In [1]:
```
%run Imports.ipynb
```

Introduction

In this report, the pandemic test, case, recovery, death, and vaccination totals are put into a relational database and analyzed. This report uses linear regression to determine the relationship between cases of and vaccinations for the virus, and time-series modeling to determine the progression of cases of and vaccinations for the virus.

Dataset of country pandemic totals.

In [2]:
```
df=Json('df1: Covid19 Totals')
df.web_scrape('https://www.worldometers.info/coronavirus/?referer=app', ['0', 'Country', 'Total_Cases', 'New_Cases', 'Total_Deaths', 'New_Deaths', 'Total_Recovered',
                           '1','Active_Cases', 'Serious_Critical', 'Tot_Cases_per_1m_Pop', 'Tot_Deaths_per_1m_Pop',
                          'Total_Tests', 'Tot_Tests_per_1m_Pop', 'Population', 'Unknown', 'Continent', 'Unknown', 'Unknown'])
df1=json_storage['df1: Covid19 Totals'][1]
df1
```

Out[2]:

	0	Country	Total_Cases	New_Cases	Total_Deaths	New_Deaths	Total_Recovered	1	Active_Cases	Serious_Critical	Tot_Cases_per_1m_Pop	Tot_Deaths_per_1m_Pop	Total_Tests	Tot_Tests_per_1m_Pop	Populatic
0	None	None	None	None	None	None	None	None	None	None	None	None	None	None	Nor
1		\nNorth America\n	35,875,404	+83,199	819,332	+2,001	27,580,673	+97,159	7,475,399	15,373					
2		\nSouth America\n	21,168,897	+136,524	552,665	+4,639	18,894,237	+128,254	1,721,995	21,166					
3		\nAsia\n	28,490,944	+182,032	428,059	+1,434	26,093,254	+112,944	1,969,631	25,421					
4		\nEurope\n	39,609,225	+223,855	913,171	+3,921	27,982,751	+117,852	10,713,303	31,419					
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
233		Total:	39,609,225	+223,855	913,171	+3,921	27,982,751	+117,852	10,713,303	31,419					
234		Total:	4,251,595	+12,213	113,036	+235	3,814,105	+8,308	324,454	3,037					
235		Total:	57,040	+400	1,142	+4	34,567	+21	21,331	6					
236		Total:	721		15		706		0	0					
237		Total:	129,453,826	+638,223	2,827,420	+12,234	104,400,293	+464,538	22,226,113	96,422	16,607.7	362.7			

238 rows × 19 columns

Dataset of USA state pandemic totals.

In [3]:
```
df=Json('df2: USA Covid19 Totals')
df.web_scrape('https://www.worldometers.info/coronavirus/country/us/', ['0', 'Country', 'Total_Cases', 'New_Cases', 'Total_Deaths', 'New_Deaths', 'Total_Recovered',
                          'Active_Cases', 'Tot_Cases_per_1m_Pop', 'Tot_Deaths_per_1m_Pop',
                         'Total_Tests', 'Tot_Tests_per_1m_Pop', 'Population','1','1'])
df2=json_storage['df2: USA Covid19 Totals'][1]
df2
```

Out[3]:

	0	Country	Total_Cases	New_Cases	Total_Deaths	New_Deaths	Total_Recovered	Active_Cases	Tot_Cases_per_1m_Pop	Tot_Deaths_per_1m_Pop	Total_Tests	Tot_Tests_per_1m_Pop	Population	1	
0	None	None	None	None	None	None	None	None	None	None	None	None	None	None	
1		USA Total	31,166,344	+68,756	565,254	+1,113	23,673,462	6,927,628	94,157	1,708	403,326,884	1,218,500			
2	1	\nCalifornia	3,669,253	\n+2,505	\n58,938	+149	\n1,940,569	\n1,669,746	92,864	1,492	\n53,786,487	1,361,262	\n39,512,223	\n[view by county] [1]	\n[pro
3	2	\nTexas	2,795,916	\n+3,058	\n48,504	+25	\n2,648,579	\n98,833	96,425	1,673	\n 25,536,604	880,698	\n28,995,881	\n[view by county] [1] [2] [3]	\n[pro
4	3	\nFlorida	2,057,735	\n+5,294	\n33,450	+84	\n1,520,607	\n503,678	95,808	1,557	\n25,367,915	1,181,126	\n21,477,737	\n[view by county] [1] [2] [3]	\n[pro
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
61	60	\nNavajo Nation	30,095	\n+16	\n1,247		\n16,398	\n12,450			\n254,374\n		\n	\n[1] [2] [3]	
62	61	\nGrand Princess Ship	122	\n	\n7		\n115	\n0			\n		\n	\n[1]	
63	62	\nWuhan Repatriated	3	\n	\n		\n3	\n0			\n3\n		\n	\n	
64	63	\nDiamond Princess Ship	46	\n	\n		\n46	\n0			\n46\n		\n	\n	
65		Total:	31,166,344	+68,756	565,254	+1,113	23,673,462	6,927,628	94,157	1,708	403,326,884	1,218,500			

66 rows × 15 columns

Dataset of country vaccination totals.

