Notebook

December 11, 2017

1 Child Mortality and economical, geographical, religious feature of the countries

```
<head>
    <style>
    ul {
        list-style-type: square;
        margin: 0;
        padding: 0;
        overflow: hidden;
        background-color: #333333;
    }

    li {
        float: left;
    }

    li a {
        display: block;
        color: white;
        text-align: center;
        padding: 16px;
        text-decoration: none;
    }

    li a:hover {
        background-color: #111111;
    }
    </style>
</head>
<body>For this analysis four data sets will be used:

<ul>
  <li><a href="http://apps.who.int/gho/data/node.main.ghe1002015-by-cause?lang=en">CHILD MORTAI
  <li><a href="http://www.imf.org/external/pubs/ft/weo/2017/01/weodata/download.aspx">WORLD COU
  <li><a href="https://www.cia.gov/library/publications/the-world-factbook/fields/2122.html">CO
  <li><a href="http://download.geonames.org/export/dump/">GEOGRAPHICAL INFORMATION</a></li>
```


All this dataset have been prepared and are available in https://github.com/MassimoSchiappa/dat
In the same place the json file downloaded from this link https://raw.githubusercontent.com/dat
</body>

In [1]:
```
import pandas as pd
import numpy as np
import folium

# ***************************** WHO DATASET *********************************
# The following short abbreviations have been used to name the column of the
# dataset in a more convenient way with respect to the original names
# ***************************************************************************
#ALRI    Acute lower respiratory infections
#BABT    Birth asphyxia and birth trauma
#OCPNC   Other communicable, perinatal and nutritional conditions
#SOICN   Sepsis and other infectious conditions of the newborn
#CA      Congenital anomalies
#DD      Diarrhoeal diseases
#AIDS    HIV/AIDS
#INJ     Injuries
#MAL     Malaria
#MEA     Measles
#MEN     Meningitis/encephalitis
#OND     Other noncommunicable diseases
#PER     Pertussis
#PRE     Prematurity
#
#N0-27D  from 0 to 27 days of life
#N1-59M  from 1 to 59 months of life
#N0-4Y   from 0 to 4 years of life (N0-27D + N1-59M)
# ***************************************************************************

df_who = pd.read_csv('./mort_child_ds.csv', sep=';', encoding = "ISO-8859-1")
```

In [2]:
```
# The list of countries is saved in a file to find the matching names with geo dataset
# to add the GEO ISO codes to the who dataset
df_who_countries = df_who[['Country']].drop_duplicates(keep='first')

# Let's create a temp directory
![ -d "./tmp" ]; then rm -fr ./tmp fi;
!mkdir tmp

# ... and now we can write there the file
df_who_countries.to_csv('tmp/df_who_countries.txt', sep=';')
```

In [3]:
```
df_who.head(5)
```

Out[3]:

	Unnamed: 0	Country	Year	N0-27D-ALRI	N1-59M-ALRI	N0-4Y-ALRI	\
0	0	Afghanistan	2015	2341	16330	18671	

```
1           1  Afghanistan  2014         2432        17046       19477
2           2  Afghanistan  2013         2552        19552       22104
3           3  Afghanistan  2012         2685        20561       23247
4           4  Afghanistan  2011         2837        21327       24164

   N0-27D-BABT  N1-59M-BABT  N0-4Y-BABT  N0-27D-OCPNC    ...      N0-4Y-MEN  \
0         9730          606       10336          2196    ...           2367
1        10063          632       10695          2259    ...           2477
2        10511          624       11135          2342    ...           3132
3        11018          649       11667          2442    ...           3343
4        11630          674       12304          2567    ...           3497

   N0-27D-OND  N1-59M-OND  N0-4Y-OND  N0-27D-PER  N1-59M-PER  N0-4Y-PER  \
0          22        6018       6040          49        1055       1104
1          23        6161       6184          51        1103       1153
2          23        5959       5982          52        1150       1203
3          24        6082       6106          54        1206       1260
4          25        6294       6320          56        1247       1303

   N0-27D-PRE  N1-59M-PRE  N0-4Y-PRE
0       11323        2426      13749
1       11367        2528      13895
2       11568        2495      14063
3       11710        2595      14304
4       11672        2696      14368

[5 rows x 45 columns]
```

In [4]:
```
df_who.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'Country', 'Year', 'N0-27D-ALRI', 'N1-59M-ALRI',
       'N0-4Y-ALRI', 'N0-27D-BABT', 'N1-59M-BABT', 'N0-4Y-BABT',
       'N0-27D-OCPNC', 'N1-59M-OCPNC', 'N0-4Y-OCPNC', 'N0-27D-SOICN',
       'N1-59M-SOICN', 'N0-4Y-SOICN', 'N0-27D-CA', 'N1-59M-CA', 'N0-4Y-CA',
       'N0-27D-DD', 'N1-59M-DD', 'N0-4Y-DD', 'N0-27D-AIDS', 'N1-59M-AIDS',
       'N0-4Y-AIDS', 'N0-27D-INJ', 'N1-59M-INJ', 'N0-4Y-INJ', 'N0-27D-MAL',
       'N1-59M-MAL', 'N0-4Y-MAL', 'N0-27D-MEA', 'N1-59M-MEA', 'N0-4Y-MEA',
       'N0-27D-MEN', 'N1-59M-MEN', 'N0-4Y-MEN', 'N0-27D-OND', 'N1-59M-OND',
       'N0-4Y-OND', 'N0-27D-PER', 'N1-59M-PER', 'N0-4Y-PER', 'N0-27D-PRE',
       'N1-59M-PRE', 'N0-4Y-PRE'],
      dtype='object')
```

In [5]:
```
# Adding a column for the sum of each cause contribution for Child Mortality
df_who['TOT']= df_who['N0-4Y-ALRI']+df_who['N0-4Y-BABT']+df_who['N0-4Y-OCPNC']+df_who[
               df_who['N0-4Y-CA']+df_who['N0-4Y-DD']+df_who['N0-4Y-AIDS']+df_who['N0-4
               df_who['N0-4Y-MAL']+df_who['N0-4Y-MEA']+df_who['N0-4Y-MEN']+df_who['N0
               df_who['N0-4Y-PER']+df_who['N0-4Y-PRE']
```

In [6]:
```
# ************************** GEOGRAPHICAL INFORMATION **************************
```

```
df_geo_ds = pd.read_csv('./geo_ds.csv', sep=';', encoding = "ISO-8859-1")
```

In [7]:
```
df_geo_ds.head(10)
```

Out[7]:

	Unnamed: 0	ISO	ISO3	Country	Capital	\
0	0	AD	AND	Andorra	Andorra la Vella	
1	1	AE	ARE	United Arab Emirates	Abu Dhabi	
2	2	AF	AFG	Afghanistan	Kabul	
3	3	AG	ATG	Antigua and Barbuda	St. John's	
4	4	AI	AIA	Anguilla	The Valley	
5	5	AL	ALB	Albania	Tirana	
6	6	AM	ARM	Armenia	Yerevan	
7	7	AO	AGO	Angola	Luanda	
8	8	AQ	ATA	Antarctica	NaN	
9	9	AQ	ATA	Antarctica	NaN	

	Area(in sq km)	Population	Continent	CurrencyCode	CurrencyName	latitude	\
0	468.0	84000	EU	EUR	Euro	42,546245	
1	82880.0	4975593	AS	AED	Dirham	23,424076	
2	647500.0	29121286	AS	AFN	Afghani	33,93911	
3	443.0	86754	NaN	XCD	Dollar	17,060816	
4	102.0	13254	NaN	XCD	Dollar	18,220554	
5	28748.0	2986952	EU	ALL	Lek	41,153332	
6	29800.0	2968000	AS	AMD	Dram	40,069099	
7	1246700.0	13068161	AF	AOA	Kwanza	-11,202692	
8	14000000.0	0	AN	NaN	NaN	-75,250973	
9	14000000.0	0	AN	NaN	NaN	-75,250973	

	longitude	TimeZoneId	rawOffset (independant of DST)
0	1,601554	Europe/Andorra	1.0
1	53,847818	Asia/Dubai	4.0
2	67,709953	Asia/Kabul	4.5
3	-61,796428	America/Antigua	-4.0
4	-63,068615	America/Anguilla	-4.0
5	20,168331	Europe/Tirane	1.0
6	45,038189	Asia/Yerevan	4.0
7	17,873887	Africa/Luanda	1.0
8	-0,071389	Antarctica/Casey	11.0
9	-0,071389	Antarctica/Davis	7.0

In [8]:
```
df_geo_ds_codes = df_geo_ds[['ISO3','Country','latitude','longitude','Population']].co
```

In [9]:
```
# Creating a file with the GEO dataset list of countries' names (duplicates are presen
# the different time zones in the same countries)
df_geo_ds_codes = df_geo_ds_codes.drop_duplicates(keep='first')
df_geo_ds_codes.to_csv('tmp/df_geo_ds_codes.csv', sep=';')
```

In [10]:
```
df_geo_ds_codes.head(10)
```

Out[10]:

	ISO3	Country	latitude	longitude	Population
0	AND	Andorra	42,546245	1,601554	84000
1	ARE	United Arab Emirates	23,424076	53,847818	4975593
2	AFG	Afghanistan	33,93911	67,709953	29121286
3	ATG	Antigua and Barbuda	17,060816	-61,796428	86754
4	AIA	Anguilla	18,220554	-63,068615	13254
5	ALB	Albania	41,153332	20,168331	2986952
6	ARM	Armenia	40,069099	45,038189	2968000
7	AGO	Angola	-11,202692	17,873887	13068161
8	ATA	Antarctica	-75,250973	-0,071389	0
18	ARG	Argentina	-38,416097	-63,616672	41343201

