Final Exam Project

May 17, 2021

1 How Indicative Can Internet Usage Be For Country Growth?

2 Data Collection

[477]:
```
!pip install folium
!pip install geopy
from geopy.geocoders import Nominatim
from sklearn import linear_model
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import folium


internet_df = pd.read_csv("https://raw.githubusercontent.com/Nick-Arrazola/
 ↪320-Workspace/main/share-of-individuals-using-the-internet%20(1).csv")
#country_mean_yearly_schooling_df = pd.read_csv("https://raw.githubusercontent.
 ↪com/Nick-Arrazola/320-Workspace/main/mean-years-of-schooling-1.csv")
country_yearly_tritary_completion_df = pd.read_csv("https://raw.
 ↪githubusercontent.com/Nick-Arrazola/320-Workspace/main/
 ↪share-of-the-population-with-completed-tertiary-education.csv")
total_yearly_employment_df = pd.read_csv("https://raw.githubusercontent.com/
 ↪Nick-Arrazola/Nick-Arrazola.github.io/main/
 ↪Total%20Employement%20Per%20Country.csv")
yearly_gni_country_df = pd.read_csv("https://raw.githubusercontent.com/
 ↪Nick-Arrazola/Nick-Arrazola.github.io/main/
 ↪GNI%20per%20country%20time%20series.csv", encoding='cp1252')
```

```
Requirement already satisfied: folium in /opt/conda/lib/python3.8/site-packages
(0.12.1)
Requirement already satisfied: jinja2>=2.9 in /opt/conda/lib/python3.8/site-
packages (from folium) (2.11.2)
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages
(from folium) (1.19.5)
Requirement already satisfied: requests in /opt/conda/lib/python3.8/site-
packages (from folium) (2.25.1)
Requirement already satisfied: branca>=0.3.0 in /opt/conda/lib/python3.8/site-
```

```
packages (from folium) (0.4.2)
Requirement already satisfied: MarkupSafe>=0.23 in
/opt/conda/lib/python3.8/site-packages (from jinja2>=2.9->folium) (1.1.1)
Requirement already satisfied: chardet<5,>=3.0.2 in
/opt/conda/lib/python3.8/site-packages (from requests->folium) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.8/site-packages (from requests->folium) (1.26.2)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.8/site-packages (from requests->folium) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.8/site-
packages (from requests->folium) (2.10)
Requirement already satisfied: geopy in /opt/conda/lib/python3.8/site-packages
(2.1.0)
Requirement already satisfied: geographiclib<2,>=1.49 in
/opt/conda/lib/python3.8/site-packages (from geopy) (1.50)
```

3 what about getting rid of missing data?

4 Data Tidying

You can see from the image below, that we have some missing data in the form of '..' or 'NaN'. We don't want to use these rows at all since they will have no purpose. The rows with 'NaN' are rows in the dataset that were just empty meaning they don't belong to a country, so we don't we need it, it would just look better if we took it out instead of having wasted space in our dataframe. Likewise, the rows with '..' are countries that have no data related to it at all, so it would seem resonable to just drop those rows/ countries as they have no data related to them at all. I could replace the '..' with zeros, but then if I were to ever calculate the average value in respect to this data, then the zeros would heavily influence the mean as they would be heavy outliers since most of the data is in the thousands and up.

The same idea would apply for the other datasets.

Some of the datasets may be formatted differently than the others:

```
- ex: country1 year1 year2 year3 year4 ...

- ex: country1 year1
      country1 year2
      country1 year3
      country1 year4
```

However, I would like to format all of the datasets to be like the 2nd example, so to do that, I will have to use a function called '.melt(...)', that accomplishes just this. The reason of formating the datasets like that is because if any year (let's say year1 following the example) was empty/ missing data, then I could just drop the row with year1 by itself, while maintaining the other rows. If I had it in the previous format (as shown in example 1), then if I wanted to drop the row with the empty year1, since they are all on the same row, all of the years would get dropped as a result.

Also, I would like to center my focus from years 1990 - 2016 as those are the years my datasets have most in common, so I will drop any columns (or rows) where the years are < 1990 or > 2016.