In [39]:
```
df=Json('df3: vaccinated_total')
df.csv('https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccinations/vaccinations.csv')
df3=json_storage['df3: vaccinated_total'][1]
df3
```

Out[39]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred	people_fully_vaccinated_pe
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	NaN	NaN	0.00	0.00	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	1367.0	NaN	NaN	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	1367.0	NaN	NaN	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN	1367.0	NaN	NaN	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN	1367.0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
9813	Zimbabwe	ZWE	2021-03-25	54892.0	54892.0	NaN	5488.0	2127.0	0.37	0.37	
9814	Zimbabwe	ZWE	2021-03-26	61093.0	61093.0	NaN	6201.0	2772.0	0.41	0.41	
9815	Zimbabwe	ZWE	2021-03-27	65466.0	65466.0	NaN	4373.0	3322.0	0.44	0.44	
9816	Zimbabwe	ZWE	2021-03-28	68511.0	68511.0	NaN	3045.0	3683.0	0.46	0.46	
9817	Zimbabwe	ZWE	2021-03-29	69751.0	69751.0	NaN	1240.0	3780.0	0.47	0.47	

9818 rows × 12 columns

Dataset of USA state vaccinations.

In [5]:
```
df=Json('df4: usa_vaccinated_total')
df.csv('vaccine_data.csv')
df4=json_storage['df4: usa_vaccinated_total'][1]
df4
```

Out[5]:

	Location	Total COVID-19 Vaccines Delivered	Total COVID-19 Vaccines Administered	Share of Delivered Vaccines That Have Been Administered	Number of People Who Have Been Vaccinated	Share of Population Vaccinated
0	United States	154199235.0	118313818.0	0.76728	77050155.0	0.234738
1	Alabama	2141210.0	1426696.0	0.66630	936863.0	0.191072
2	Alaska	514835.0	356055.0	0.69159	214185.0	0.292784
3	Arizona	3444465.0	2760907.0	0.80155	1714739.0	0.235583
4	Arkansas	1407320.0	952919.0	0.67712	634546.0	0.210267
...	...	...	...	...	...	...
69	*Vaccines delivered* are cumulative counts of ...	NaN	NaN	NaN	NaN	NaN
70	NaN	NaN	NaN	NaN	NaN	NaN
71	*Vaccines administered* are cumulative counts ...	NaN	NaN	NaN	NaN	NaN
72	NaN	NaN	NaN	NaN	NaN	NaN
73	*Number of people who have been vaccinated* re...	NaN	NaN	NaN	NaN	NaN

74 rows × 6 columns

Dataset of country cases time-series.

In [40]:
```
df=Json('df5: time_series')
df.csv('https://covid19.who.int/WHO-COVID-19-global-data.csv')
df5=json_storage['df5: time_series'][1]
df5
```

Out[40]:

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
0	2020-01-03	AF	Afghanistan	EMRO	0	0	0	0
1	2020-01-04	AF	Afghanistan	EMRO	0	0	0	0
2	2020-01-05	AF	Afghanistan	EMRO	0	0	0	0
3	2020-01-06	AF	Afghanistan	EMRO	0	0	0	0
4	2020-01-07	AF	Afghanistan	EMRO	0	0	0	0
...	...	...	...	...	...	...	...	...
107139	2021-03-27	ZW	Zimbabwe	AFRO	27	36805	0	1518
107140	2021-03-28	ZW	Zimbabwe	AFRO	13	36818	1	1519
107141	2021-03-29	ZW	Zimbabwe	AFRO	4	36822	1	1520
107142	2021-03-30	ZW	Zimbabwe	AFRO	17	36839	0	1520
107143	2021-03-31	ZW	Zimbabwe	AFRO	19	36858	1	1521

107144 rows × 8 columns

Dataset of USA state cases time-series.