In [11]:
```
# A file containing only the WHO Countries' names is created: columns with ';' separa
!awk 'BEGIN {FS=";"}; {print $2}' tmp/df_who_countries.txt > tmp/df_who_countries_name

# A file containing only the GEO Countries' names is created: column with ';' separat
!awk 'BEGIN {FS=";"}; {print $3}' tmp/df_geo_ds_codes.csv > tmp/df_geo_ds_codes_names

# A file with rows in df_geo_ds_codes_names.csv and not in df_who_countries_names.txt
# is created
!awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/df_geo_ds_codes_names.csv tmp/df_who_countri

# A file with rows in df_who_countries_names.txt and not in df_geo_ds_codes_names.csv
# is created
!awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/df_who_countries_names.txt tmp/df_geo_ds_co
```

In [12]:
```
!cat tmp/not_in_geo.txt
```

```
Bolivia (Plurinational State of)
Brunei Darussalam
CÃ΄te d'Ivoire
Cabo Verde
Congo
Democratic People's Republic of Korea
Iran (Islamic Republic of)
Lao People's Democratic Republic
Micronesia (Federated States of)
Republic of Korea
Republic of Moldova
Russian Federation
Syrian Arab Republic
The former Yugoslav republic of Macedonia
Timor-Leste
United Kingdom of Great Britain and Northern Ireland
United Republic of Tanzania
United States of America
Venezuela (Bolivarian Republic of)
Viet Nam
```

In [13]: !cat tmp/not_in_who.txt

```
Anguilla
Antarctica
American Samoa
Aruba
Aland Islands
Saint Barthelemy
Bermuda
Brunei
Bolivia
Bonaire, Saint Eustatius and Saba
Bouvet Island
Cocos Islands
Republic of the Congo
Ivory Coast
Cape Verde
Curacao
Christmas Island
Western Sahara
Falkland Islands
Micronesia
Faroe Islands
United Kingdom
French Guiana
Guernsey
Gibraltar
Greenland
Guadeloupe
South Georgia and the South Sandwich Islands
Guam
Hong Kong
Heard Island and McDonald Islands
Isle of Man
British Indian Ocean Territory
Iran
Jersey
North Korea
South Korea
Kosovo
Cayman Islands
Laos
Liechtenstein
Moldova
Saint Martin
Macedonia
Macao
Northern Mariana Islands
```

```
Martinique
Montserrat
New Caledonia
Norfolk Island
French Polynesia
Saint Pierre and Miquelon
Pitcairn
Puerto Rico
Palestinian Territory
Reunion
Russia
Saint Helena
Svalbard and Jan Mayen
Sint Maarten
Syria
Turks and Caicos Islands
French Southern Territories
Tokelau
East Timor
Taiwan
Tanzania
United States Minor Outlying Islands
United States
Vatican
Venezuela
British Virgin Islands
U.S. Virgin Islands
Vietnam
Wallis and Futuna
Mayotte
Serbia and Montenegro
Netherlands Antilles


<head>
    <style>
    ul {
        list-style-type: square;
        margin: 0;
        padding: 0;
        overflow: hidden;
        background-color: #333333;
    }

    li {
        float: left;
    }
```

```
    li a {
        display: block;
        color: white;
        text-align: center;
        padding: 16px;
        text-decoration: none;
    }

    li a:hover {
        background-color: #111111;
    }
    </style>
</head>
<body>
    Comparing the two list we can match some record and then replace the
    corresponding geo values in the who file
</body>
```

```
In [14]: df_who['Country'].replace('Brunei Darussalam','Brunei', inplace=True)
         df_who['Country'].replace('CÃ´te d''Ivoire','Ivory Coast', inplace=True)
         df_who['Country'].replace('Cabo Verde','Cape Verde', inplace=True)
         df_who['Country'].replace('Congo','Republic of the Congo', inplace=True)
         df_who['Country'].replace('Democratic People''s Republic of Korea','North Korea', inpl
         df_who['Country'].replace('Iran (Islamic Republic of)','Iran', inplace=True)
         df_who['Country'].replace('Lao People''s Democratic Republic','Laos', inplace=True)
         df_who['Country'].replace('Micronesia (Federated States of)','Micronesia', inplace=Tru
         df_who['Country'].replace('Republic of Korea','South Korea', inplace=True)
         df_who['Country'].replace('Republic of Moldova','Moldova', inplace=True)
         df_who['Country'].replace('Russian Federation','Russia', inplace=True)
         df_who['Country'].replace('Syrian Arab Republic','Syria', inplace=True)
         df_who['Country'].replace('The former Yugoslav republic of Macedonia','Macedonia', inp
         df_who['Country'].replace('Timor-Leste','East Timor', inplace=True)
         df_who['Country'].replace('United Kingdom of Great Britain and Northern Ireland','Unit
         df_who['Country'].replace('United Republic of Tanzania','Tanzania', inplace=True)
         df_who['Country'].replace('United States of America','United States', inplace=True)
         df_who['Country'].replace('Venezuela (Bolivarian Republic of)','Venezuela', inplace=Tr
         df_who['Country'].replace('Viet Nam','Vietnam', inplace=True)
```

```
In [15]: # Now it is possible to merge the geo and the who dataset to add the ISO code to who
         df_who_geo = pd.merge(df_who, df_geo_ds_codes, left_on='Country', right_on='Country',
```

```
In [16]: # The value of the 'Year' column must be a string to be trasposed in a column
         df_who_geo[['Year']] = df_who_geo[['Year']].astype(str)
         df_who_geo.head(5)
```

Out[16]:

	Unnamed: 0	Country	Year	NO-27D-ALRI	N1-59M-ALRI	NO-4Y-ALRI	\
0	0	Afghanistan	2015	2341	16330	18671	
1	1	Afghanistan	2014	2432	17046	19477	
2	2	Afghanistan	2013	2552	19552	22104	

```
3           3  Afghanistan  2012         2685        20561       23247
4           4  Afghanistan  2011         2837        21327       24164

   NO-27D-BABT  N1-59M-BABT  NO-4Y-BABT  NO-27D-OCPNC     ...      N1-59M-PER  \
0         9730          606       10336          2196     ...            1055
1        10063          632       10695          2259     ...            1103
2        10511          624       11135          2342     ...            1150
3        11018          649       11667          2442     ...            1206
4        11630          674       12304          2567     ...            1247

   NO-4Y-PER  NO-27D-PRE  N1-59M-PRE  NO-4Y-PRE     TOT  ISO3  latitude  \
0       1104       11323        2426      13749   93469   AFG  33,93911
1       1153       11367        2528      13895   96327   AFG  33,93911
2       1203       11568        2495      14063   99946   AFG  33,93911
3       1260       11710        2595      14304  107457   AFG  33,93911
4       1303       11672        2696      14368  111845   AFG  33,93911

   longitude  Population
0  67,709953    29121286
1  67,709953    29121286
2  67,709953    29121286
3  67,709953    29121286
4  67,709953    29121286

[5 rows x 50 columns]
```

In [17]:
```
# From who-geo dataset we are going to create a new dataset using Year values as colu
df_who_geo_years = df_who_geo.pivot_table(index='ISO3', columns='Year', values='TOT')
df_who_geo_years.reset_index(inplace = True)
```

In [18]:
```
df_who_geo_years.head(5)
```

Out[18]:
```
Year ISO3    2000    2001    2002    2003    2004    2005    2006    2007  \
0     AFG  128035  133243  127301  121622  122184  122868  123654  120218
1     AGO  153126  159124  164018  164447  166426  170073  173280  174407
2     ALB    1375    1250    1125    1015     913     820     736     661
3     AND       2       2       2       2       2       2       1       1
4     ARE     565     543     523     514     535     584     616     674

Year    2008    2009    2010    2011    2012    2013    2014    2015
0     118695  118890  115177  111845  107457   99946   96327   93469
1     175447  176306  175412  174518  173673  173311  172508  167290
2        596     557     538     538     564     592     610     616
3          1       1       1       1       1       1       1       1
4        732     770     774     798     780     741     703     660
```

In [19]:
```
df_who_geo_years.columns
```

```
Out[19]: Index(['ISO3', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007',
                '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015'],
               dtype='object', name='Year')

In [20]: df_who_geo_years.set_index('ISO3')
```

Out[20]:

Year	2000	2001	2002	2003	2004	2005	2006	2007	2008
ISO3									
AFG	128035	133243	127301	121622	122184	122868	123654	120218	118695
AGO	153126	159124	164018	164447	166426	170073	173280	174407	175447
ALB	1375	1250	1125	1015	913	820	736	661	596
AND	2	2	2	2	2	2	1	1	1
ARE	565	543	523	514	535	584	616	674	732
ARG	14023	13841	13757	13632	13368	12904	12390	11853	11422
ARM	1089	1052	1054	1071	1081	1075	1048	1002	945
ATG	24	25	23	23	20	19	16	17	15
AUS	1533	1494	1463	1471	1468	1457	1481	1494	1507
AUT	412	398	394	399	396	394	387	376	362
AZE	9555	8555	7831	7335	6968	6704	6490	6305	6188
BDI	43294	42980	40416	40033	39995	39861	39679	39260	39098
BEL	637	615	603	596	599	604	605	607	605
BEN	42350	43489	40735	40085	39457	39416	39111	39072	39232
BFA	100325	96468	96996	94842	89794	87203	84987	81578	77612
BGD	313118	297297	280211	265459	260383	243939	223323	194535	181863
BGR	1398	1332	1262	1183	1112	1057	1030	1019	1021
BHR	165	157	158	157	158	163	162	161	159
BHS	83	79	80	81	83	85	87	87	86
BIH	441	382	324	269	226	210	217	247	283
BLR	1322	1209	1113	1027	973	914	860	793	739
BLZ	193	187	182	177	170	161	157	153	149
BRA	121539	112257	102788	93409	84486	75907	68021	61178	55051
BRB	58	60	62	62	61	60	59	57	54
BRN	63	63	60	60	58	58	56	55	54
BTN	1278	1219	1087	998	921	858	810	765	727
BWA	4502	3861	3711	3609	3491	3181	2924	2942	2968
CAF	25353	25501	25412	24940	25372	24310	23223	22939	22641
CAN	2041	2005	2011	2014	2064	2122	2149	2180	2206
CHE	431	416	400	382	370	366	363	356	358
...	...	...	...	...	...	...	...	...	...
SWE	310	317	339	373	391	407	398	388	367
SWZ	4361	4485	4535	4553	4532	4481	4238	4091	4029
SYC	19	19	22	23	23	24	25	24	24
SYR	12042	11516	10779	10019	9297	8929	8680	8663	8522
TCD	72305	77728	79727	80615	84875	83225	83034	83200	83952
TGO	23734	23043	21506	21762	21835	21673	21704	21662	21618
THA	20360	19182	18139	17270	16371	15390	14476	13616	12755
TJK	17727	16243	14731	13351	12289	11618	11250	11192	11327
TKM	7644	7549	7764	8109	8339	8329	8073	7606	7020

