam	14.	108	0.0	2.0	2.0	2.0	2.0	2.0	...	18	185	188	194	195	197	198	200	205	206
5	NaN	West Bank and Gaza	31.	35.	0.0	0.0	0.0	0.0	0.0	0.0	...	15	159	159	160	161	161	162	162	163	163

5	NaN	Yemen	15.	48.	0.(	0.(	0.(	0.(	0.(	0.(	...	21	212	212	212	212	212	212	212	213	213
5	NaN	Zambia	-13	27.	0.(	0.(	0.(	0.(	0.(	0.(	...	54	550	562	574	590	604	614	626	635	646
5	NaN	Zimbabwe	-19	29.	0.(	0.(	0.(	0.(	0.(	0.(	...	33	335	338	339	341	343	344	345	346	347

273 rows × 389 columns

[11]:

```
df_time_confirmed.columns
```

[11]:

```
Index(['Province/State', 'Country/Region', 'Lat', 'Long',
'1/22/20', '1/23/20',
       '1/24/20', '1/25/20', '1/26/20', '1/27/20',
       ...
       '1/31/21', '2/01/2021', '2/02/2021', '2/03/2021',
'2/04/2021',
       '2/05/2021', '2/06/2021', '2/07/2021', '2/08/2021',
'2/09/2021'],
      dtype='object', length=389)
```

[12]:

y axis

```
#exchange the x axis and y axis
df_time_confirmed.T
```

[12]:

	1	3	5	7	9	11	13	15	17	19	...	527	529	531	53	535	53	53	54	54	545
Province/State	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Australian Capital Territory	New South Wales	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Country/Region	Afghanistan	Albania	Algeria	Andorra	Angola	Antigua and Barbuda	Argentina	Armenia	Australia	Australia	...	United Kingdom	Uruguay	Uzbekistan	Vanuatu	Venezuela	Vietnam	West Bank and Gaza	Yemen	Zambia	Zimbabwe
Lat	33.9	41	28	42	-1	17	-38.	40	-35	-33	...	55.3	-32	41.	-1	6.4	14	31	15	-1	-19
Long	67.7	20	1.6	1.5	17	-61	-63.	45	149	15	...	-3.4	-55	64.	16	-66	10	35	48	27	29.
1/22/20	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2/05/2021	552	83	10	10	20	27	1.97	16	118	51	...	3.91	44	789	1	129	19	16	21	60	343
2/06/2021	553	84	10	10	20	28	1.97	16	118	51	...	3.92	44	790	1	129	19	16	21	61	344
2/07/2021	553	85	10	10	20	29	1.98	16	118	51	...	3.94	45	790	1	130	20	16	21	62	345
2/08/2021	553	86	10	10	20	31	1.98	16	118	51	...	3.95	45	791	1	130	20	16	21	63	346
2/09/2021	553	87	10	10	20	31	1.99	16	118	51	...	3.97	46	792	1	131	20	16	21	64	347

389 rows × 273 columns

[13]:

```
#make a group
df_time_confirmed_sum =
df_time_confirmed.drop(columns=['Province/State', 'Lat',
'Long']).groupby('Country/Region').mean().T
```

[14]:

```
df_time_confirmed_sum.columns
```

[14]:

```
Index(['Afghanistan', 'Albania', 'Algeria', 'Andorra',
'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia',
'Australia', 'Austria',
       ...
       'United Kingdom', 'Uruguay', 'Uzbekistan', 'Vanuatu',
'Venezuela',
       'Vietnam', 'West Bank and Gaza', 'Yemen', 'Zambia',
'Zimbabwe'],
      dtype='object', name='Country/Region', length=192)
```

[15]:

```
y code list
```

```
#read the country code list
df_region = pd.read_csv('country_region_list.csv',  encoding=
'unicode_escape')
```

[16]:

```
df_region
```

[16]:

	Country	Region	Global South
0	Andorra	Europe	Global North

1	United Arab Emirates	Middle east	Global South
2	Afghanistan	Asia & Pacific	Global South
3	Antigua and Barbuda	South/Latin America	Global South
4	Anguilla	South/Latin America	Global South
...	...	...	...
243	Guernsey	Europe	Global North
244	Isle of Man	Europe	Global North
245	Jersey	Europe	Global North
246	Saint Barthelemy	South/Latin America	Global South
247	Saint Martin	South/Latin America	Global South

248 rows × 3 columns

```
[17]:
y
```

```
#check if imported file is working correctly
pycountry.countries.get(name='Japan').alpha_3
```

```
[17]:
'JPN'
```

```
[18]:
```

```
pycountry.countries.get(name='Japan').alpha_2
```

```
[18]:
```

```
'JP'
```