Likewise, to make the tables look cleaner, I will drop any columns (or rows) that seem unnecessary.

[478]:
```
print(total_yearly_employment_df)
```

```
                Country Name Country Code  \
0                Afghanistan          AFG
1                    Albania          ALB
2                    Algeria          DZA
3             American Samoa          ASM
4                    Andorra          AND
..                       …          …
242                      NaN          NaN
243                      NaN          NaN
244                      NaN          NaN
245  Data from database: Jobs          NaN
246  Last Updated: 02/17/2021          NaN

                           Series Name  Series Code 1990 [YR1990]  \
0    Total employment, total (ages 15+)  SL.EMP.TOTL            ..
1    Total employment, total (ages 15+)  SL.EMP.TOTL            ..
2    Total employment, total (ages 15+)  SL.EMP.TOTL            ..
3    Total employment, total (ages 15+)  SL.EMP.TOTL            ..
4    Total employment, total (ages 15+)  SL.EMP.TOTL            ..
..                                  …          …           …
242                                 NaN          NaN          NaN
243                                 NaN          NaN          NaN
244                                 NaN          NaN          NaN
245                                 NaN          NaN          NaN
246                                 NaN          NaN          NaN

    1991 [YR1991] 1992 [YR1992] 1993 [YR1993] 1994 [YR1994] 1995 [YR1995]  \
0         3062014       3358951       3564855       3828863       4228323
1         1115397       1103796       1056370       1043688       1033693
2         5363479       5393082       5606495       5741095       5658066
3              ..            ..            ..            ..            ..
4              ..            ..            ..            ..            ..
..            …             …             …             …             …
242           NaN           NaN           NaN           NaN           NaN
243           NaN           NaN           NaN           NaN           NaN
244           NaN           NaN           NaN           NaN           NaN
245           NaN           NaN           NaN           NaN           NaN
246           NaN           NaN           NaN           NaN           NaN

     … 2007 [YR2007] 2008 [YR2008] 2009 [YR2009] 2010 [YR2010]  \
0    …       6697893       6743809       7043625       7159144
1    …       1081808       1073591       1057915       1051676
2    …       9017737       9453083       9758014      10027744
3    …            ..            ..            ..            ..
```

```
4    …             ..             ..             ..             ..
..   …            …              …              …              …
242  …            NaN            NaN            NaN            NaN
243  …            NaN            NaN            NaN            NaN
244  …            NaN            NaN            NaN            NaN
245  …            NaN            NaN            NaN            NaN
246  …            NaN            NaN            NaN            NaN

    2011 [YR2011] 2012 [YR2012] 2013 [YR2013] 2014 [YR2014] 2015 [YR2015]  \
0         7457762       7881429       8286322       8734840       9193492
1         1094029       1146944       1053747       1052702       1105349
2        10227664      10325539      10879749      10408373      10477364
3              ..            ..            ..            ..            ..
4              ..            ..            ..            ..            ..
..            …             …             …             …             …
242          NaN           NaN           NaN           NaN           NaN
243          NaN           NaN           NaN           NaN           NaN
244          NaN           NaN           NaN           NaN           NaN
245          NaN           NaN           NaN           NaN           NaN
246          NaN           NaN           NaN           NaN           NaN

    2016 [YR2016]
0         9618882
1         1129074
2        10719466
3              ..
4              ..
..            …
242          NaN
243          NaN
244          NaN
245          NaN
246          NaN

[247 rows x 31 columns]
```