TLS	4254	3919	3606	3304	3001	2709	2437	2196	2013
TON	46	46	47	46	47	48	48	48	48
TTO	528	530	533	538	540	539	532	520	505
TUN	5326	4952	4599	4264	4007	3780	3608	3470	3355
TUR	55724	51563	47334	43146	39252	36059	33110	30629	28464
TUV	12	11	11	9	8	7	7	6	6
TZA	172470	166522	161133	155796	150125	144153	137616	130384	123338
UGA	167348	161180	161790	153940	138899	133095	127234	121218	115269
UKR	7544	6954	6482	6171	5991	6010	6179	6415	6663
URY	891	849	823	793	768	737	707	678	657
USA	31515	31613	32070	32670	33574	33699	33518	32978	32170
UZB	34776	33238	31804	30574	29692	29212	29167	29409	29795
VCT	49	45	44	42	43	40	40	42	42
VEN	12578	12256	11975	11676	11362	10979	10651	10368	10195
VNM	44658	41678	41234	41272	41785	42315	42467	41605	40307
VUT	168	166	164	165	169	169	178	188	202
WSM	111	109	105	103	104	103	102	102	103
YEM	63988	58950	56970	57341	57040	54154	52564	48614	46511
ZAF	87736	89005	88359	85662	82342	79753	78656	79487	79261
ZMB	74787	71305	68933	62345	57952	55100	52982	50924	49461
ZWE	41415	41884	42297	42266	42755	43404	43686	43631	43793

Year	2009	2010	2011	2012	2013	2014	2015
ISO3							
AFG	118890	115177	111845	107457	99946	96327	93469
AGO	176306	175412	174518	173673	173311	172508	167290
ALB	557	538	538	564	592	610	616
AND	1	1	1	1	1	1	1
ARE	770	774	798	780	741	703	660
ARG	11018	10681	10422	10224	9973	9770	9513
ARM	888	830	779	724	671	620	576
ATG	15	15	14	13	13	13	12
AUS	1504	1480	1397	1322	1244	1171	1146
AUT	351	342	326	316	311	302	288
AZE	6180	6308	6591	6961	7278	7386	7189
BDI	38979	38642	38540	38213	37683	37199	36840
BEL	599	589	579	561	559	537	524
BEN	39301	38840	38614	38047	38122	37844	37005
BFA	91369	72924	68839	74831	63774	61684	60368
BGD	167487	156200	148234	137349	130120	123823	118912
BGR	1019	1001	950	861	772	690	635
BHR	156	151	154	151	146	142	138
BHS	88	85	82	81	75	73	70
BIH	309	311	295	265	227	194	170
BLR	692	659	628	617	604	579	568
BLZ	143	142	139	140	139	138	137
BRA	50454	47835	46872	47449	49090	51111	52414
BRB	51	49	48	47	45	46	44

```
BRN       52      55      58      62      69      73      77
BTN      686     636     586     534     484     444     414
BWA     2841    2771    2726    2676    2582    2518    2480
CAF    22105   21840   22376   21436   21974   21108   20754
CAN     2216    2201    2119    2055    1978    1871    1824
CHE      360     362     357     357     350     337     335
...      ...     ...     ...     ...     ...     ...     ...
SWE      348     345     332     334     335     339     344
SWZ     3720    3180    2866    2651    2472    2289    2219
SYC       24      23      25      23      23      23      23
SYR     8472    8030    7794    7995    7147    6310    6017
TCD    83588   83916   84566   83823   82752   82463   81744
TGO    21326   21160   20784   20608   20441   19798   19442
THA    12017   11420   10885   10397   10007    9486    9127
TJK    11567   11786   11951   12077   12124   12026   11772
TKM     6500    6162    5968    5928    5941    5947    5856
TLS     1921    1937    2050    2246    2459    2603    2632
TON       49      49      49      48      47      46      44
TTO      491     474     458     439     419     405     387
TUN     3248    3177    3115    3088    3064    3005    2899
TUR    26599   24984   23693   22468   21428   20134   18938
TUV        6       6       6       6       6       6       6
TZA   117273  110954  105955  103204  101788   99194   97990
UGA   110019  105221   99920   93734   90039   86797   85138
UKR     6715    6592    6211    5631    4971    4424    3996
URY      637     614     595     572     542     513     487
USA    30845   29941   28688   27790   26599   25628   24923
UZB    30040   30040   29730   29177   28351   27304   26150
VCT       41      41      38      38      35      31      30
VEN    10022    9905    9806    9645    9422    9190    8945
VNM    39645   38531   37348   36541   35898   36941   34078
VUT      210     216     216     212     203     195     187
WSM      132     104     100      95      89      84      80
YEM    44479   42388   40836   39036   36885   35471   34134
ZAF    76189   68220   61160   56296   48599   43751   41925
ZMB    47570   47754   46150   43473   40950   39596   38901
ZWE    44041   43847   43021   40252   38842   38393   37957

[190 rows x 16 columns]
```

```
In [21]: # ***************************** IMF ****************************************
         df_imf = pd.read_csv('./imf_weo_ds.csv',sep=';', encoding = "ISO-8859-1")

In [22]: df_imf.columns

Out[22]: Index(['WEO Country Code', 'ISO', 'WEO Subject Code', 'Country',
                'Subject Descriptor', 'Subject Notes', 'Units', 'Scale',
                'Country/Series-specific Notes', '1980', '1981', '1982', '1983', '1984',
```

```
       '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
       '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002',
       '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
       '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020',
       '2021', '2022', 'Estimates Start After'],
      dtype='object')
```

In [23]:
```
df_imf_filt = df_imf[['ISO','Country','Units','Scale','2000','2001', '2002',
       '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
       '2012', '2013', '2014', '2015','Estimates Start After','WEO Subject Code','Sub
```

In [24]:
```
# Let's see how this dataset is composed for a single country
df_imf_filt[df_imf_filt['Country']=='Italy'][['WEO Subject Code','Subject Descriptor'
```

Out[24]:

	WEO Subject Code	Subject Descriptor
3520	NGDP_R	Gross domestic product, constant prices
3521	NGDP_RPCH	Gross domestic product, constant prices
3522	NGDP	Gross domestic product, current prices
3523	NGDPD	Gross domestic product, current prices
3524	NGDP_D	Gross domestic product, deflator
3525	NGDPRPC	Gross domestic product per capita, constant pr...
3526	NGDPPC	Gross domestic product per capita, current prices
3527	NGDPDPC	Gross domestic product per capita, current prices
3528	NGAP_NPGDP	Output gap in percent of potential GDP
3529	PPPGDP	Gross domestic product based on purchasing-pow...
3530	PPPPC	Gross domestic product based on purchasing-pow...
3531	PPPSH	Gross domestic product based on purchasing-pow...
3532	PPPEX	Implied PPP conversion rate
3533	NID_NGDP	Total investment
3534	NGSD_NGDP	Gross national savings
3535	PCPI	Inflation, average consumer prices
3536	PCPIPCH	Inflation, average consumer prices
3537	PCPIE	Inflation, end of period consumer prices
3538	PCPIEPCH	Inflation, end of period consumer prices
3539	FLIBOR6	Six-month London interbank offered rate (LIBOR)
3540	TM_RPCH	Volume of imports of goods and services
3541	TMG_RPCH	Volume of Imports of goods
3542	TX_RPCH	Volume of exports of goods and services
3543	TXG_RPCH	Volume of exports of goods
3544	LUR	Unemployment rate
3545	LE	Employment
3546	LP	Population
3547	GGR	General government revenue
3548	GGR_NGDP	General government revenue
3549	GGX	General government total expenditure
3550	GGX_NGDP	General government total expenditure
3551	GGXCNL	General government net lending/borrowing
3552	GGXCNL_NGDP	General government net lending/borrowing

3553	GGSB	General government structural balance
3554	GGSB_NPGDP	General government structural balance
3555	GGXONLB	General government primary net lending/borrowing
3556	GGXONLB_NGDP	General government primary net lending/borrowing
3557	GGXWDN	General government net debt
3558	GGXWDN_NGDP	General government net debt
3559	GGXWDG	General government gross debt
3560	GGXWDG_NGDP	General government gross debt
3561	NGDP_FY	Gross domestic product corresponding to fiscal...
3562	BCA	Current account balance
3563	BCA_NGDPD	Current account balance