In [7]:
```
df=Json('df6: usa_time_series_cases_deaths')
df.csv('https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-states.csv')
df6=json_storage['df6: usa_time_series_cases_deaths'][1]
df6
```

Out[7]:

	date	state	fips	cases	deaths
0	2020-01-21	Washington	53	1	0
1	2020-01-22	Washington	53	1	0
2	2020-01-23	Washington	53	1	0
3	2020-01-24	Illinois	17	1	0
4	2020-01-24	Washington	53	1	0
...	...	...	...	...	...
21624	2021-03-30	Virginia	51	617941	10242
21625	2021-03-30	Washington	53	365816	5302
21626	2021-03-30	West Virginia	54	141322	2640
21627	2021-03-30	Wisconsin	55	635523	7293
21628	2021-03-30	Wyoming	56	56236	695

21629 rows × 5 columns

Dataset of government responses to covid 19.

Data Mining

Country pandemic totals data information.

In [112]:
```
df1.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238 entries, 0 to 237
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   0                      237 non-null    object
 1   Country                237 non-null    object
 2   Total_Cases            237 non-null    object
 3   New_Cases              237 non-null    object
 4   Total_Deaths           237 non-null    object
 5   New_Deaths             237 non-null    object
 6   Total_Recovered        237 non-null    object
 7   1                      237 non-null    object
 8   Active_Cases           237 non-null    object
 9   Serious_Critical       237 non-null    object
 10  Tot_Cases_per_1m_Pop   237 non-null    object
 11  Tot_Deaths_per_1m_Pop  237 non-null    object
 12  Total_Tests            237 non-null    object
 13  Tot_Tests_per_1m_Pop   237 non-null    object
 14  Population             237 non-null    object
 15  Unknown                237 non-null    object
 16  Continent              237 non-null    object
 17  Unknown                237 non-null    object
 18  Unknown                237 non-null    object
dtypes: object(19)
memory usage: 35.5+ KB
```

USA state vaccination totals.

In [113]:
```
df2.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74 entries, 0 to 73
Data columns (total 6 columns):
 #   Column                                                Non-Null Count  Dtype  
---  ------                                                --------------  -----  
 0   Location                                              70 non-null     object 
 1   Total COVID-19 Vaccines Delivered                     60 non-null     float64
 2   Total COVID-19 Vaccines Administered                  60 non-null     float64
 3   Share of Delivered Vaccines That Have Been Administered  60 non-null     float64
 4   Number of People Who Have Been Vaccinated             60 non-null     float64
 5   Share of Population Vaccinated                        53 non-null     float64
dtypes: float64(5), object(1)
memory usage: 3.6+ KB
```

Country vaccination time-series data information.

In [60]:
```
df3.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6495 entries, 0 to 6494
Data columns (total 12 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   location                            6495 non-null   object 
 1   iso_code                            6131 non-null   object 
 2   date                                6495 non-null   object 
 3   total_vaccinations                  4229 non-null   float64
 4   people_vaccinated                   3782 non-null   float64
 5   people_fully_vaccinated             2587 non-null   float64
 6   daily_vaccinations_raw              3595 non-null   float64
 7   daily_vaccinations                  6299 non-null   float64
 8   total_vaccinations_per_hundred      4229 non-null   float64
 9   people_vaccinated_per_hundred       3782 non-null   float64
 10  people_fully_vaccinated_per_hundred 2587 non-null   float64
 11  daily_vaccinations_per_million      6299 non-null   float64
dtypes: float64(9), object(3)
memory usage: 609.0+ KB
```

Country cases time-series data information.

In [61]:
```
df4.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103306 entries, 0 to 103305
Data columns (total 8 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   Date_reported    103306 non-null  object
 1   Country_code     102868 non-null  object
 2   Country          103306 non-null  object
 3   WHO_region       103306 non-null  object
 4   New_cases        103306 non-null  int64 
 5   Cumulative_cases 103306 non-null  int64 
 6   New_deaths       103306 non-null  int64 
 7   Cumulative_deaths 103306 non-null  int64 
dtypes: int64(4), object(4)
memory usage: 6.3+ MB
```

USA state cases time-series data information.

In [29]:
```
df7.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20914 entries, 0 to 20913
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   date    20914 non-null  object
 1   states  20914 non-null  object
 2   cases   20914 non-null  int64 
 3   deaths  20914 non-null  int64 
 4   State   20914 non-null  object
dtypes: int64(2), object(3)
memory usage: 817.1+ KB
```

USA state pandemic totals data information.