[479]:
```
#----------------------------------------
#i don't want these columns in the dataframe
columns_drop = ['Series Name', 'Series Code']


for index, row in total_yearly_employment_df.iterrows():
    for i in range(0, len(row)):
        if (row[i] == '..'):
            row[i] = np.nan

#going to re-name columns
```

```
total_yearly_employment_df = total_yearly_employment_df.rename(columns = {'1991␣
↪[YR1991]' : '1991', '1992 [YR1992]' : '1992', '1993 [YR1993]' : '1993',
                                                                '1994␣
↪[YR1994]' : '1994', '1995 [YR1995]' : '1995', '1996 [YR1996]' : '1996',
                                                                '1997␣
↪[YR1997]' : '1997', '1998 [YR1998]' : '1998', '1999 [YR1999]' : '1999',
                                                                '2000␣
↪[YR2000]' : '2000', '2001 [YR2001]' : '2001', '2002 [YR2002]' : '2002',
                                                                '2003␣
↪[YR2003]' : '2003', '2004 [YR2004]' : '2004', '2005 [YR2005]' : '2005',
                                                                '2006␣
↪[YR2006]' : '2006', '2007 [YR2007]' : '2007', '2008 [YR2008]' : '2008',
                                                                '2009␣
↪[YR2009]' : '2009', '2010 [YR2010]' : '2010', '2011 [YR2011]' : '2011',
                                                                '2012␣
↪[YR2012]' : '2012', '2013 [YR2013]' : '2013', '2014 [YR2014]' : '2014',
                                                                '2015␣
↪[YR2015]' : '2015', '2016 [YR2016]' : '2016'
                                                              })

sub_yearly_employment_df = total_yearly_employment_df.drop(columns_drop, axis =␣
↪'columns', inplace = False)
employment_melt_df = pd.melt(sub_yearly_employment_df, id_vars = ['Country␣
↪Name', 'Country Code']).sort_values(by = ['Country Name', 'variable'],␣
↪ascending = True)

employment_melt_df = employment_melt_df.dropna()
employment_melt_df = employment_melt_df.rename(columns = {'variable' : 'Year',␣
↪'value' : 'Employment'})

#changes the column elements from being of datatype string to being of datatype␣
↪float
employment_melt_df['Employment'] = employment_melt_df['Employment'].
↪astype(float)
employment_melt_df
#----------------------------------------
```

[479]:

	Country Name	Country Code	Year	Employment
247	Afghanistan	AFG	1991	3062014.0
494	Afghanistan	AFG	1992	3358951.0
741	Afghanistan	AFG	1993	3564855.0
988	Afghanistan	AFG	1994	3828863.0
1235	Afghanistan	AFG	1995	4228323.0
…	…	…	…	…
5675	Zimbabwe	ZWE	2012	6749094.0
5922	Zimbabwe	ZWE	2013	6922911.0

```
6169     Zimbabwe          ZWE  2014   7102680.0
6416     Zimbabwe          ZWE  2015   7287739.0
6663     Zimbabwe          ZWE  2016   7478802.0

[5504 rows x 4 columns]
```

[480]:
```
#----------------------------------------
#i don't want these columns from the dataframe
columns_to_drop = ['2017', '2018', '2019', '2020']

#done to not overwrite the dataframe in memory (if 'inplace = TRUE')
#because I might need to use the columns that I got rid of now, later
sub_yearly_gni_df = yearly_gni_country_df.drop(columns_to_drop, axis =␣
↪'columns', inplace = False)

sub_yearly_gni_df.dropna()
gni_melt_df = pd.melt(sub_yearly_gni_df, id_vars = ['Country Name', 'Country␣
↪Code']).sort_values(by = ['Country Name', 'variable'], ascending = True)

gni_melt_df = gni_melt_df.dropna()
gni_melt_df = gni_melt_df.rename(columns = {'variable' : 'Year', 'value' :␣
↪'GNI'})
gni_melt_df
#----------------------------------------
```

[480]:
```
     Country Name Country Code  Year     GNI
5017  Afghanistan          AFG  2009  1520.0
5281  Afghanistan          AFG  2010  1710.0
5545  Afghanistan          AFG  2011  1700.0
5809  Afghanistan          AFG  2012  1920.0
6073  Afghanistan          AFG  2013  2020.0
…           …          …    …     …
6071     Zimbabwe          ZWE  2012  2070.0
6335     Zimbabwe          ZWE  2013  2310.0
6599     Zimbabwe          ZWE  2014  2360.0
6863     Zimbabwe          ZWE  2015  2410.0
7127     Zimbabwe          ZWE  2016  2560.0

[6084 rows x 4 columns]
```