	Subject Notes
3520	Expressed in billions of national currency uni...
3521	Annual percentages of constant price GDP are y...
3522	Expressed in billions of national currency uni...
3523	Values are based upon GDP in national currency...
3524	The GDP deflator is derived by dividing curren...
3525	GDP is expressed in constant national currency...
3526	GDP is expressed in current national currency ...
3527	GDP is expressed in current U.S. dollars per p...
3528	Output gaps for advanced economies are calcula...
3529	These data form the basis for the country weig...
3530	Expressed in GDP in PPP dollars per person. Da...
3531	Expressed in percent of world GDP in PPP dolla...
3532	Expressed in national currency per current int...
3533	Expressed as a ratio of total investment in cu...
3534	Expressed as a ratio of gross national savings...
3535	Expressed in averages for the year, not end-of...
3536	Annual percentages of average consumer prices ...
3537	Expressed in end of the period, not annual ave...
3538	Annual percentages of end of period consumer ...
3539	NaN
3540	Percent change of volume of imports refers to ...
3541	Percent change of volume of imports of goods r...
3542	Percent change of volume of exports refers to ...
3543	Percent change of volume of exports of goods r...
3544	Unemployment rate can be defined by either the...
3545	Employment can be defined by either the nation...
3546	For census purposes, the total population of t...
3547	Revenue consists of taxes, social contribution...
3548	Revenue consists of taxes, social contribution...
3549	Total expenditure consists of total expense an...
3550	Total expenditure consists of total expense an...
3551	Net lending (+)/ borrowing (?) is calculated a...
3552	Net lending (+)/ borrowing (?) is calculated a...
3553	The structural budget balance refers to the ge...
3554	The structural budget balance refers to the ge...

```
3555  Primary net lending/borrowing is net lending (...
3556  Primary net lending/borrowing is net lending (...
3557  Net debt is calculated as gross debt minus fin...
3558  Net debt is calculated as gross debt minus fin...
3559  Gross debt consists of all liabilities that re...
3560  Gross debt consists of all liabilities that re...
3561  Gross domestic product corresponding to fiscal...
3562  Current account is all transactions other than...
3563  Current account is all transactions other than...
```

In [25]:
```
# Population
df_imf_pop = df_imf_filt[df_imf_filt['WEO Subject Code']=='LP']

# Employment
df_imf_empl = df_imf_filt[df_imf_filt['WEO Subject Code']=='LE']

# Unemployment rate
df_imf_unempl_rate = df_imf_filt[df_imf_filt['WEO Subject Code']=='LUR']

# GDP procapita
df_imf_gdp_pc = df_imf_filt[df_imf_filt['WEO Subject Code']=='NGDPDPC']

# GDP
df_imf_gdp = df_imf_filt[df_imf_filt['WEO Subject Code']=='NGDPD']

# GDP procapita based on purchasing-power-parity (PPP)
df_imf_gdp_xcppp_cp = df_imf_filt[df_imf_filt['WEO Subject Code']=='PPPPC']
```

In [26]:
```
# Let's create a new imf dataset for population along years from 2000 to 2015
df_imf_pop_years = df_imf_pop[['ISO','2000','2001', '2002','2003', '2004', '2005', '20
df_imf_pop_years.replace('n/a',0,inplace=True)
```

In [27]:
```
# Let's create a new imf dataset for gdp pro capita along years from 2000 to 2015
df_imf_gdp_pc_years = df_imf_gdp_pc[['ISO','2000','2001', '2002','2003', '2004', '2005
df_imf_gdp_pc_years.replace('n/a',0,inplace=True)
```

In [28]:
```
# We now merge the who-geo-years dataset with the imf-pop-years dataset using _pop an
# to distinguish the value of pop and who for the same year
df_who_pop_geo_years = pd.merge(df_who_geo_years, df_imf_pop_years, left_on='ISO3', r
df_who_pop_geo_years.head(5)
```

Out[28]:

	ISO3	2000_who	2001_who	2002_who	2003_who	2004_who	2005_who	2006_who	\
0	AFG	128035	133243	127301	121622	122184	122868	123654	
1	AGO	153126	159124	164018	164447	166426	170073	173280	
2	ALB	1375	1250	1125	1015	913	820	736	
3	ARE	565	543	523	514	535	584	616	
4	ARG	14023	13841	13757	13632	13368	12904	12390	

	2007_who	2008_who	...	2006_pop	2007_pop	2008_pop	2009_pop	\

```
0    120218    118695   ...        25631     26349     27032     27708
1    174407    175447   ...        20358     20969     21598     22246
2       661       596   ...         2993      2970      2947      2928
3       674       732   ...         5012      6219      8074      8200
4     11853     11422   ...        38971     39356     39746     40134

   2010_pop  2011_pop  2012_pop 2013_pop 2014_pop 2015_pop
0     28398     29105     29825    30550    31279    32007
1     22913     23601     24309    25038    25789    26563
2      2913      2905      2900     2897     2894     2889
3      8264      8512      8768     9031     9302     9581
4     40788     41261     41733    42203    42670    43132

[5 rows x 34 columns]
```

In [29]:
```
# And now we merge the who-geo-years dataset with the imf-gdp-xc-years dataset using
# to distinguish the value of pop and who for the same year
df_who_gdp_geo_years = pd.merge(df_who_geo_years, df_imf_gdp_pc_years, left_on='ISO3'
df_who_gdp_geo_years.head(5)
```

Out[29]:
```
  ISO3  2000_who  2001_who  2002_who  2003_who  2004_who  2005_who  2006_who  \
0  AFG    128035    133243    127301    121622    122184    122868    123654
1  AGO    153126    159124    164018    164447    166426    170073    173280
2  ALB      1375      1250      1125      1015       913       820       736
3  ARE       565       543       523       514       535       584       616
4  ARG     14023     13841     13757     13632     13368     12904     12390

   2007_who  2008_who      ...        2006_gdp    2007_gdp    2008_gdp  \
0    120218    118695      ...          270189      324705      380910
1    174407    175447      ...       2,052.721   2,882.797   3,897.512
2       661       596      ...       2,975.623   3,594.101   4,377.040
3       674       732      ...      44,313.586  41,472.293  39,074.838
4     11853     11422      ...       5,976.082   7,315.726   9,146.790

     2009_gdp    2010_gdp    2011_gdp    2012_gdp    2013_gdp    2014_gdp  \
0      435472      539667      614661      680500      660221      650663
1   3,393.552   3,599.272   4,411.575   4,744.884   4,988.923   4,915.923
2   4,130.931   4,098.125   4,439.559   4,249.039   4,413.283   4,574.800
3  30,920.447  34,628.629  40,943.563  42,591.437  43,030.321  43,213.255
4   8,337.811  10,412.945  12,787.806  13,889.792  14,488.829  13,208.832

     2015_gdp
0      615091
1   3,876.197
2   3,943.217
3  38,649.912
4  14,643.922

[5 rows x 34 columns]
```

```
In [30]: df_who_gdp_geo_years.columns
```

```
Out[30]: Index(['ISO3', '2000_who', '2001_who', '2002_who', '2003_who', '2004_who',
                '2005_who', '2006_who', '2007_who', '2008_who', '2009_who', '2010_who',
                '2011_who', '2012_who', '2013_who', '2014_who', '2015_who', 'ISO',
                '2000_gdp', '2001_gdp', '2002_gdp', '2003_gdp', '2004_gdp', '2005_gdp',
                '2006_gdp', '2007_gdp', '2008_gdp', '2009_gdp', '2010_gdp', '2011_gdp',
                '2012_gdp', '2013_gdp', '2014_gdp', '2015_gdp'],
               dtype='object')
```

```
In [31]: # Loop to clean the gdp values from the characters ',' and '.'
         for year in ['2000', '2001', '2002', '2003', '2004','2005', '2006', '2007', '2008', '2
                '2011', '2012', '2013', '2014', '2015']:
             df_who_gdp_geo_years[year+'_gdp'] = df_who_gdp_geo_years[year+'_gdp'].astype(np.st
             df_who_gdp_geo_years[year+'_gdp'] = df_who_gdp_geo_years[year+'_gdp'].astype(np.st

         df_who_gdp_geo_years.head(5)
```

Out[31]:

	ISO3	2000_who	2001_who	2002_who	2003_who	2004_who	2005_who	2006_who	\
0	AFG	128035	133243	127301	121622	122184	122868	123654	
1	AGO	153126	159124	164018	164447	166426	170073	173280	
2	ALB	1375	1250	1125	1015	913	820	736	
3	ARE	565	543	523	514	535	584	616	
4	ARG	14023	13841	13757	13632	13368	12904	12390	

	2007_who	2008_who	...	2006_gdp	2007_gdp	2008_gdp	2009_gdp	\
0	120218	118695	...	270189	324705	380910	435472	
1	174407	175447	...	2052721	2882797	3897512	3393552	
2	661	596	...	2975623	3594101	4377040	4130931	
3	674	732	...	44313586	41472293	39074838	30920447	
4	11853	11422	...	5976082	7315726	9146790	8337811	

	2010_gdp	2011_gdp	2012_gdp	2013_gdp	2014_gdp	2015_gdp
0	539667	614661	680500	660221	650663	615091
1	3599272	4411575	4744884	4988923	4915923	3876197
2	4098125	4439559	4249039	4413283	4574800	3943217
3	34628629	40943563	42591437	43030321	43213255	38649912
4	10412945	12787806	13889792	14488829	13208832	14643922

[5 rows x 34 columns]

```
In [32]: df_who_pop_geo_years.columns
```

```
Out[32]: Index(['ISO3', '2000_who', '2001_who', '2002_who', '2003_who', '2004_who',
                '2005_who', '2006_who', '2007_who', '2008_who', '2009_who', '2010_who',
                '2011_who', '2012_who', '2013_who', '2014_who', '2015_who', 'ISO',
                '2000_pop', '2001_pop', '2002_pop', '2003_pop', '2004_pop', '2005_pop',
                '2006_pop', '2007_pop', '2008_pop', '2009_pop', '2010_pop', '2011_pop',
                '2012_pop', '2013_pop', '2014_pop', '2015_pop'],
               dtype='object')
```