[19]:
list up all missing countries

```python
#list up all missing countries
list_country_code = []
list_country_ = []


for i in list(df_time_confirmed_sum.columns):
    try:

list_country_code.append(pycountry.countries.get(name=i).alpha_3)
        list_country_.append(i)
    except:
        print(i)
```

```
Bolivia
Brunei
Burma
Congo (Brazzaville)
Congo (Kinshasa)
Cote d'Ivoire
Diamond Princess
Holy See
Iran
Korea, South
Kosovo
Laos
MS Zaandam
Micronesia
Moldova
Russia
Syria
Taiwan*
Tanzania
US
Venezuela
Vietnam
```

```
West Bank and Gaza
```

[20]:
y code

```
#list up all countries which successfully passed the country 
code
dict(zip(list_country_, list_country_code))
```

[20]:

```
{'Afghanistan': 'AFG',
 'Albania': 'ALB',
 'Algeria': 'DZA',
 'Andorra': 'AND',
 'Angola': 'AGO',
 'Antigua and Barbuda': 'ATG',
 'Argentina': 'ARG',
 'Armenia': 'ARM',
 'Australia': 'AUS',
 'Austria': 'AUT',
 'Azerbaijan': 'AZE',
 'Bahamas': 'BHS',
 'Bahrain': 'BHR',
 'Bangladesh': 'BGD',
 'Barbados': 'BRB',
 'Belarus': 'BLR',
 'Belgium': 'BEL',
 'Belize': 'BLZ',
 'Benin': 'BEN',
 'Bhutan': 'BTN',
 'Bosnia and Herzegovina': 'BIH',
 'Botswana': 'BWA',
 'Brazil': 'BRA',
 'Bulgaria': 'BGR',
 'Burkina Faso': 'BFA',
 'Burundi': 'BDI',
 'Cabo Verde': 'CPV',
 'Cambodia': 'KHM',
 'Cameroon': 'CMR',
 'Canada': 'CAN',
 'Central African Republic': 'CAF',
 'Chad': 'TCD',
 'Chile': 'CHL',
 'China': 'CHN',
 'Colombia': 'COL',
 'Comoros': 'COM',
 'Costa Rica': 'CRI',
 'Croatia': 'HRV',
 'Cuba': 'CUB',
 'Cyprus': 'CYP',
 'Czechia': 'CZE',
 'Denmark': 'DNK',
 'Djibouti': 'DJI',
 'Dominica': 'DMA',
 'Dominican Republic': 'DOM',
 'Ecuador': 'ECU',
 'Egypt': 'EGY',
 'El Salvador': 'SLV',
 'Equatorial Guinea': 'GNQ',
 'Eritrea': 'ERI',
 'Estonia': 'EST',
 'Eswatini': 'SWZ',
 'Ethiopia': 'ETH',
 'Fiji': 'FJI',
 'Finland': 'FIN',
 'France': 'FRA',
 'Gabon': 'GAB',
 'Gambia': 'GMB',
 'Georgia': 'GEO',
 'Germany': 'DEU',
 'Ghana': 'GHA',
 'Greece': 'GRC',
 'Grenada': 'GRD',
 'Guatemala': 'GTM',
 'Guinea': 'GIN',
 'Guinea-Bissau': 'GNB',
 'Guyana': 'GUY',
 'Haiti': 'HTI',
 'Honduras': 'HND',
 'Hungary': 'HUN',
 'Iceland': 'ISL',
 'India': 'IND',
 'Indonesia': 'IDN',
 'Iraq': 'IRQ',
 'Ireland': 'IRL',
 'Israel': 'ISR',
 'Italy': 'ITA',
 'Jamaica': 'JAM',
 'Japan': 'JPN',
 'Jordan': 'JOR',
 'Kazakhstan': 'KAZ',
 'Kenya': 'KEN',
 'Kuwait': 'KWT',
 'Kyrgyzstan': 'KGZ',
 'Latvia': 'LVA',
 'Lebanon': 'LBN',
 'Lesotho': 'LSO',
 'Liberia': 'LBR',
 'Libya': 'LBY',
 'Liechtenstein': 'LIE',
 'Lithuania': 'LTU',
 'Luxembourg': 'LUX',
 'Madagascar': 'MDG',
 'Malawi': 'MWI',
 'Malaysia': 'MYS',
 'Maldives': 'MDV',
 'Mali': 'MLI',
 'Malta': 'MLT',
 'Marshall Islands': 'MHL',
 'Mauritania': 'MRT',
 'Mauritius': 'MUS',
 'Mexico': 'MEX',
 'Monaco': 'MCO',
 'Mongolia': 'MNG',
 'Montenegro': 'MNE',
 'Morocco': 'MAR',
 'Mozambique': 'MOZ',
 'Namibia': 'NAM',
 'Nepal': 'NPL',
 'Netherlands': 'NLD',
 'New Zealand': 'NZL',
 'Nicaragua': 'NIC',
 'Niger': 'NER',
 'Nigeria': 'NGA',
 'North Macedonia': 'MKD',
 'Norway': 'NOR',
 'Oman': 'OMN',
 'Pakistan': 'PAK',
 'Panama': 'PAN',
 'Papua New Guinea': 'PNG',
 'Paraguay': 'PRY',
 'Peru': 'PER',
 'Philippines': 'PHL',
 'Poland': 'POL',
 'Portugal': 'PRT',
 'Qatar': 'QAT',
 'Romania': 'ROU',
 'Rwanda': 'RWA',
 'Saint Kitts and Nevis': 'KNA',
 'Saint Lucia': 'LCA',
 'Saint Vincent and the Grenadines': 'VCT',
 'Samoa': 'WSM',
 'San Marino': 'SMR',
 'Sao Tome and Principe': 'STP',
 'Saudi Arabia': 'SAU',
 'Senegal': 'SEN',
 'Serbia': 'SRB',
 'Seychelles': 'SYC',
 'Sierra Leone': 'SLE',
 'Singapore': 'SGP',
 'Slovakia': 'SVK',
 'Slovenia': 'SVN',
 'Solomon Islands': 'SLB',
 'Somalia': 'SOM',
 'South Africa': 'ZAF',
 'South Sudan': 'SSD',
 'Spain': 'ESP',
 'Sri Lanka': 'LKA',
 'Sudan': 'SDN',
 'Suriname': 'SUR',
 'Sweden': 'SWE',
 'Switzerland': 'CHE',
 'Tajikistan': 'TJK',
 'Thailand': 'THA',
 'Timor-Leste': 'TLS',
 'Togo': 'TGO',
 'Trinidad and Tobago': 'TTO',
 'Tunisia': 'TUN',
 'Turkey': 'TUR',
 'Uganda': 'UGA',
 'Ukraine': 'UKR',
 'United Arab Emirates': 'ARE',
 'United Kingdom': 'GBR',
 'Uruguay': 'URY',
 'Uzbekistan': 'UZB',
 'Vanuatu': 'VUT',
 'Yemen': 'YEM',
 'Zambia': 'ZMB',
 'Zimbabwe': 'ZWE'}
```

```
[21]:
```

```
dict_country_code = dict(zip(list_country_,
list_country_code))
```