[481]:
```
internet_df = internet_df.rename(columns = {'Entity' : 'Country Name', 'Code' :␣
↪'Country Code',
                                            'Individuals using the Internet (%␣
↪of population)' : 'Internet Usage (% of pop)'
                                           })
internet_df = internet_df.loc[internet_df['Year'] < 2017]
```

```
internet_df = internet_df.dropna().sort_values(by = ['Country Name', 'Year'],␣
↪ascending = True)
internet_df
```

[481]:

	Country Name	Country Code	Year	Internet Usage (% of pop)
0	Afghanistan	AFG	1990	0.000000
1	Afghanistan	AFG	2001	0.004723
2	Afghanistan	AFG	2002	0.004561
3	Afghanistan	AFG	2003	0.087891
4	Afghanistan	AFG	2004	0.105809
…	…	…	…	…
5924	Zimbabwe	ZWE	2012	12.000000
5925	Zimbabwe	ZWE	2013	15.500000
5926	Zimbabwe	ZWE	2014	16.364740
5927	Zimbabwe	ZWE	2015	22.742818
5928	Zimbabwe	ZWE	2016	23.119989

[4730 rows x 4 columns]

[482]:
```
#the original df has rows with years from 1970, but I only want the rows from␣
↪years 1990 and up
country_yearly_tritary_completion_df = country_yearly_tritary_completion_df.
↪loc[country_yearly_tritary_completion_df['Year'] >= 1990]
country_yearly_tritary_completion_df = country_yearly_tritary_completion_df.
↪rename(columns = {'Entity' : 'Country Name',
                                                                              ␣
↪               'Code' : 'Country Code',
                                                                              ␣
↪               'Barro-Lee: Percentage of population age 15+ with tertiary␣
↪schooling. Completed Tertiary' : 'Completed Tertiary (% of pop)'
                                                                              ␣
↪             })

country_yearly_tritary_completion_df
```

[482]:

	Country Name	Country Code	Year	Completed Tertiary (% of pop)
4	Afghanistan	AFG	1990	2.91
5	Afghanistan	AFG	1995	3.28
6	Afghanistan	AFG	2000	3.50
7	Afghanistan	AFG	2005	3.61
8	Afghanistan	AFG	2010	3.65
…	…	…	…	…
1291	Zimbabwe	ZWE	1990	1.83
1292	Zimbabwe	ZWE	1995	1.32
1293	Zimbabwe	ZWE	2000	0.53
1294	Zimbabwe	ZWE	2005	0.44

```
1295     Zimbabwe          ZWE  2010                          0.38

[720 rows x 4 columns]
```

I was planning on merging the most of the tables into one, larger table containing columns: Country Name, Country Code, Year, GNI, Internet Usage, Tertiary, and Employment, but to do so without having any NaN, I would have to perform some sort of intersect where all of the rows with the same Country Name and Year would be included on the same row in the merge table. Since not all of the tables have the same amount of country and year occurences, some of the rows we have in the individual tables may not be included in the merge table. Therefore, to not lose any more data/observations, I will keep the tables individually.

Now, the real data our tables collectively have are:

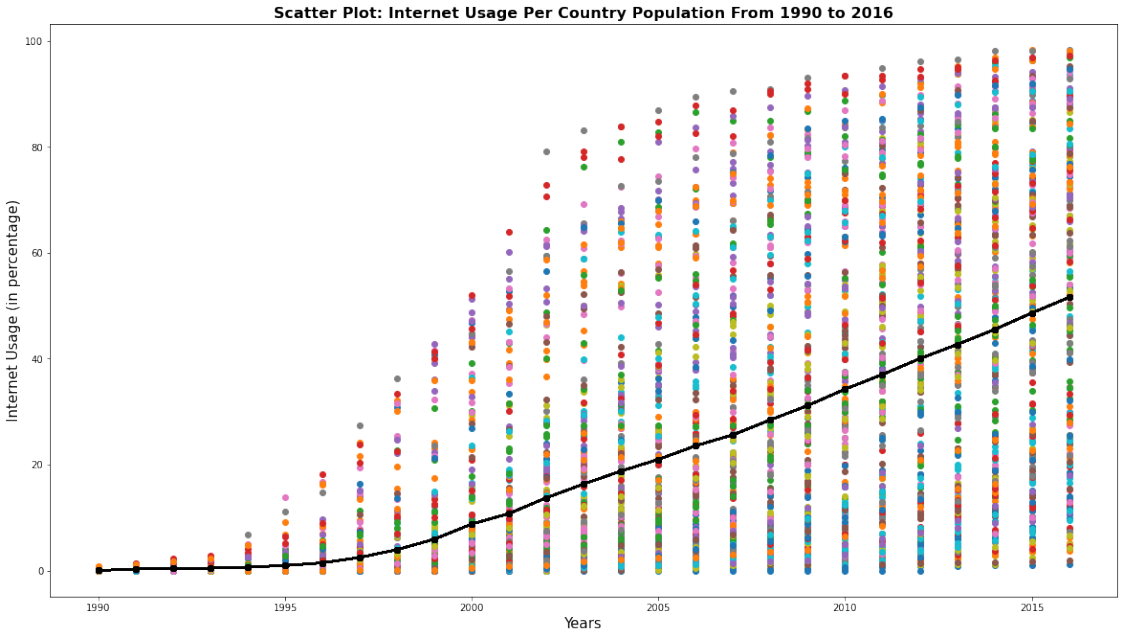
```
- Country Name: The name of the country
- Country Code: An identifier for the country
- Year: The year our observation took place
- GNI: The Gross National Income of a country in a certain year
- Internet Usage: Penetration percentage on number of internet users in repect country pop.
- Employment: The total employment of a country in a certain year
- Teritary: Percent of people who completed teritary level education in a country that year
    - Teritary level includes colleges, university, etc.
```

5 Explanatory Analysis & Data Visualization

Cool, so we know have the data, but what does that data suggest when illustarted? Will the internet usage have some relation with the developemental properties of these countries? Well, that is what will like to find out. First, let us see what the internet usuage per year looks like between these countries

[483]:
```
#gets the unique country names in our dataframe
unique_country_names = pd.unique(internet_df['Country Name'])
mean_list = []

plt.figure(figsize = (20,11))

#finding the mean of internet usage per year
mean_usage_df = internet_df[['Year','Internet Usage (% of pop)']].
 ↪groupby('Year').mean()

for country in unique_country_names:
    #gets the sub-dataframe that has all the rows that includes the current␣
 ↪country
    curr_country_stats_df = internet_df.loc[internet_df['Country Name'] ==␣
 ↪country]

    #plots the internet usage of the current country from 1990 to 2016
```

```
    plt.scatter(curr_country_stats_df['Year'], curr_country_stats_df['Internet␣
 ↪Usage (% of pop)'])

    plt.plot(mean_usage_df, marker = 'o', markersize = 5, color = 'black',
             markerfacecolor = 'black', linewidth = 2)

plt.title("Scatter Plot: Internet Usage Per Country Population From 1990 to␣
 ↪2016", fontsize = 16, fontweight = "bold")
plt.xlabel("Years", fontsize = 15)
plt.ylabel("Internet Usage (in percentage)", fontsize = 15)
plt.show()
```



Wow, this is interesting, we can see how once we passed 1993 (when the World Wide Web was made public), the porportion of internet users for many countries appears to have increased rather ploynomially (yes, linear is a polynomial, but here it looks like the high order term of this function is >= 2) to where in around 10 years, many countries have begun to reach almost full internet penetration, at to which point, the growth of internet use for those many countries began to level out. Many countries have seemingly reached the equilibrium rate of growth.

Though this is fascinating, we can still see how in relatively recent years, (granted this data is a bit out dated) there still are plentiful quantites of countries that still have low percentages of internet penetration as we can see many below the moving average (black line), how would those countries stand on an economic basis?

Before we do that, I would like to first see where the lowest 50 countries are located worldwide, this could give us more insights towards why they had low internet usage and the general condition of the country. To do I plan on using the data from the most recent year, since that shows use the

distribution of data closest to nowaways.