In [33]:
```
# Calculating the ratio between total who number and total population for each countr
# each year
for year in ['2000', '2001', '2002', '2003', '2004','2005', '2006', '2007', '2008', '
       '2011', '2012', '2013', '2014', '2015']:
    df_who_pop_geo_years[year+'_pop'] = df_who_pop_geo_years[year+'_pop'].astype(np.st
    df_who_pop_geo_years[year+'_pop'] = df_who_pop_geo_years[year+'_pop'].astype(np.st
    df_who_pop_geo_years[year+'_who']= df_who_pop_geo_years[year+'_who'].astype(float)

# Renaming columns to get the original names for the years
df_who_pop_geo_years.rename(columns={'2000_who':'2000', '2001_who':'2001', '2002_who'
       '2005_who':'2005', '2006_who':'2006', '2007_who':'2007', '2008_who':'2008', '20
       '2011_who':'2011', '2012_who':'2012', '2013_who':'2013', '2014_who':'2014', '20

# let's drop columns from imf (population) no more useful in this dataset
columns = ['2000_pop', '2001_pop', '2002_pop', '2003_pop', '2004_pop', '2005_pop',
       '2006_pop', '2007_pop', '2008_pop', '2009_pop', '2010_pop', '2011_pop',
       '2012_pop', '2013_pop', '2014_pop', '2015_pop', 'ISO']
df_who_pop_geo_years.drop(columns, inplace=True, axis=1)

# The value 1 means that the population for that country in that year was not present
df_who_pop_geo_years.replace(1,0, inplace=True)
df_who_pop_geo_years.head(200)
```

Out[33]:

	ISO3	2000	2001	2002	2003	2004	2005	\
0	AFG	0.000000	0.000000	0.005701	0.005234	0.005061	0.004918	
1	AGO	0.008901	0.008980	0.008986	0.008749	0.008598	0.008531	
2	ALB	0.000445	0.000408	0.000369	0.000334	0.000302	0.000272	
3	ARE	0.000189	0.000171	0.000156	0.000145	0.000142	0.000142	
4	ARG	0.000381	0.000372	0.000367	0.000360	0.000350	0.000334	
5	ARM	0.000353	0.000344	0.000345	0.000352	0.000357	0.000357	
6	ATG	0.000316	0.000325	0.000295	0.000291	0.000253	0.000237	
7	AUS	0.000080	0.000077	0.000075	0.000074	0.000073	0.000072	
8	AUT	0.000051	0.000049	0.000049	0.000049	0.000048	0.000048	
9	AZE	0.001173	0.001038	0.000939	0.000870	0.000817	0.000778	
10	BDI	0.006436	0.006205	0.005667	0.005484	0.005320	0.005278	
11	BEL	0.000062	0.000060	0.000058	0.000058	0.000058	0.000058	
12	BEN	0.006057	0.006025	0.005464	0.005202	0.004955	0.004794	
13	BFA	0.008575	0.008041	0.007824	0.007413	0.006860	0.006466	
14	BGD	0.002360	0.002202	0.002041	0.001904	0.001840	0.001701	
15	BGR	0.000172	0.000169	0.000161	0.000152	0.000143	0.000137	
16	BHR	0.000259	0.000237	0.000222	0.000205	0.000192	0.000183	
17	BHS	0.000274	0.000257	0.000256	0.000256	0.000259	0.000261	
18	BIH	0.000117	0.000101	0.000085	0.000070	0.000058	0.000054	
19	BLR	0.000132	0.000121	0.000112	0.000104	0.000100	0.000094	
20	BLZ	0.000771	0.000727	0.000686	0.000646	0.000600	0.000551	
21	BRA	0.000700	0.000638	0.000576	0.000517	0.000462	0.000410	
22	BRB	0.000216	0.000222	0.000229	0.000228	0.000223	0.000220	
23	BRN	0.000194	0.000189	0.000174	0.000171	0.000161	0.000157	

24	BTN	0.002261	0.002094	0.001814	0.001618	0.001451	0.001318
25	BWA	0.002554	0.002160	0.002048	0.001968	0.001881	0.001693
26	CAF	0.006921	0.006838	0.006701	0.006470	0.006473	0.006100
27	CAN	0.000067	0.000065	0.000064	0.000064	0.000065	0.000066
28	CHE	0.000060	0.000058	0.000055	0.000052	0.000050	0.000049
29	CHL	0.000181	0.000168	0.000158	0.000150	0.000144	0.000138
..	...	...	...	...	...	...	...
154	SWE	0.000035	0.000036	0.000038	0.000042	0.000043	0.000045
155	SWZ	0.004522	0.004602	0.004625	0.004615	0.004566	0.004474
156	SYC	0.000235	0.000235	0.000265	0.000277	0.000280	0.000289
157	SYR	0.000729	0.000679	0.000618	0.000558	0.000502	0.000467
158	TCD	0.009576	0.010039	0.010045	0.009287	0.009537	0.009127
159	TGO	0.004926	0.004656	0.004240	0.004192	0.004112	0.003992
160	THA	0.000327	0.000305	0.000286	0.000270	0.000253	0.000236
161	TJK	0.002828	0.002541	0.002259	0.002007	0.001809	0.001676
162	TKM	0.001695	0.001656	0.001685	0.001742	0.001772	0.001751
163	TLS	0.004720	0.004293	0.003896	0.003517	0.003142	0.002785
164	TON	0.000464	0.000460	0.000470	0.000455	0.000465	0.000475
165	TTO	0.000416	0.000416	0.000417	0.000419	0.000418	0.000415
166	TUN	0.000557	0.000513	0.000472	0.000433	0.000403	0.000377
167	TUR	0.000860	0.000785	0.000712	0.000642	0.000577	0.000523
168	TUV	0.000000	0.000000	0.001221	0.000999	0.000799	0.000700
169	TZA	0.005227	0.004927	0.004656	0.004370	0.004092	0.003820
170	UGA	0.006840	0.006381	0.006208	0.005711	0.004972	0.004616
171	UKR	0.000155	0.000144	0.000136	0.000130	0.000127	0.000129
172	URY	0.000266	0.000253	0.000246	0.000238	0.000230	0.000220
173	USA	0.000112	0.000111	0.000111	0.000112	0.000114	0.000114
174	UZB	0.001394	0.001317	0.001245	0.001184	0.001140	0.001109
175	VCT	0.000462	0.000424	0.000411	0.000392	0.000398	0.000370
176	VEN	0.000515	0.000494	0.000475	0.000455	0.000436	0.000415
177	VNM	0.000575	0.000529	0.000517	0.000510	0.000509	0.000513
178	VUT	0.000879	0.000851	0.000819	0.000808	0.000808	0.000789
179	WSM	0.000634	0.000619	0.000593	0.000578	0.000581	0.000572
180	YEM	0.003555	0.003175	0.002975	0.002902	0.002798	0.002575
181	ZAF	0.001952	0.001952	0.001923	0.001842	0.001749	0.001672
182	ZMB	0.007016	0.006522	0.006150	0.005427	0.004918	0.004554
183	ZWE	0.003529	0.003578	0.003623	0.003618	0.003632	0.003656

	2006	2007	2008	2009	2010	2011	2012	\
0	0.004801	0.004542	0.004372	0.004272	0.004039	0.003828	0.003590	
1	0.008440	0.008249	0.008058	0.007863	0.007597	0.007340	0.007094	
2	0.000246	0.000223	0.000202	0.000190	0.000185	0.000185	0.000194	
3	0.000123	0.000108	0.000091	0.000094	0.000094	0.000094	0.000089	
4	0.000318	0.000301	0.000287	0.000274	0.000262	0.000253	0.000245	
5	0.000349	0.000335	0.000317	0.000299	0.000280	0.000262	0.000244	
6	0.000197	0.000207	0.000181	0.000179	0.000176	0.000163	0.000151	
7	0.000072	0.000071	0.000070	0.000069	0.000067	0.000062	0.000058	
8	0.000047	0.000045	0.000043	0.000042	0.000041	0.000039	0.000038	