[22]:

```
len(dict_country_code)
```

[22]:
169

[23]:
code

```
#manual adopt the code
dict_country_code.update(
{"Bolivia": "BOL",
"Brunei": "BRN",
"Burma": "MMR",
"Congo (Brazzaville)": "COG",
"Congo (Kinshasa)": "COG",
"Cote d'Ivoire": "CIV",
"Holy See": "VAT",
"Iran": "IRN",
"Korea, South": "PRK",
"Laos": "LAO",
"Moldova": "MDA",
"Russia": "RUS",
"Syria": "SYR",
"Taiwan*": "TWN",
"Tanzania": "TZA",
"US": "USA",
"Venezuela": "VEN",
"Vietnam": "VNM"}
)
```

[25]:

```
#drop the unnecessry item
df_time_confirmed_sum =
df_time_confirmed_sum.drop(columns=["Diamond Princess",
"Kosovo", "MS Zaandam", "West Bank and Gaza","Micronesia"])
```

```
---------------------------------------------------------------
--------------
KeyError                                  Traceback (most
recent call last)
<ipython-input-25-7e963a89e579> in <module>
      1 #drop the unnecessry item
----> 2 df_time_confirmed_sum =
df_time_confirmed_sum.drop(columns=["Diamond Princess",
"Kosovo", "MS Zaandam", "West Bank and Gaza","Micronesia"])

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/
frame.py in drop(self, labels, axis, index, columns, level,
inplace, errors)
   4172             level=level,
   4173             inplace=inplace,
-> 4174             errors=errors,
   4175         )
   4176

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/
generic.py in drop(self, labels, axis, index, columns, level,
inplace, errors)
   3887         for axis, labels in axes.items():
```

```
   3888                 if labels is not None:
-> 3889                     obj = obj._drop_axis(labels, axis,
level=level, errors=errors)
   3890
   3891             if inplace:

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/
generic.py in _drop_axis(self, labels, axis, level, errors)
   3921                     new_axis = axis.drop(labels,
level=level, errors=errors)
   3922                 else:
-> 3923                     new_axis = axis.drop(labels,
errors=errors)
   3924                 result = self.reindex(**{axis_name:
new_axis})
   3925

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/
indexes/base.py in drop(self, labels, errors)
   5285             if mask.any():
   5286                 if errors != "ignore":
-> 5287                     raise KeyError(f"{labels[mask]} not
found in axis")
   5288                 indexer = indexer[~mask]
   5289             return self.delete(indexer)

KeyError: "['Diamond Princess' 'Kosovo' 'MS Zaandam' 'West
Bank and Gaza'\n 'Micronesia'] not found in axis"
```