[484]:
```
sort_df = internet_df.loc[internet_df['Year'] == 2016].sort_values(by =␣
↪'Internet Usage (% of pop)', ascending = True)
sort_50_df = sort_df[0:50]
sort_50_df.head()
```

[484]:

	Country Name	Country Code	Year	Internet Usage (% of pop)
1540	Eritrea	ERI	2016	1.177119
4944	Somalia	SOM	2016	1.880000
2202	Guinea-Bissau	GNB	2016	3.761414
880	Central African Republic	CAF	2016	4.000000
3963	Niger	NER	2016	4.322758

Great, I now have the list, but I somehow need to find the geographical location of these countries. Thankfully, we can use geopy to convert the name of our desired location into lat and long coordinates so we can map them using folium (a library that allows developers to make maps using python). Let's how it works.

[485]:
```
map_osm = folium.Map()

count = 1
for index, row in sort_50_df.iterrows():
    location = geolocator.geocode(row[0])

    folium.Marker(
        location = [location.latitude, location.longitude],
        popup = "Rank: " + str(count) + ", " + row[0] + ", " + str(row[3]) + "␣
↪penetration",
        icon = folium.Icon(color = 'beige')
        ).add_to(map_osm)

    count += 1
map_osm
```

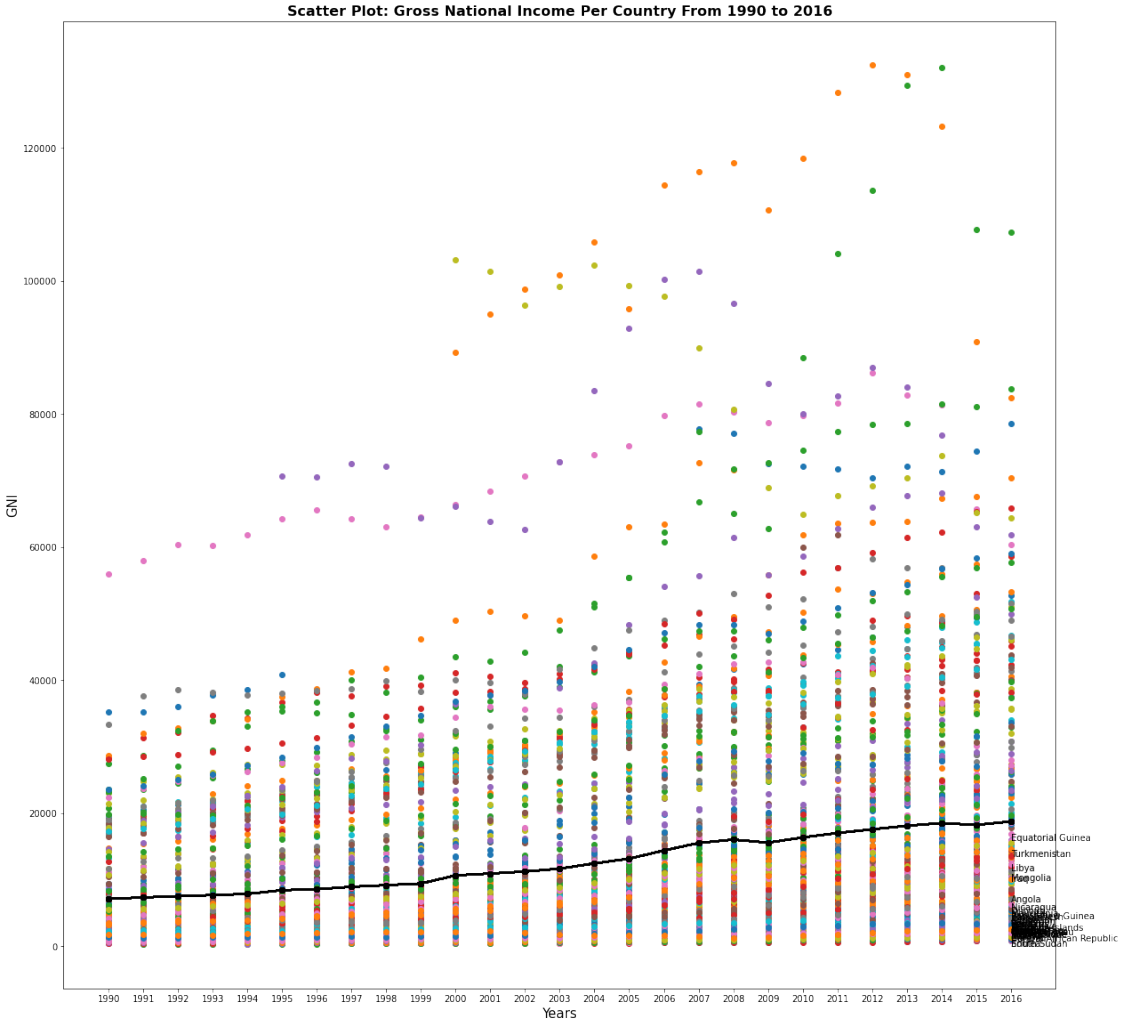
[485]:
```
<folium.folium.Map at 0x7fb134b0efa0>
```