9	0.000745	0.000716	0.000695	0.000688	0.000697	0.000722	0.000756
10	0.005166	0.005012	0.004875	0.004747	0.004597	0.004477	0.004336
11	0.000058	0.000057	0.000057	0.000056	0.000054	0.000053	0.000051
12	0.004610	0.004467	0.004353	0.004235	0.004068	0.003933	0.003771
13	0.006121	0.005712	0.005252	0.006015	0.004683	0.004284	0.004515
14	0.001539	0.001327	0.001228	0.001119	0.001033	0.000969	0.000887
15	0.000134	0.000133	0.000134	0.000135	0.000133	0.000130	0.000118
16	0.000169	0.000155	0.000144	0.000132	0.000122	0.000129	0.000124
17	0.000264	0.000261	0.000255	0.000258	0.000246	0.000236	0.000230
18	0.000055	0.000063	0.000072	0.000079	0.000080	0.000076	0.000068
19	0.000089	0.000083	0.000077	0.000073	0.000069	0.000066	0.000065
20	0.000521	0.000490	0.000463	0.000443	0.000438	0.000418	0.000410
21	0.000363	0.000323	0.000287	0.000261	0.000245	0.000237	0.000238
22	0.000215	0.000207	0.000196	0.000185	0.000177	0.000173	0.000169
23	0.000146	0.000141	0.000144	0.000137	0.000142	0.000148	0.000155
24	0.001215	0.001125	0.001049	0.000972	0.000886	0.000803	0.000719
25	0.001536	0.001524	0.001516	0.001431	0.001379	0.001340	0.001300
26	0.005727	0.005554	0.005381	0.005155	0.004996	0.005020	0.004718
27	0.000066	0.000066	0.000066	0.000066	0.000065	0.000062	0.000059
28	0.000049	0.000047	0.000047	0.000047	0.000046	0.000045	0.000045
29	0.000135	0.000131	0.000129	0.000125	0.000123	0.000120	0.000117
..	...	...	...	...	...	...	...
154	0.000044	0.000042	0.000040	0.000037	0.000037	0.000035	0.000035
155	0.004183	0.003995	0.003893	0.003554	0.003008	0.002679	0.002449
156	0.000294	0.000282	0.000276	0.000276	0.000255	0.000287	0.000261
157	0.000443	0.000431	0.000414	0.000402	0.000375	0.000000	0.000000
158	0.008886	0.008688	0.008554	0.008312	0.008142	0.008006	0.007744
159	0.003909	0.003664	0.003559	0.003417	0.003300	0.003155	0.003046
160	0.000221	0.000206	0.000192	0.000179	0.000170	0.000161	0.000153
161	0.001590	0.001549	0.001534	0.001534	0.001545	0.001530	0.001514
162	0.001678	0.001563	0.001425	0.001304	0.001221	0.001167	0.001145
163	0.002458	0.002170	0.001943	0.001809	0.001814	0.001884	0.002028
164	0.000470	0.000470	0.000470	0.000476	0.000476	0.000476	0.000466
165	0.000408	0.000397	0.000384	0.000371	0.000357	0.000343	0.000327
166	0.000356	0.000339	0.000325	0.000311	0.000301	0.000292	0.000286
167	0.000475	0.000434	0.000398	0.000366	0.000339	0.000317	0.000297
168	0.000700	0.000600	0.000600	0.000600	0.000600	0.000545	0.000545
169	0.003549	0.003275	0.003019	0.002799	0.002584	0.002409	0.002292
170	0.004266	0.003933	0.003612	0.003333	0.003085	0.002839	0.002576
171	0.000133	0.000139	0.000145	0.000147	0.000145	0.000137	0.000124
172	0.000210	0.000202	0.000195	0.000189	0.000181	0.000174	0.000167
173	0.000112	0.000109	0.000106	0.000100	0.000097	0.000092	0.000088
174	0.001093	0.001085	0.001081	0.001072	0.001053	0.001021	0.000980
175	0.000367	0.000385	0.000385	0.000376	0.000376	0.000345	0.000345
176	0.000396	0.000380	0.000368	0.000356	0.000347	0.000339	0.000328
177	0.000509	0.000494	0.000473	0.000461	0.000443	0.000425	0.000412
178	0.000812	0.000839	0.000881	0.000897	0.000899	0.000881	0.000844
179	0.000563	0.000560	0.000563	0.000713	0.000559	0.000534	0.000502

180	0.002425	0.002176	0.002020	0.001874	0.001734	0.001622	0.001506
181	0.001628	0.001623	0.001597	0.001514	0.001336	0.001181	0.001071
182	0.004261	0.003982	0.003757	0.003509	0.003420	0.003207	0.002931
183	0.003624	0.003611	0.003600	0.003588	0.003541	0.003441	0.003072

	2013	2014	2015
0	0.003261	0.003070	0.002912
1	0.006874	0.006645	0.006258
2	0.000204	0.000211	0.000213
3	0.000082	0.000076	0.000069
4	0.000236	0.000229	0.000221
5	0.000225	0.000208	0.000193
6	0.000149	0.000148	0.000135
7	0.000053	0.000050	0.000048
8	0.000037	0.000035	0.000033
9	0.000785	0.000790	0.000763
10	0.004176	0.004027	0.003895
11	0.000050	0.000048	0.000047
12	0.003680	0.003561	0.003396
13	0.003759	0.003528	0.003358
14	0.000830	0.000782	0.000743
15	0.000107	0.000096	0.000089
16	0.000117	0.000112	0.000107
17	0.000211	0.000203	0.000192
18	0.000059	0.000050	0.000044
19	0.000064	0.000061	0.000060
20	0.000397	0.000386	0.000374
21	0.000244	0.000252	0.000256
22	0.000162	0.000165	0.000157
23	0.000170	0.000177	0.000185
24	0.000641	0.000579	0.000531
25	0.001240	0.001195	0.001164
26	0.004743	0.004470	0.004311
27	0.000056	0.000053	0.000051
28	0.000044	0.000041	0.000041
29	0.000112	0.000109	0.000105
..	...	...	...
154	0.000035	0.000035	0.000035
155	0.002257	0.002065	0.001979
156	0.000255	0.000253	0.000247
157	0.000000	0.000000	0.000000
158	0.007461	0.007255	0.007018
159	0.002941	0.002775	0.002653
160	0.000147	0.000138	0.000133
161	0.001489	0.001446	0.001387
162	0.001132	0.001118	0.001086
163	0.002181	0.002266	0.002250
164	0.000452	0.000442	0.000423

```
165  0.000311  0.000299  0.000284
166  0.000281  0.000273  0.000261
167  0.000279  0.000259  0.000240
168  0.000545  0.000545  0.000545
169  0.002216  0.002118  0.002051
170  0.002389  0.002236  0.002130
171  0.000110  0.000103  0.000094
172  0.000158  0.000149  0.000140
173  0.000084  0.000080  0.000078
174  0.000937  0.000891  0.000844
175  0.000318  0.000282  0.000273
176  0.000316  0.000304  0.000292
177  0.000400  0.000407  0.000372
178  0.000789  0.000741  0.000695
179  0.000468  0.000437  0.000414
180  0.001382  0.001290  0.001205
181  0.000911  0.000807  0.000762
182  0.002679  0.002512  0.002394
183  0.002884  0.002779  0.002678

[184 rows x 17 columns]
```

In [34]:
```
index=['2000', '2001', '2002', '2003', '2004','2005', '2006', '2007', '2008', '2009',
       '2011', '2012', '2013', '2014', '2015']
columns = ['Correlation']
gdp_who_data_corr_years = pd.DataFrame(index=index, columns=columns)

# We expect a negative coorelation for gdp vs who (increase gdp should imply a who de
for year in ['2000', '2001', '2002', '2003', '2004','2005', '2006', '2007', '2008', '
       '2011', '2012', '2013', '2014', '2015']:
    df_who_gdp_geo_years[year+'_gdp'] = df_who_gdp_geo_years[year+'_gdp'].astype(float
    df_who_gdp_geo_years[year+'_who'] = df_who_gdp_geo_years[year+'_who'].astype(float
    gdp_who_data_corr_years.at[year,'Correlation'] = df_who_gdp_geo_years[year+'_gdp']

gdp_who_data_corr_years.head(16)
```

Out[34]:

	Correlation
2000	-0.138816
2001	-0.140749
2002	-0.139883
2003	-0.138332
2004	-0.140008
2005	-0.14155
2006	-0.142126
2007	-0.142893
2008	-0.14857
2009	-0.14848
2010	-0.151082

```
2011   -0.150105
2012   -0.153503
2013   -0.154748
2014   -0.156382
2015   -0.159302
```

In [35]:
```
# ************************** REL ****************************************

df_rel = pd.read_csv('./religions_corr_ds_new.csv', sep=';', encoding = "ISO-8859-1")
```

In [36]:
```
df_rel.head(5)
```

Out[36]:

	Country	TOT	Muslim	Catholic	Protestant	Buddhism	\
0	Afghanistan	100	99.7	NaN	NaN	NaN	
1	Albania	100	58.8	10.0	NaN	NaN	
2	Algeria	100	99.0	NaN	NaN	NaN	
3	American Samoa	100	NaN	NaN	NaN	NaN	
4	Andorra	100	NaN	90.0	9.0	NaN	

	Orthodox Christian	Others	Hinduism	Shintoism	Jewish	Christians	\
0	NaN	0.3	NaN	NaN	NaN	NaN	
1	6.8	5.7	NaN	NaN	NaN	NaN	
2	NaN	1.0	NaN	NaN	NaN	NaN	
3	NaN	1.0	NaN	NaN	NaN	98.3	
4	NaN	1.0	NaN	NaN	NaN	NaN	

	Atheism	Jehova's Witness	Taoism	None	Unspecified	NOTE
0	NaN	NaN	NaN	NaN	NaN	NaN
1	2.5	NaN	NaN	NaN	16.2	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	0.7	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN

In [37]:
```
# Let's replace NaN with 0
df_rel = df_rel.fillna(0)
```

In [38]:
```
df_rel_countries = df_rel[['Country']]
df_rel_countries.to_csv('tmp/rel_countries.txt')
```

In [39]:
```
# A file containing only the WHO Countries' names is created: column with ',' separat
!awk 'BEGIN {FS=","}; {print $2}' tmp/rel_countries.txt > tmp/rel_countries_names.txt

# A file with rows in df_geo_ds_codes_names.csv and not in df_who_countries_names.txt
!awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/df_geo_ds_codes_names.csv tmp/rel_countries_

# A file with rows in df_who_countries_names.txt and not in df_geo_ds_codes_names.csv
!awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/rel_countries_names.txt tmp/df_geo_ds_codes_
```