In [62]:
```
df8.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66 entries, 0 to 65
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   0                      65 non-null     object
 1   Country                65 non-null     object
 2   Total_Cases            65 non-null     object
 3   New_Cases              65 non-null     object
 4   Total_Deaths           65 non-null     object
 5   New_Deaths             65 non-null     object
 6   Total_Recovered        65 non-null     object
 7   Active_Cases           65 non-null     object
 8   Tot_Cases_per_1m_Pop   65 non-null     object
 9   Tot_Deaths_per_1m_Pop  65 non-null     object
 10  Total_Tests            65 non-null     object
 11  Tot_Tests_per_1m_Pop   65 non-null     object
 12  Population             65 non-null     object
 13  1                      65 non-null     object
 14  1                      64 non-null     object
dtypes: object(15)
memory usage: 7.9+ KB
```

Government responses to covid 19 data information.

In [63]:
```
df9.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62700 entries, 0 to 62699
Data columns (total 43 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   country                 62700 non-null  object 
 1   geoid                   62425 non-null  object 
 2   iso                     62700 non-null  object 
 3   d                       62700 non-null  object 
 4   cases                   57750 non-null  float64
 5   deaths                  57750 non-null  float64
 6   school                  45758 non-null  float64
 7   school_local            45758 non-null  float64
 8   domestic                55275 non-null  float64
 9   domestic_local          55275 non-null  float64
 10  travel                  55275 non-null  float64
 11  travel_partial          55275 non-null  float64
 12  travel_dom              55275 non-null  float64
 13  travel_dom_partial      55275 non-null  float64
 14  curf                    55275 non-null  float64
 15  curf_partial            55275 non-null  float64
 16  mass                    55275 non-null  float64
 17  mass_partial            55275 non-null  float64
 18  elect                   14834 non-null  float64
 19  elect_partial           14834 non-null  float64
 20  sport                   55275 non-null  float64
 21  sport_partial           55275 non-null  float64
 22  rest                    55275 non-null  float64
 23  rest_local              55275 non-null  float64
 24  testing                 55275 non-null  float64
 25  testing_narrow          55275 non-null  float64
 26  masks                   51459 non-null  float64
 27  masks_partial           51459 non-null  float64
 28  surveillance            55275 non-null  float64
 29  surveillance_partial    55275 non-null  float64
 30  state                   55275 non-null  float64
 31  state_partial           55275 non-null  float64
 32  cash                    54450 non-null  float64
 33  wage                    54450 non-null  float64
 34  credit                  54450 non-null  float64
 35  taxc                    54450 non-null  float64
 36  taxd                    54450 non-null  float64
 37  export                  54450 non-null  float64
 38  rate                    54450 non-null  float64
 39  Rigidity_Public_Health  55275 non-null  float64
 40  Economic_Measures       54450 non-null  float64
 41  population_2019         57750 non-null  float64
 42  continent               61875 non-null  object 
dtypes: float64(38), object(5)
memory usage: 20.6+ MB
```

Data Cleaning

Cleans country pandemic totals dataframe by droping parts of dataframe that are undesired, replacing undesired values to desired values, and turning some columns into numeric datatype.