This is very informative, as we can see, the majority of countries (as of 2016) that have the lowest usage of internet are distributed throughout Africa. You may be wondering, "how is this map useful?", well this map can give us an indiction of what to execpt of a countries developement stage. The map could be suggesting that these countries marked are also low in economic establishment, however, from this map alone, the relation is not yet definitive. How about we plot something else.

I would now like to proceed in ploting how the countries gross national income has changed throughout the years.

[486]:
```
#gets the unique country names in our dataframe
unique_country_names = pd.unique(gni_melt_df['Country Name'])
```

```
sort_50_country_lst = list(sort_50_df['Country Name'])
mean_list = []

plt.figure(figsize = (20,20))

#done so that the years would start in a weird order on the x-axis
start_df = gni_melt_df.loc[gni_melt_df['Country Name'] == 'Arab World']
plt.scatter(start_df['Year'], start_df['GNI'])

#finding the mean of internet usage per year
mean_usage_df = gni_melt_df[['Year','GNI']].groupby('Year').mean()

for country in unique_country_names:

    if country == 'Arab World':
        continue

    curr_country_stats_df = gni_melt_df.loc[gni_melt_df['Country Name'] ==␣
 ↪country]

    #checks to see if the current country is one of the bottom 50 countries in␣
 ↪internet usage
    if (country in sort_50_country_lst):
        country_2016_gni = -1
        for index, row in curr_country_stats_df.iterrows():
            if row[2] == '2016':
                country_2016_gni = row[3]
         #going to mark the country in the plot
        plt.annotate(country, ('2016', country_2016_gni))


    #plots the internet usage of the current country from 1990 to 2016
    plt.scatter(curr_country_stats_df['Year'], curr_country_stats_df['GNI'])

    plt.plot(mean_usage_df, marker = 'o', markersize = 5, color = 'black',
             markerfacecolor = 'black', linewidth = 2)

plt.title("Scatter Plot: Gross National Income Per Country From 1990 to 2016",␣
 ↪fontsize = 16, fontweight = "bold")
plt.xlabel("Years", fontsize = 15)
plt.ylabel("GNI", fontsize = 15)
plt.show()
```