In [40]:
```
!cat tmp/not_in_geo_rel.txt
```

```
Bahamas The
Bosnia and Herzegovina
Burma
Cabo Verde
Congo Democratic Republic of the
Congo Republic of the
Cote d'Ivoire
European Union
Gambia The
Gaza Strip
Holy See
Korea North
Korea South
Macau
Micronesia Federated States of
Niue Ekalesia
Pitcairn Islands
Saint Helena Ascension and Tristan da Cunha
Timor-Leste
Virgin Islands
West Bank
World
```

```
In [41]: !cat tmp/not_in_rel.txt
```

```
Antarctica
Aland Islands
Bosnia and Herzegovina
Bonaire, Saint Eustatius and Saba
Bahamas
Bouvet Island
Democratic Republic of the Congo
Republic of the Congo
Ivory Coast
Cape Verde
Micronesia
French Guiana
Gambia
Guadeloupe
South Georgia and the South Sandwich Islands
Heard Island and McDonald Islands
British Indian Ocean Territory
North Korea
South Korea
Myanmar
Macao
Martinique
```

```
Niue
Pitcairn
Palestinian Territory
Reunion
Saint Helena
Svalbard and Jan Mayen
French Southern Territories
East Timor
United States Minor Outlying Islands
Vatican
U.S. Virgin Islands
Mayotte
Serbia and Montenegro
Netherlands Antilles


<head>
    <style>
    ul {
        list-style-type: square;
        margin: 0;
        padding: 0;
        overflow: hidden;
        background-color: #333333;
    }

    li {
        float: left;
    }

    li a {
        display: block;
        color: white;
        text-align: center;
        padding: 16px;
        text-decoration: none;
    }

    li a:hover {
        background-color: #111111;
    }
    </style>
</head>
<body>
    Comparing the two list we can match some record and then replace the
    corresponding geo values in the rel file
</body>
```

```
In [42]: df_rel['Country'].replace('Bahamas The','Bahamas',inplace=True)
```

```
df_rel['Country'].replace('Bosnia and Herzegovina','Bosnia and Herzegovina',inplace=Tr
df_rel['Country'].replace('Cabo Verde','Cape Verde',inplace=True)
df_rel['Country'].replace('Congo Democratic Republic of the','Democratic Republic of t
df_rel['Country'].replace('Congo Republic of the','Republic of the Congo',inplace=True
df_rel['Country'].replace('Cote d''Ivoire','Ivory Coast',inplace=True)
df_rel['Country'].replace('Gambia The','Gambia',inplace=True)
df_rel['Country'].replace('Holy See','Vatican',inplace=True)
df_rel['Country'].replace('Korea North','North Korea',inplace=True)
df_rel['Country'].replace('Korea South','South Korea',inplace=True)
df_rel['Country'].replace('Macau','Macao',inplace=True)
df_rel['Country'].replace('Micronesia Federated States of','Micronesia',inplace=True)
df_rel['Country'].replace('Niue Ekalesia','Niue',inplace=True)
df_rel['Country'].replace('Pitcairn Islands','Pitcairn',inplace=True)
df_rel['Country'].replace('Saint Helena Ascension and Tristan da Cunha','Saint Helena
df_rel['Country'].replace('Timor-Leste','East Timor',inplace=True)
df_rel['Country'].replace('Virgin Islands','U.S. Virgin Islands',inplace=True)
df_rel['Country'].replace('Burma','Myanmar',inplace=True)
df_rel['Country'].replace('West Bank','Palestinian Territory',inplace=True)
```

In [43]:
```
# Now we can merge the geo datset with the rel dataset and add geo codes to rel datas
df_rel_geo = pd.merge(df_rel, df_geo_ds_codes, left_on='Country', right_on='Country',
```

In [44]:
```
df_rel_geo.head(5)
```

Out[44]:

	Country	TOT	Muslim	Catholic	Protestant	Buddhism	\
0	Afghanistan	100	99.7	0.0	0.0	0.0	
1	Albania	100	58.8	10.0	0.0	0.0	
2	Algeria	100	99.0	0.0	0.0	0.0	
3	American Samoa	100	0.0	0.0	0.0	0.0	
4	Andorra	100	0.0	90.0	9.0	0.0	

	Orthodox Christian	Others	Hinduism	Shintoism	...	Atheism	\
0	0.0	0.3	0.0	0.0	...	0.0	
1	6.8	5.7	0.0	0.0	...	2.5	
2	0.0	1.0	0.0	0.0	...	0.0	
3	0.0	1.0	0.0	0.0	...	0.0	
4	0.0	1.0	0.0	0.0	...	0.0	

	Jehova's Witness	Taoism	None	Unspecified	NOTE	ISO3	latitude	\
0	0.0	0.0	0.0	0.0	0	AFG	33,93911	
1	0.0	0.0	0.0	16.2	0	ALB	41,153332	
2	0.0	0.0	0.0	0.0	0	DZA	28,033886	
3	0.0	0.0	0.7	0.0	0	ASM	-14,270972	
4	0.0	0.0	0.0	0.0	0	AND	42,546245	

	longitude	Population
0	67,709953	29121286
1	20,168331	2986952

```
2     1,659626   34586184
3  -170,132217      57881
4     1,601554      84000

[5 rows x 22 columns]
```

In [45]:
```
df_rel_geo.columns
```

Out[45]:
```
Index(['Country', 'TOT', 'Muslim', 'Catholic', 'Protestant', 'Buddhism',
       'Orthodox Christian', 'Others', 'Hinduism', 'Shintoism', 'Jewish',
       'Christians', 'Atheism', 'Jehova's Witness', 'Taoism', 'None',
       'Unspecified', 'NOTE', 'ISO3', 'latitude', 'longitude', 'Population'],
      dtype='object')
```

In [46]:
```
# Now we can merge the who datset with the rel dataset
df_rel_who = pd.merge(df_rel_geo, df_who_geo_years, on='ISO3',how='inner')
```

In [47]:
```
index1=['Muslim', 'Catholic', 'Protestant', 'Buddhism','Orthodox Christian', 'Hinduis
       'Shintoism', 'Jewish','Christians', 'Atheism']
columns1 = ['Correlation']
rel_who_data_corr_years = pd.DataFrame(index=index1, columns=columns1)

for col in index1:
    df_rel_who[col] = df_rel_who[col].astype(float)
    rel_who_data_corr_years.at[col,'Correlation'] = df_rel_who['2015'].corr(df_rel_who

rel_who_data_corr_years.head(10)
```

Out[47]:

	Correlation
Muslim	0.114851
Catholic	-0.115766
Protestant	-0.0955993
Buddhism	-0.0303042
Orthodox Christian	-0.0586622
Hinduism	0.423479
Shintoism	-0.0182075
Jewish	-0.0203663
Christians	0.0355385
Atheism	-0.0477882

In [48]:
```
index2=['latitude','longitude']
columns2 = ['Correlation']
rel_geo_data_corr_years = pd.DataFrame(index=index2, columns=columns2)

for col in index2:
    df_rel_who[col] = df_rel_who[col].str.replace(',','.')
    df_rel_who[col] = df_rel_who[col].astype(float)
    rel_geo_data_corr_years.at[col,'Correlation'] = df_rel_who['2015'].corr(df_rel_who

rel_geo_data_corr_years.head(2)
```

```
Out[48]:            Correlation
          latitude     -0.05926
          longitude    0.101765

In [49]: # The file countries.json will be used to get data in a geographical map using folium
         # Let's get from countries.json the list of ISO codes used in this file and let's com
         # to the list of ISO code in the datasets: json codes will be listed in the file
         # tmp/geojson_codes.txt
         !python -m json.tool countries.geojson  | grep -i \"adm0_a3\": | sed s/'

         # Let's now create files containing the geo and imf datasets' geo codes
         !awk 'BEGIN {FS=";"}; {print $2}' tmp/df_geo_ds_codes.csv > tmp/geo_country_codes.txt
         !awk 'BEGIN {FS=";"}; {print $2}' imf_weo_ds.csv | uniq > tmp/imf_country_codes.txt

         # And now we create files containing the rel and who datasets' geo codes
         df_rel_geo[['ISO3']].to_csv('tmp/rel_geo_countries.txt')
         df_who_geo[['ISO3']].to_csv('tmp/who_geo_countries.txt')

         # Finally we create files containing only the rel and who datasets' geo codes
         !awk 'BEGIN {FS=","}; {print $2}' tmp/rel_geo_countries.txt > tmp/rel_geo_country_cod
         !awk 'BEGIN {FS=","}; {print $2}' tmp/who_geo_countries.txt | uniq > tmp/who_geo_coun

         # Codes used in geojson file vs codes used in imf dataset
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/geojson_codes.txt tmp/imf_country_codes.txt
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/imf_country_codes.txt tmp/geojson_codes.txt

         # Codes used in geojson file vs codes used in geo dataset
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/geojson_codes.txt tmp/geo_country_codes.txt
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/geo_country_codes.txt tmp/geojson_codes.txt

         # Codes used in geojson file vs codes used in rel dataset
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/geojson_codes.txt tmp/rel_geo_country_codes
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/rel_geo_country_codes.txt tmp/geojson_codes

         # Codes used in geojson file vs codes used in who dataset
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/geojson_codes.txt tmp/who_geo_country_codes
         !awk 'FNR==NR {a[$0]++; next} !a[$0]' tmp/who_geo_country_codes.txt tmp/geojson_codes

In [50]: !cat tmp/not_in_imf_from_geojson.txt| grep -v ISO | awk -vORS=, '{ print $1 }' | sed

ATA','ATF','CUB','CYN','FLK','GRL','KOS','NCL','PRK','PSX','SAH','SDS','SOL','SOM


In [51]: # We now can remove the codes in the geojson file not present in the IMF dataset
         !ogr2ogr -f "GeoJSON" filtered_imf.geojson -dialect SQLITE -sql "SELECT * FROM OGRGeo

In [52]: !cat tmp/not_in_who_from_geojson.txt| grep -v ISO3 | awk -vORS=, '{ print $1 }' | sed

ATA','ATF','BOL','CIV','CYN','FLK','GRL','KOS','LAO','NCL','PRI','PRK','PSX','SAH','SDS','SOL'
```

```
In [53]: # We now can remove the codes in the geojson file not present in the WHO dataset
         !ogr2ogr -f "GeoJSON" filtered_who.geojson -dialect SQLITE -sql "SELECT * FROM OGRGeo

In [54]: !cat tmp/not_in_rel_from_geojson.txt| grep -v ISO3 | awk -vORS=, '{ print $1 }' | sed

ATA','ATF','BIH','CIV','CYN','KOS','PSX','SAH','SDS','SOL


In [55]: # We now can remove the codes in the geojson file not present in the REL dataset
         !ogr2ogr -f "GeoJSON" filtered_rel.geojson -dialect SQLITE -sql "SELECT * FROM OGRGeo

In [56]: !cat tmp/not_in_geo_from_geojson.txt| grep -v ISO3 | awk -vORS=, '{ print $1 }' | sed

CYN','KOS','PSX','SAH','SDS','SOL


In [57]: # We now can remove the codes in the geojson file not present in the REL dataset
         !ogr2ogr -f "GeoJSON" filtered_geo.geojson -dialect SQLITE -sql "SELECT * FROM OGRGeo

In [58]: df_data_plot = df_imf_gdp_pc[['ISO','Country','Units','Scale','2000']]

In [59]: !cat tmp/not_in_geojson_from_imf.txt| grep -v ISO | awk -vORS=, '{ print $1 }' | sed

ATG','BHR','BRB','CPV','COM','DMA','GRD','HKG','KIR','UVK','MAC','MDV','MLT','MHL','MUS','FSM'


In [60]: df_data_plot = df_data_plot[~df_data_plot['ISO'].isin(['ATG','BHR','BRB','CPV','COM',
         ])]

In [61]: df_data_plot.replace('n/a',0,inplace=True)
         df_data_plot['2000'] = df_data_plot['2000'].astype(np.str).str.replace(',','')
         df_data_plot['2000'] = df_data_plot['2000'].astype(np.str).str.replace('.','')
         df_data_plot['2000'] = df_data_plot['2000'].astype(np.int64)
         df_data_plot.head(5)
```