[26]:

```
list_country_code_columns = []
for i in list(df_time_confirmed_sum.columns):
    list_country_code_columns.append(dict_country_code[i])
```

[27]:

```
df_time_confirmed_sum.columns = list_country_code_columns
```

[28]:

```
 if the conversion is succeeded or not
```

```
#check if the conversion is succeeded or not
df_time_confirmed_sum.head()
```

[28]:

	AFG	ALB	DZA	AND	AGO	ATG	ARG	ARM	AUS	AUT	...	ARE	GBR	URY	UZB	VUT	VEN	VNM	YEM	ZMB	ZWE
1/22/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/23/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
1/24/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
1/25/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
1/26/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0

5 rows × 187 columns

4. Results and Discussion

in this section, by using all the data prepared in previous section, i will see the result. i will see the result seperately, price index and COVID

4.1 price index

this section, i will see the result of price index. at the end i will see the result in map style. the more expensive country will have darker coler and cheaper countries with lighter color. because of input data, some countries information was not able to collect. therefore those countries will be colored in white.

```
[29]:
#preparea a map
```

```
#preparea a map
price_map = folium.Map(location=[40, 10], zoom_start=2)
geojson = r'world_geo.json'
```

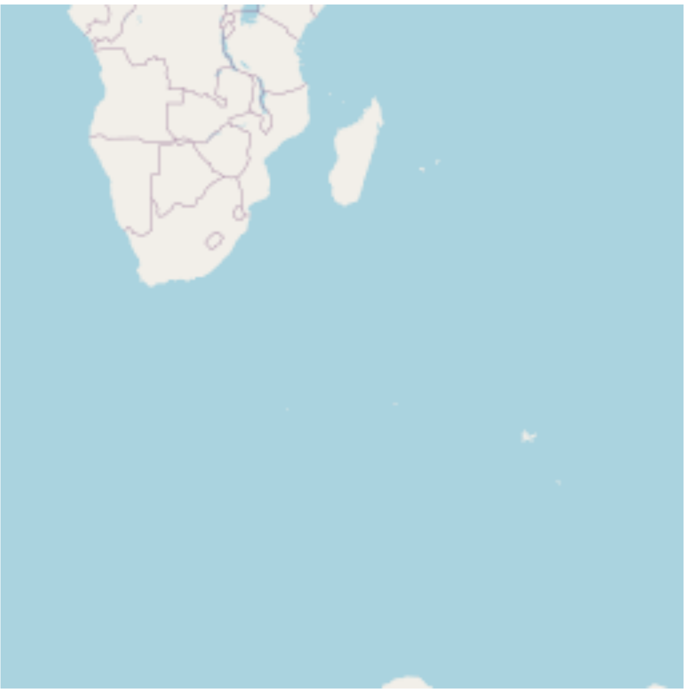
```
[31]:
```

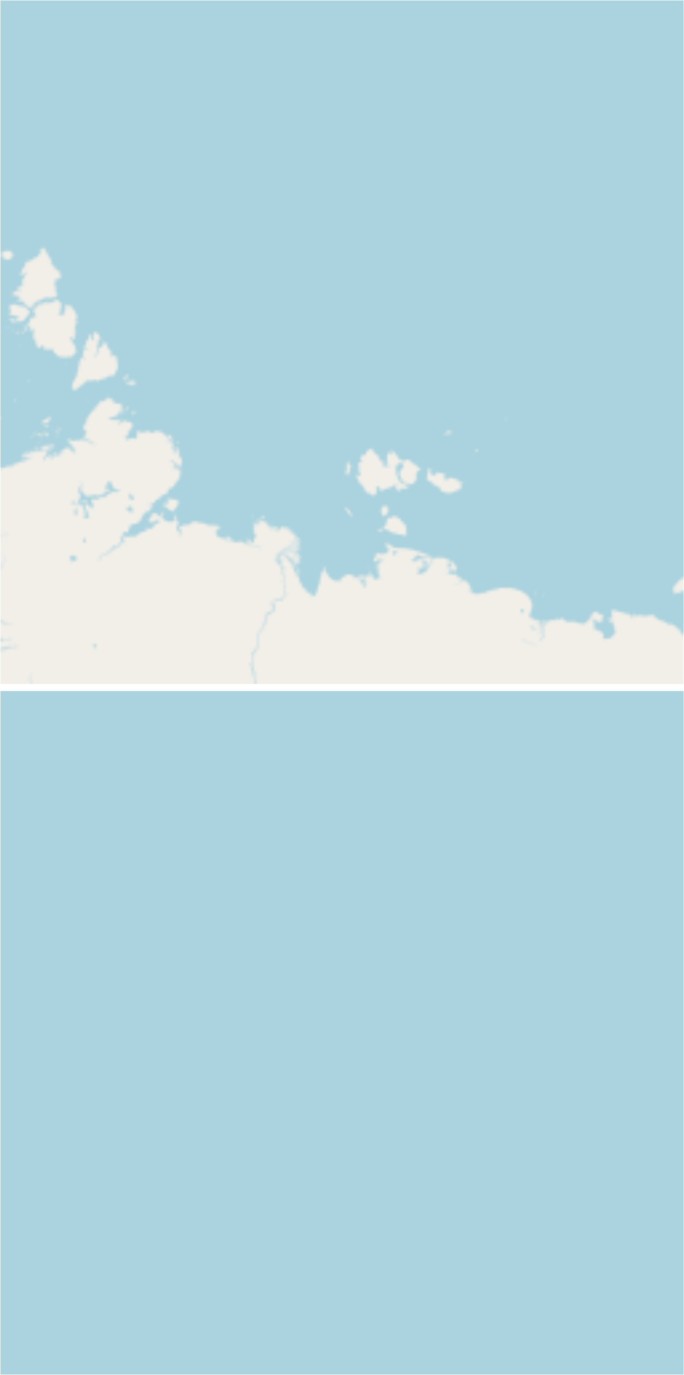
```
_index
```

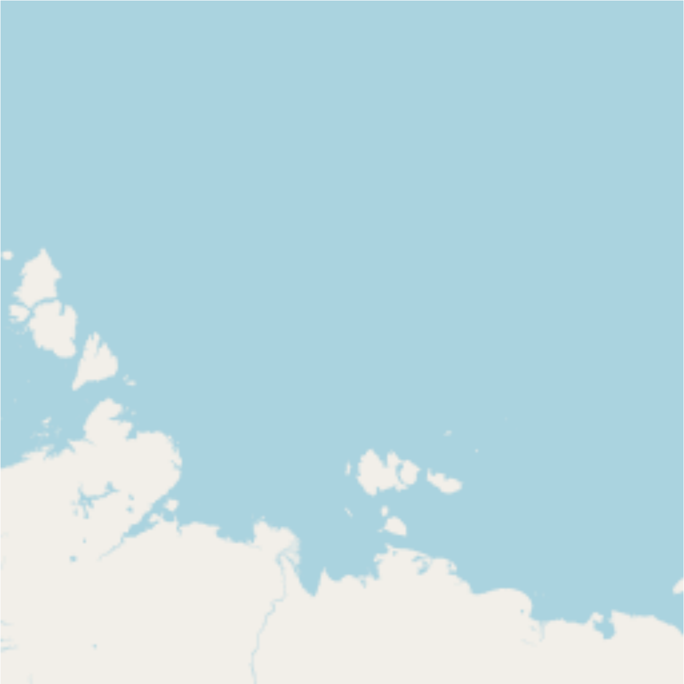
```
#add information
price_map.choropleth(
geo_data=geojson,
name='choropleth',
data=price_index,
columns=['iso_a3', 'dollar_price'],
key_on='feature.id',
fill_color='OrRd',
fill_opacity=0.7,
line_opacity=1,
legend_name='big mac index dollar_price'
)

price_map
```

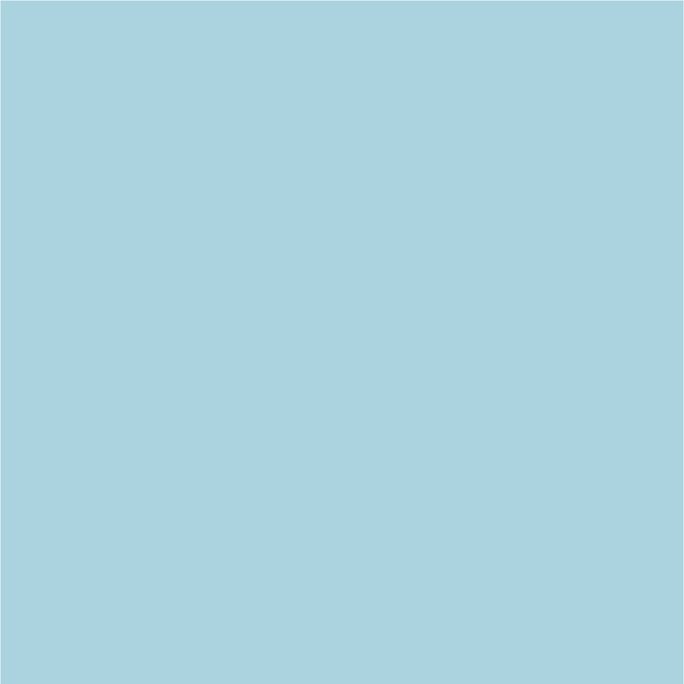
[31]:

Make this Notebook Trusted to load map: File -> Trust Notebook









+

–

1.8
2.7
3.6
4.5
5.5
6.4
7.3
big mac index dollar_price

Leaflet

4.2 COVID result

in this section, by using the all the data in previous section, i will see the result of COVID number. the higher number shows the higher risk.