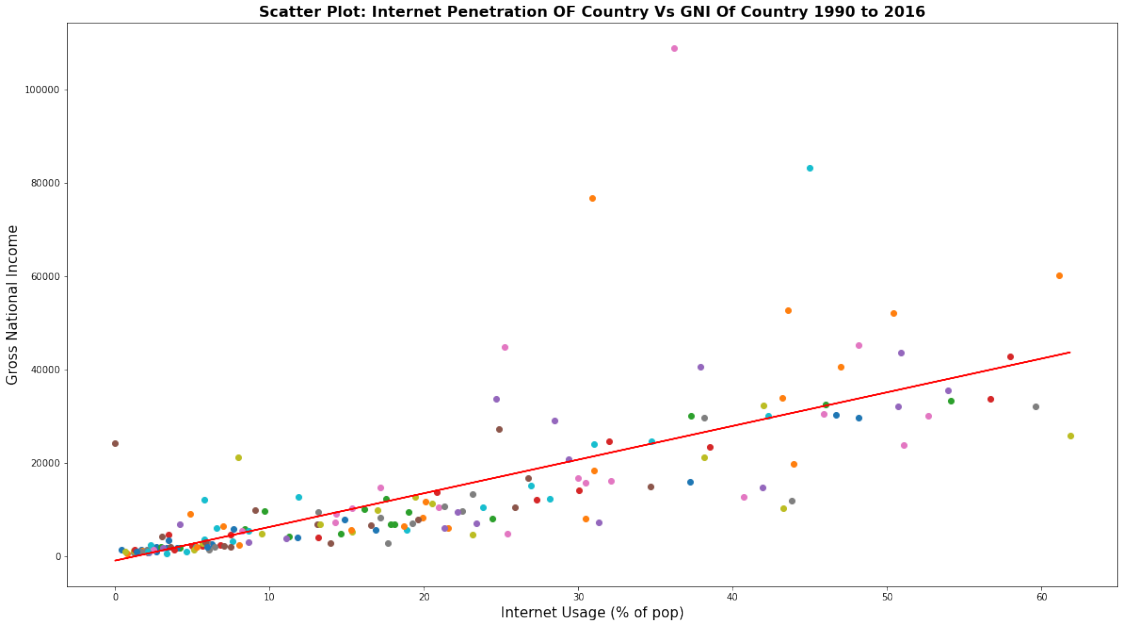
Ok, there's a bit to unpack here. First off, once again, we can see the the black line to depict the moving average GNI throughout the years. Notice how the line is a lot closer towards the bottom, this most likely is indicative that the majority of the countries have a low GNI number. I say this because there must be a large concentration of GNI scores below 20,000 if the average is depecit being within that interval, it would actually be lower if not for the outliers above the average. Take note that there appears to be a lot of countries high in GNI as the years progress, but if the average is depicted being below 20,000, then that must mean that there are a lot more countries low in GNI than high (huge concentration below 20,000, I should have used a violin plot to show the distribution better).

Next, if we take a look as how the years progress, we can see that the GNI distribution looked fairly similar upto 1993, but once past that year, we can notice a huge increae in GNI scores for a lot of countries as if it is following the growth of the internet usage in those countries. Furthermore, the dark, messy (sorry about that) cluster of names represents the names of the countries that were the lowest 50 in internet penetraction. It's shocking to see how all 50 countries with the lowest internet usage (mainly in Africa) also have the lowest GNI at 2016, we can see how they are all below

the mean GNI. This is a bit stronger evidence that a relation between internet use and economic development of a country may be related.

To actually see if a realtion exists, let us create a plot that visualizes internet usage vs GNI score.

[496]:
```
#gets the unique country names in our dataframe
unique_country_names = pd.unique(internet_df['Country Name'])
internet_as_2d_lst = []
gni_list = []

plt.figure(figsize = (20,11))


for country in unique_country_names:
    #gets the sub-dataframe that has all the rows that includes the current␣
 ↪country
    country_internet_df = internet_df.loc[internet_df['Country Name'] ==␣
 ↪country].mean()
    country_gni_df = gni_melt_df.loc[gni_melt_df['Country Name'] == country].
 ↪mean()

    #plots the internet usage of the current country from 1990 to 2016
    plt.scatter(country_internet_df['Internet Usage (% of pop)'],␣
 ↪country_gni_df['GNI'])

    if (not np.isnan(np.min(country_gni_df['GNI']))):
        internet_as_2d_lst.append([country_internet_df['Internet Usage (% of␣
 ↪pop)']])
        gni_list.append([country_gni_df['GNI']])



#creates the linear regression model
lm = linear_model.LinearRegression()

#passing in our dataset so that it can be modeled with linear regression
lm.fit(internet_as_2d_lst, gni_list)


estimated_outputs = lm.predict(internet_as_2d_lst)
plt.plot(internet_as_2d_lst, estimated_outputs, color = 'red')

plt.title("Scatter Plot: Internet Penetration OF Country Vs GNI Of Country 1990␣
 ↪to 2016", fontsize = 16, fontweight = "bold")
plt.xlabel("Internet Usage (% of pop)", fontsize = 15)
plt.ylabel("Gross National Income", fontsize = 15)
plt.show()
```



Wow, we can see from above that there does appear to a positive co-variance / relation between internet usage and a country's economic devlopement.

Sorry, I wasn't able to add/ finish I got short on time. My apologizes.