In [8]:
```
df1=df1.drop(0, axis=0)
df1=df1.drop(1, axis=0)
df1=df1.drop(2, axis=0)
df1=df1.drop(3, axis=0)
df1=df1.drop(4, axis=0)
df1=df1.drop(5, axis=0)
df1=df1.drop(6, axis=0)
df1=df1.drop(7, axis=0)
df1=df1.drop(230, axis=0)
df1=df1.drop(231, axis=0)
df1=df1.drop(232, axis=0)
df1=df1.drop(233, axis=0)
df1=df1.drop(234, axis=0)
df1=df1.drop(235, axis=0)
df1=df1.drop(236, axis=0)
df1=df1.drop(237, axis=0)
df1=df1.replace('USA', 'United States')
df1=df1.replace('UK', 'United Kingdom')
df1=df1.replace('UAE', 'United Arab Emirates')
df1=df1.replace('S. Korea', 'South Korea')
df1=df1.replace('S. Korea', 'South Korea')
df1=df1.replace('CAR', 'Central African Republic')
df1=df1.replace('CAR', 'Central African Republic')
df1=df1.replace('DRC', 'Democratic Republic of the Congo')
df1=df1.replace(' ', 0)
df1=df1.replace('N/A', 0)
df1['Total_Cases'] = df1['Total_Cases'].str.replace(",","")
df1['Total_Cases'] = pd.to_numeric(df1['Total_Cases'])
df1['New_Cases'] = df1['New_Cases'].str.replace(",","")
df1['New_Cases'] = df1['New_Cases'].str.replace("+","")
df1['New_Cases'] = pd.to_numeric(df1['New_Cases'])
df1['Total_Deaths'] = df1['Total_Deaths'].str.replace(",","")
df1['Total_Deaths'] = pd.to_numeric(df1['Total_Deaths'])
df1['New_Deaths'] = df1['New_Deaths'].str.replace("+","")
df1['New_Deaths'] = df1['New_Deaths'].str.replace(",","")
df1['New_Deaths'] = pd.to_numeric(df1['New_Deaths'])
df1['Total_Recovered'] = df1['Total_Recovered'].str.replace(",","")
df1['Total_Recovered'] = pd.to_numeric(df1['Total_Recovered'])
df1['Active_Cases'] = df1['Active_Cases'].str.replace("+","")
df1['Active_Cases'] = df1['Active_Cases'].str.replace(",","")
df1['Active_Cases'] = pd.to_numeric(df1['Active_Cases'])
df1['Serious_Critical'] = df1['Serious_Critical'].str.replace(",","")
df1['Serious_Critical'] = pd.to_numeric(df1['Serious_Critical'])
df1['Tot_Cases_per_1m_Pop'] = df1['Tot_Cases_per_1m_Pop'].str.replace(",","")
df1['Tot_Cases_per_1m_Pop'] = pd.to_numeric(df1['Tot_Cases_per_1m_Pop'])
df1['Tot_Deaths_per_1m_Pop'] = df1['Tot_Deaths_per_1m_Pop'].str.replace(",","")
df1['Tot_Deaths_per_1m_Pop'] = pd.to_numeric(df1['Tot_Deaths_per_1m_Pop'])
df1['Total_Tests'] = df1['Total_Tests'].str.replace(",","")
df1['Total_Tests'] = pd.to_numeric(df1['Total_Tests'])
df1['Tot_Tests_per_1m_Pop'] = df1['Tot_Tests_per_1m_Pop'].str.replace(",","")
df1['Tot_Tests_per_1m_Pop'] = pd.to_numeric(df1['Tot_Tests_per_1m_Pop'])
df1['Population'] = df1['Population'].str.replace(",","")
df1['Population'] = pd.to_numeric(df1['Population'])
df1=df1.replace(np.nan, 0)
df1
```

Out[8]:

	0	Country	Total_Cases	New_Cases	Total_Deaths	New_Deaths	Total_Recovered	1	Active_Cases	Serious_Critical	Tot_Cases_per_1m_Pop	Tot_Deaths_per_1m_Pop	Total_Tests	Tot_Tests_per_1m_Pop	Popula
8		World	129453826	638223.0	2827420.0	12234.0	104400293.0	+464,538	22226113.0	96422.0	16608.0	362.7	0.0	0.0	0.000000€
9	1	United States	31166344	68756.0	565254.0	1113.0	23673462.0	+86,548	6927628.0	8764.0	93747.0	1700.0	403326884.0	1213186.0	3.324526€
10	2	Brazil	12753258	89200.0	321886.0	3950.0	11169937.0	+95,454	1261435.0	8318.0	59682.0	1506.0	28600000.0	133840.0	2.136876€
11	3	India	12220669	72182.0	162960.0	458.0	11472494.0	+40,442	585215.0	8944.0	8791.0	117.0	243672940.0	175289.0	1.390124€
12	4	France	4644423	41907.0	95640.0	303.0	294638.0	+1,842	4254145.0	5053.0	71036.0	1463.0	63999096.0	978855.0	6.538156€
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
225	217	Marshall Islands	4	0.0	0.0	0.0	4.0		0.0	0.0	67.0	0.0	0.0	0.0	5.948800€
226	218	Samoa	3	0.0	0.0	0.0	2.0		1.0	0.0	15.0	0.0	0.0	0.0	1.993990€
227	219	Vanuatu	3	0.0	0.0	0.0	1.0		2.0	0.0	10.0	0.0	470.0	1504.0	3.124660€
228	220	Micronesia	1	0.0	0.0	0.0	1.0		0.0	0.0	9.0	0.0	0.0	0.0	1.159220€
229	221	China	90201	11.0	4636.0	0.0	85385.0	+4	180.0	2.0	63.0	3.0	160000000.0	111163.0	1.439324€

222 rows × 19 columns

Cleans country vaccinations dataframe by replacing undesired values with desired ones, renaming undesired column names to desired ones, and storing global vaccinations total in a variable wvt.