[32]:

```
df_time_confirmed_sum = df_time_confirmed_sum.round()
```

[33]:

```
df_time_confirmed_sum.max()
```

[33]:

```
AFG     55384.0
ALB     87528.0
DZA    109559.0
AND     10312.0
AGO     20163.0
         ...
VEN    131096.0
VNM      2064.0
YEM      2131.0
ZMB     64610.0
ZWE     34781.0
Length: 187, dtype: float64
```

[34]:

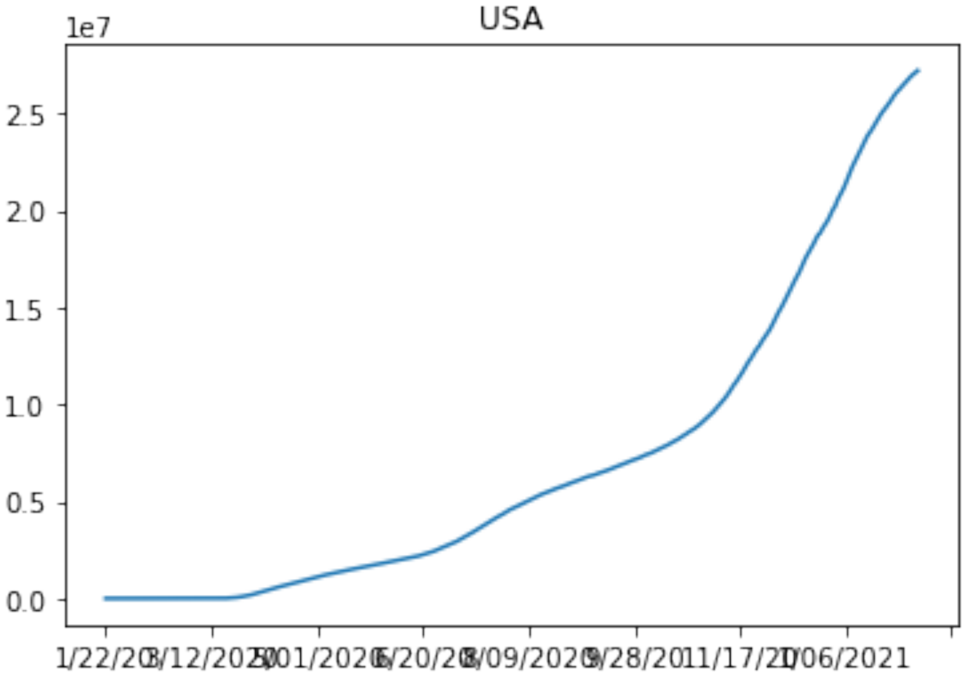
```
to see the highest number
```

```
#sort the list in order to see the highest number
df_time_confirmed_sum.max().sort_values(ascending=False)
```

```
[34]:
USA    27192455.0
IND    10858371.0
BRA     9599565.0
RUS     3953970.0
ESP     3005487.0
          ...
VAT          27.0
SLB          17.0
MHL           4.0
WSM           2.0
VUT           1.0
Length: 187, dtype: float64
```

```
[35]:
)
```

```
#plotting the USA's result.(USA have the highest number of 
COVID)
country = "USA"
df_time_confirmed_sum[country].plot()
plt.title(country)
plt.show()
```



[36]:

```
#list up top10 countries
```

```
#list up top10 countries
list_top10_country = list(df_time_confirmed_sum.max().sort_values(ascending=False)[0:10].index)
df_time_confirmed_sum[list_top10_country]
```

[36]:

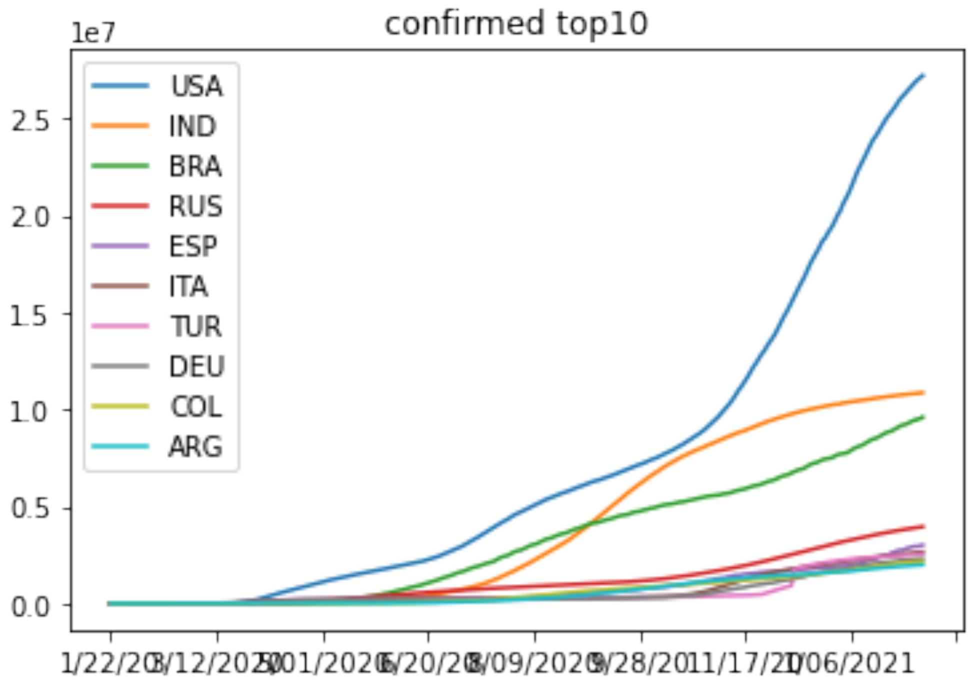
	USA	IND	BRA	RUS	ESP	ITA	TUR	DEU	COL	ARG
1/22/20	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/23/20	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/24/20	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/25/20	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1/26/20	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
2/05/2021	268137	108143	944716	389127	294199	261165	251688	227637	214266	197000
2/06/2021	269177	108263	944716	390765	294199	262509	252478	228500	215120	197668
2/07/2021	270073	108381	952464	392346	294199	263673	253145	229167	215721	198034
2/08/2021	270970	108473	952464	393916	298908	264470	253955	229632	216146	198550
2/09/2021	271924	108583	959956	395397	300548	265531	254819	230205	216690	199329

385 rows × 10 columns

```
[37]:
plot all those top10 countries in the same figure to compare
```

```
#plot all those top10 countries in the same figure to compare
df_time_confirmed_sum[list_top10_country].plot()
plt.title("confirmed top10")
plt.show()
```



```
[38]:
#list up bottom10, which means less COVID number
```

```
#list up bottom10, which means less COVID number
list_bottom10_country =
list(df_time_confirmed_sum.max().sort_values(ascending=True)
[0:10].index)
df_time_confirmed_sum[list_bottom10_country]
```

[38]:

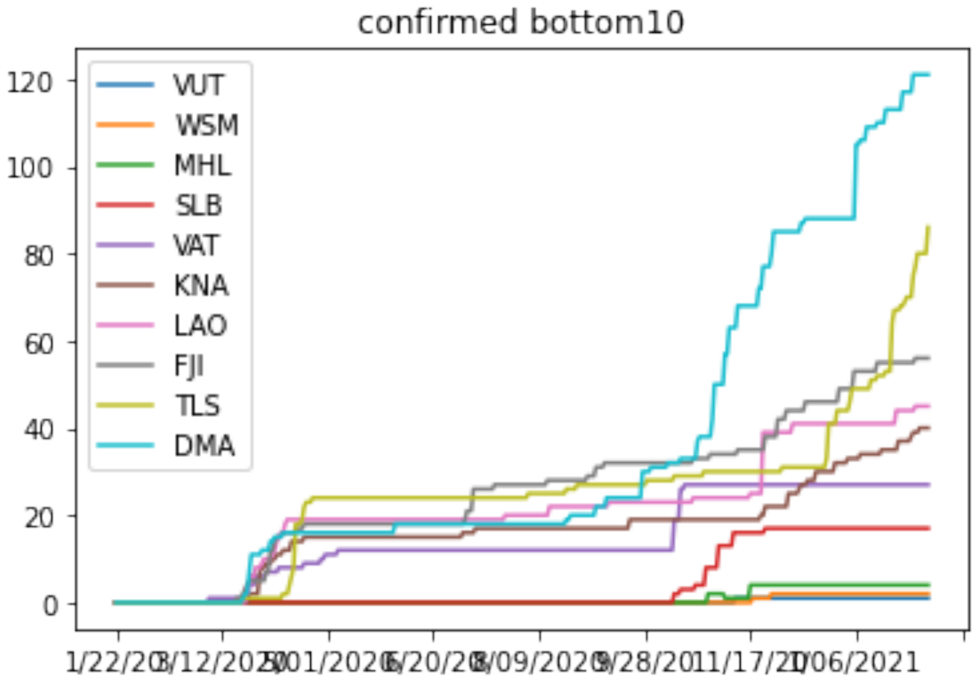
	VUT	WSM	MHL	SLB	VAT	KNA	LAO	FJI	TLS	DMA
1/22/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/23/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/24/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/25/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1/26/20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...