Out[61]:

	ISO	Country	Units	Scale	2000
7	AFG	Afghanistan	U.S. dollars	Units	0
51	ALB	Albania	U.S. dollars	Units	1127640
95	DZA	Algeria	U.S. dollars	Units	1794695
139	AGO	Angola	U.S. dollars	Units	535473
227	ARG	Argentina	U.S. dollars	Units	8386586

```
In [62]: df_data_plot2 = df_imf_gdp_pc[['ISO','Country','Units','Scale','2015']]
         df_data_plot2 = df_data_plot2[~df_data_plot2['ISO'].isin(['ATG','BHR','BRB','CPV','CO
         ])]
         df_data_plot2.replace('n/a',0,inplace=True)
         df_data_plot2['2015'] = df_data_plot2['2015'].astype(np.str).str.replace(',','')
         df_data_plot2['2015'] = df_data_plot2['2015'].astype(np.str).str.replace('.','')
         df_data_plot2['2015'] = df_data_plot2['2015'].astype(np.int)
         df_data_plot2.head(5)
```

Out[62]:

	ISO	Country	Units	Scale	2015
7	AFG	Afghanistan	U.S. dollars	Units	615091
51	ALB	Albania	U.S. dollars	Units	3943217
95	DZA	Algeria	U.S. dollars	Units	4123297
139	AGO	Angola	U.S. dollars	Units	3876197
227	ARG	Argentina	U.S. dollars	Units	14643922

In [63]:
```
df_data_plot3 = df_imf_pop[['ISO','Country','Units','Scale','2015']]
df_data_plot3 = df_data_plot3[~df_data_plot3['ISO'].isin(['ATG','BHR','BRB','CPV','CO
])]
df_data_plot3.replace('n/a',0,inplace=True)
df_data_plot3['2015'] = df_data_plot3['2015'].astype(np.str).str.replace(',','')
df_data_plot3['2015'] = df_data_plot3['2015'].astype(np.str).str.replace('.','')
df_data_plot3['2015'] = df_data_plot3['2015'].astype(np.int)
df_data_plot3.head(5)
```

Out[63]:

	ISO	Country	Units	Scale	2015
26	AFG	Afghanistan	Persons	Millions	32007
70	ALB	Albania	Persons	Millions	2889
114	DZA	Algeria	Persons	Millions	39963
158	AGO	Angola	Persons	Millions	26563
246	ARG	Argentina	Persons	Millions	43132

In [64]:
```
df_data_plot4 = df_imf_gdp[['ISO','Country','Units','Scale','2015']]
df_data_plot4 = df_data_plot4[~df_data_plot4['ISO'].isin(['ATG','BHR','BRB','CPV','CO
])]
df_data_plot4.replace('n/a',0,inplace=True)
df_data_plot4['2015'] = df_data_plot4['2015'].astype(np.str).str.replace(',','')
#df_data_plot4['2015'] = df_data_plot4['2015'].astype(np.str).str.replace('.','')
df_data_plot4['2015'] = df_data_plot4['2015'].astype(np.float)
df_data_plot4.head(5)
```

Out[64]:

	ISO	Country	Units	Scale	2015
3	AFG	Afghanistan	U.S. dollars	Billions	19687.0
47	ALB	Albania	U.S. dollars	Billions	11393.0
91	DZA	Algeria	U.S. dollars	Billions	164779.0
135	AGO	Angola	U.S. dollars	Billions	102962.0
223	ARG	Argentina	U.S. dollars	Billions	631621.0

In [65]:
```
map_imf = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
json_geo = "filtered_imf.geojson"
map_imf.choropleth(geo_path=json_geo,data=df_data_plot,columns=['ISO', '2000'],key_on=
                   fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
map_imf
```

Out[65]: <folium.folium.Map at 0x7fcdba9e90f0>

In [66]:
```
map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
json_geo = "filtered_imf.geojson"
```

```
        map.choropleth(geo_path=json_geo,data=df_data_plot2,columns=['ISO', '2015'],key_on='f
                       fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
        #map.choropleth(geo_path=json_geo)
        map
```

```
Out[66]: <folium.folium.Map at 0x7fcdb9e2ba20>
```

```
In [67]: map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
         json_geo = "filtered_imf.geojson"

         map.choropleth(geo_path=json_geo,data=df_data_plot3,columns=['ISO', '2015'],key_on='f
                       fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2,legend_name='P
         #map.choropleth(geo_path=json_geo)
         map
```

```
Out[67]: <folium.folium.Map at 0x7fcdb9be5ba8>
```

```
In [68]: map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
         json_geo = "filtered_imf.geojson"
         map.choropleth(geo_path=json_geo,data=df_data_plot4,columns=['ISO', '2015'],key_on='f
                       fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
         #map.choropleth(geo_path=json_geo)
         map
```

```
Out[68]: <folium.folium.Map at 0x7fcdb5743c50>
```

```
In [69]: map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
         json_geo = "filtered_who.geojson"
         map.choropleth(geo_path=json_geo,data=df_who_pop_geo_years,columns=['ISO3', '2000'],k
                       fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
         map
```

```
Out[69]: <folium.folium.Map at 0x7fcdb550b6d8>
```

```
In [70]: map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
         json_geo = "filtered_who.geojson"
         map.choropleth(geo_path=json_geo,data=df_who_pop_geo_years,columns=['ISO3', '2015'],k
                       fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
         map
```

```
Out[70]: <folium.folium.Map at 0x7fcdb5328c88>
```

```
In [71]: df_rel_geo_mus = df_rel_geo[['Muslim','ISO3', 'Country']].copy()
         df_rel_geo_cat = df_rel_geo[['Catholic','ISO3', 'Country']].copy()
         df_rel_geo_pro = df_rel_geo[['Protestant','ISO3', 'Country']].copy()
         df_rel_geo_bud = df_rel_geo[['Buddhism','ISO3', 'Country']].copy()
```

```
In [72]: df_rel_geo_mus.head(5)
```

Out[72]:

	Muslim	ISO3	Country
0	99.7	AFG	Afghanistan
1	58.8	ALB	Albania
2	99.0	DZA	Algeria
3	0.0	ASM	American Samoa
4	0.0	AND	Andorra

In [73]:
```
df_rel_geo_mus['Muslim'] = df_rel_geo_mus['Muslim'].astype(str)
df_rel_geo_mus['Muslim']  = df_rel_geo_mus['Muslim'].apply(pd.to_numeric, errors='coe

df_rel_geo_cat['Catholic'] = df_rel_geo_cat['Catholic'].astype(str)
df_rel_geo_cat['Catholic']  = df_rel_geo_cat['Catholic'].apply(pd.to_numeric, errors=

df_rel_geo_pro['Protestant'] = df_rel_geo_pro['Protestant'].astype(str)
df_rel_geo_pro['Protestant']  = df_rel_geo_pro['Protestant'].apply(pd.to_numeric, erro

df_rel_geo_bud['Buddhism'] = df_rel_geo_bud['Buddhism'].astype(str)
df_rel_geo_bud['Buddhism']  = df_rel_geo_bud['Buddhism'].apply(pd.to_numeric, errors=

df_rel_geo_mus.head(5)
```

Out[73]:

	Muslim	ISO3	Country
0	99.7	AFG	Afghanistan
1	58.8	ALB	Albania
2	99.0	DZA	Algeria
3	0.0	ASM	American Samoa
4	0.0	AND	Andorra

In [74]:
```
map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
json_geo = "filtered_rel.geojson"
map.choropleth(geo_path=json_geo,data=df_rel_geo_mus,columns=['ISO3', 'Muslim'],key_or
               fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='M
map
```

Out[74]: <folium.folium.Map at 0x7fcdb50cf828>

In [75]:
```
map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
json_geo = "filtered_rel.geojson"
map.choropleth(geo_path=json_geo,data=df_rel_geo_cat,columns=['ISO3', 'Catholic'],key_
               fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='C
map
```

Out[75]: <folium.folium.Map at 0x7fcdb4e98668>

In [76]:
```
map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
json_geo = "filtered_rel.geojson"
map.choropleth(geo_path=json_geo,data=df_rel_geo_pro,columns=['ISO3', 'Protestant'],ke
               fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='P
map
```

```
Out[76]: <folium.folium.Map at 0x7fcdb4c81ac8>

In [77]: map = folium.Map(location=[41.87, 12.57], zoom_start=1.5)
         json_geo = "filtered_rel.geojson"
         map.choropleth(geo_path=json_geo,data=df_rel_geo_bud,columns=['ISO3', 'Buddhism'],key
                        fill_color='YlGnBu', fill_opacity=0.7, line_opacity=0.2, legend_name='
         map

Out[77]: <folium.folium.Map at 0x7fcdb4ab5780>
```