In [41]:
```
df3=df3.replace(np.nan,0)
df3=df3.rename(columns={'location':'Country'})
wvt=df3.loc[df3['Country']=='World']['total_vaccinations'].iloc[-1]
df3
```

Out[41]:

	Country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred	people_fully_vaccinated_pe
0	Afghanistan	AFG	2021-02-22	0.0	0.0	0.0	0.0	0.0	0.00	0.00	
1	Afghanistan	AFG	2021-02-23	0.0	0.0	0.0	0.0	1367.0	0.00	0.00	
2	Afghanistan	AFG	2021-02-24	0.0	0.0	0.0	0.0	1367.0	0.00	0.00	
3	Afghanistan	AFG	2021-02-25	0.0	0.0	0.0	0.0	1367.0	0.00	0.00	
4	Afghanistan	AFG	2021-02-26	0.0	0.0	0.0	0.0	1367.0	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	...	
9813	Zimbabwe	ZWE	2021-03-25	54892.0	54892.0	0.0	5488.0	2127.0	0.37	0.37	
9814	Zimbabwe	ZWE	2021-03-26	61093.0	61093.0	0.0	6201.0	2772.0	0.41	0.41	
9815	Zimbabwe	ZWE	2021-03-27	65466.0	65466.0	0.0	4373.0	3322.0	0.44	0.44	
9816	Zimbabwe	ZWE	2021-03-28	68511.0	68511.0	0.0	3045.0	3683.0	0.46	0.46	
9817	Zimbabwe	ZWE	2021-03-29	69751.0	69751.0	0.0	1240.0	3780.0	0.47	0.47	

9818 rows × 12 columns

Makes a new dataframe df_stats that the function stats will populate with desired data.

In [10]:
```
df_stats=pd.DataFrame(columns=['Country','Population','Tests', 'Tests/Population','Cases', 'Active_Cases', 'Cases/Tests','Recoveries',
                               'Recoveries/Cases','Deaths','Deaths/Cases','Vaccinations','Vaccinations/Population'])
df_stats
```

Out[10]:

Country	Population	Tests	Tests/Population	Cases	Active_Cases	Cases/Tests	Recoveries	Recoveries/Cases	Deaths	Deaths/Cases	Vaccinations	Vaccinations/Population

In [11]:
```
countries=sorted(df1['Country'].unique())
stats(countries, df1, 'Country', df3, df_stats)
json_storage['df_stats: Country_Stats_df']=df_stats
```

Makes a dataframe containing global pandemic totals.

In [12]:
```
world_df=df_stats.loc[df_stats['Country']=='World']
df_stats=df_stats.drop(218, axis=0)
world_df=world_df.rename(columns={'Country':'Location'})
world_df['Population']=sum(df_stats['Population'])
world_df['Tests']=sum(df_stats['Tests'])
world_df['Tests/Population']=(world_df['Tests']/world_df['Population'])*100
world_df['Active_Cases']=int(world_df['Active_Cases'])
world_df['Cases/Tests']=(world_df['Cases']/world_df['Tests'])*100
world_df['Vaccinations']=int(wvt)
world_df['Vaccinations/Population']=(world_df['Vaccinations']/world_df['Population'])*100

json_storage['world_df: World_Stats_df']=world_df
```

Cleans USA State pandemic totals dataframe by droping parts of dataframe that are undesired, replacing undesired values to desired values, turning some columns into numeric datatype, storing in usa_total variable the total population, and removing white space from the end of each state name.