2/05/2021	1.0	2.0	4.0	17.0	27.0	40.0	45.0	56.0	80.0	121.0
2/06/2021	1.0	2.0	4.0	17.0	27.0	40.0	45.0	56.0	80.0	121.0
2/07/2021	1.0	2.0	4.0	17.0	27.0	40.0	45.0	56.0	80.0	121.0
2/08/2021	1.0	2.0	4.0	17.0	27.0	40.0	45.0	56.0	80.0	121.0
2/09/2021	1.0	2.0	4.0	17.0	27.0	40.0	45.0	56.0	86.0	121.0

385 rows × 10 columns

```
[102]:
a same figure to compare
```

```
#plot the bottom10 countries in a same figure to compare
df_time_confirmed_sum[list_bottom10_country].plot()
plt.title("confirmed bottom10")
plt.show()
```



[103]:

```
#list up top10
```

```
#list up top10
df_time_confirmed_sum[list_top10_country][-1:]
```

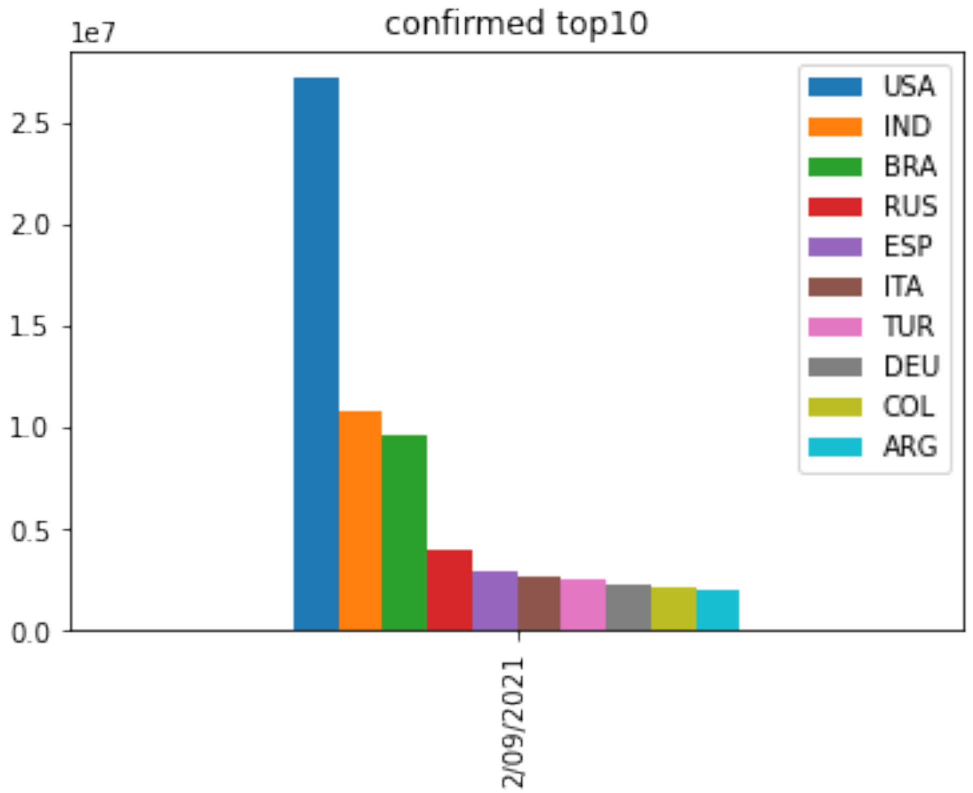
[103]:

	USA	IND	BRA	RUS	ESP	ITA	TUR	DEU	COL	ARG
2/09/ 2021	271924	108583	959956	395397	300548	265531	254819	230205	216690	199329

[104]:

```
plot in different method. histogram
```

```
#plot in different method. histogram
df_time_confirmed_sum[list_top10_country][-1:].plot.bar()
plt.title("confirmed top10")
plt.show()
```



5. Conclusion

From the result of price index, i can see developed countries are showing relatively higher price. many countries are missing data, so this is where i can improve of this result. the result of COVID information, i can see the US, India, Brazil, Russia, Spain have more COVID number. on the other hand, Vanuatu, Samoa, Solomon Island are showing less COVID number. NZ is one the most successful country from pandemic wise. but NZ seems expensive to live. but most of these bottom-10 countries are not expensive but also less COVID number. these coutries are more island-ish places, and offers more nature to you. we can keep the distance and avoid physical contact very good. therefore, these island/countries can be one of best places to visit for my next holiday.

```
[ ]:
```

[ ]:

[ ]:

[ ]:

reference

geo data and big mac index https://github.com/johan/world.geo.json/blob/master/countries.geo.json?short_path=afdfc39