In [13]:
```
df2=df2.drop(0, axis=0)
df2=df2.drop(53, axis=0)
df2=df2.drop(54, axis=0)
df2=df2.drop(55, axis=0)
df2=df2.drop(56, axis=0)
df2=df2.drop(57, axis=0)
df2=df2.drop(58, axis=0)
df2=df2.drop(59, axis=0)
df2=df2.drop(60, axis=0)
df2=df2.drop(61, axis=0)
df2=df2.drop(62, axis=0)
df2=df2.drop(63, axis=0)
df2=df2.drop(64, axis=0)
df2=df2.drop(65, axis=0)
df2=df2.replace(np.nan, 0)
df2=df2.replace(' ', 0)
df2=df2.replace('N/A', 0)
df2['Country']=df2['Country'].astype('str')
df2['Country'] = df2['Country'].str.replace("\n","")
df2['Total_Cases']=df2['Total_Cases'].astype('str')
df2['Total_Cases'] = df2['Total_Cases'].str.replace(",","")
df2['Total_Cases'] = df2['Total_Cases'].str.replace("\n","")
df2['Total_Cases'] = pd.to_numeric(df2['Total_Cases'])
df2['New_Cases']=df2['New_Cases'].astype('str')
df2['New_Cases'] = df2['New_Cases'].str.replace(",","")
df2['New_Cases'] = df2['New_Cases'].str.replace("+","")
df2['New_Cases'] = df2['New_Cases'].str.replace("\n","")
df2['New_Cases'] = pd.to_numeric(df2['New_Cases'])
df2['Total_Deaths']=df2['Total_Deaths'].astype('str')
df2['Total_Deaths'] = df2['Total_Deaths'].str.replace(",","")
df2['Total_Deaths'] = df2['Total_Deaths'].str.replace("\n","")
df2['Total_Deaths'] = pd.to_numeric(df2['Total_Deaths'])
df2['New_Deaths']=df2['New_Deaths'].astype('str')
df2['New_Deaths'] = df2['New_Deaths'].str.replace("+","")
df2['New_Deaths'] = df2['New_Deaths'].str.replace(",","")
df2['New_Deaths'] = df2['New_Deaths'].str.replace("\n","")
df2['New_Deaths'] = pd.to_numeric(df2['New_Deaths'])
df2['Total_Recovered']=df2['Total_Recovered'].astype('str')
df2['Total_Recovered'] = df2['Total_Recovered'].str.replace(",","")
df2['Total_Recovered'] = df2['Total_Recovered'].str.replace("\n","")
df2['Total_Recovered'] = df2['Total_Recovered'].str.replace("N/A","")
df2['Total_Recovered'] = df2['Total_Recovered'].str.replace(" ",'')
df2['Total_Recovered'] = pd.to_numeric(df2['Total_Recovered'])
df2['Active_Cases']=df2['Active_Cases'].astype('str')
df2['Active_Cases'] = df2['Active_Cases'].str.replace("+","")
df2['Active_Cases'] = df2['Active_Cases'].str.replace(",","")
df2['Active_Cases'] = df2['Active_Cases'].str.replace("\n","")
df2['Active_Cases'] = df2['Active_Cases'].str.replace("N/A","")
df2['Active_Cases'] = df2['Active_Cases'].str.replace(" ",'')
df2['Active_Cases'] = pd.to_numeric(df2['Active_Cases'])
df2['Tot_Cases_per_1m_Pop']=df2['Tot_Cases_per_1m_Pop'].astype('str')
df2['Tot_Cases_per_1m_Pop'] = df2['Tot_Cases_per_1m_Pop'].str.replace(",","")
df2['Tot_Cases_per_1m_Pop'] = df2['Tot_Cases_per_1m_Pop'].str.replace("\n","")
df2['Tot_Cases_per_1m_Pop'] = pd.to_numeric(df2['Tot_Cases_per_1m_Pop'])
df2['Tot_Deaths_per_1m_Pop']=df2['Tot_Deaths_per_1m_Pop'].astype('str')
df2['Tot_Deaths_per_1m_Pop'] = df2['Tot_Deaths_per_1m_Pop'].str.replace(",","")
df2['Tot_Deaths_per_1m_Pop'] = df2['Tot_Deaths_per_1m_Pop'].str.replace("\n","")
```

```
df2['Tot_Deaths_per_1m_Pop'] = pd.to_numeric(df2['Tot_Deaths_per_1m_Pop'])
df2['Total_Tests']=df2['Total_Tests'].astype('str')
df2['Total_Tests'] = df2['Total_Tests'].str.replace(",","")
df2['Total_Tests'] = df2['Total_Tests'].str.replace("\n","")
df2['Total_Tests'] = pd.to_numeric(df2['Total_Tests'])
df2['Tot_Tests_per_1m_Pop']=df2['Tot_Tests_per_1m_Pop'].astype('str')
df2['Tot_Tests_per_1m_Pop'] = df2['Tot_Tests_per_1m_Pop'].str.replace(",","")
df2['Tot_Tests_per_1m_Pop'] = df2['Tot_Tests_per_1m_Pop'].str.replace("\n","")
df2['Tot_Tests_per_1m_Pop'] = pd.to_numeric(df2['Tot_Tests_per_1m_Pop'])
df2['Population']=df2['Population'].astype('str')
df2['Population'] = df2['Population'].str.replace(",","")
df2['Population'] = df2['Population'].str.replace("\n","")
df2['Population'] = pd.to_numeric(df2['Population'])
df2=df2.replace(np.nan, 0)
df2=df2.replace(' ', 0)
df2=df2.replace('N/A', 0)
usa_total=df2.iloc[0]
df2=df2.drop(1, axis=0)
df2=df2.rename(columns={'Country':'State'})
df2['State']=df2['State'].replace('District Of Columbia ','District of Columbia ')
df2['State'] = df2['State'].apply(lambda x: x.strip())
df2
```

Out[13]:

	0	State	Total_Cases	New_Cases	Total_Deaths	New_Deaths	Total_Recovered	Active_Cases	Tot_Cases_per_1m_Pop	Tot_Deaths_per_1m_Pop	Total_Tests	Tot_Tests_per_1m_Pop	Population	
2	1	California	3669253	2505.0	58938	149	1940569.0	1669746.0	92864	1492	53786487	1361262	39512223.0	\n[view by cou
3	2	Texas	2795916	3058.0	48504	25	2648579.0	98833.0	96425	1673	25536604	880698	28995881.0	\n[view by county] [1]
4	3	Florida	2057735	5294.0	33450	84	1520607.0	503678.0	95808	1557	25367915	1181126	21477737.0	\n[view by county] [1]
5	4	New York	1915254	8628.0	50587	87	1124544.0	740123.0	98453	2600	44936152	2309919	19453561.0	\n[view by county] [1]
6	5	Illinois	1244585	2592.0	23579	37	1158104.0	62902.0	98217	1861	20313050	1603010	12671821.0	\n[1] [2] [3]
7	6	Georgia	1059548	1807.0	19055	68	692672.0	347821.0	99793	1795	8291638	780946	10617423.0	\n
8	7	Pennsylvania	1030587	4223.0	25210	42	932619.0	72758.0	80502	1969	11687576	912950	12801989.0	\n[view by cou
9	8	Ohio	1017566	1989.0	18609	0	966333.0	32624.0	87053	1592	10979647	939306	11689100.0	\n[view by county] [1]
10	9	North Carolina	914132	1929.0	12112	25	876108.0	25912.0	87159	1155	11336690	1080911	10488084.0	\n[1] [2]
11	10	New Jersey	908816	3672.0	24561	75	685539.0	198716.0	102319	2765	12100950	1362384	8882190.0	\n[view by cou
12	11	Arizona	841811	733.0	16967	26	803013.0	21831.0	115654	2331	4498406	618022	7278717.0	
13	12	Tennessee	811842	1313.0	11904	10	786959.0	12979.0	118879	1743	7219625	1057174	6829174.0	\n[1] [2]
14	13	Michigan	746351	7107.0	17119	11	569460.0	159772.0	74733	1714	12065261	1208114	9986857.0	\n[1] [2]
15	14	Indiana	686497	1044.0	13039	1	655004.0	18454.0	101972	1937	8903743	1322557	6732219.0	\n
16	15	Massachusetts	635580	2499.0	17185	37	586484.0	31911.0	92213	2493	18821008	2730649	6892503.0	
17	16	Virginia	618976	1035.0	10252	10	51481.0	557243.0	72518	1201	8396642	983730	8535519.0	\n
18	17	Wisconsin	577195	563.0	6622	10	563534.0	7039.0	99133	1137	3309569	568417	5822434.0	\n[1] [2] [3] [4]
19	18	Missouri	577008	698.0	9081	0	428548.0	139379.0	94015	1480	6191242	1008768	6137428.0	\n[1] [2] [3] [4] [5]
20	19	South Carolina	551630	961.0	9146	20	0.0	0.0	107139	1776	6756672	1312303	5148714.0	\n[1]
21	20	Minnesota	519529	1648.0	6916	12	499395.0	13218.0	92121	1226	8211019	1455949	5639632.0	\n
22	21	Alabama	515388	408.0	10554	27	315743.0	189091.0	105113	2152	2534623	516934	4903185.0	
23	22	Colorado	462081	1763.0	6238	12	328754.0	127089.0	80240	1083	2725439	473270	5758736.0	
24	23	Louisiana	444933	519.0	10141	9	429935.0	4857.0	95709	2181	6495793	1397307	4648794.0	\n[view by cou
25	24	Oklahoma	438364	390.0	4953	103	422816.0	10595.0	110783	1252	3783639	956196	3956971.0	\n[1]
26	25	Kentucky	426876	803.0	6090	25	49678.0	371108.0	95548	1363	4949269	1107796	4467673.0	\n[1] [2]
27	26	Maryland	411344	1366.0	8286	13	9862.0	393196.0	68039	1371	8766709	1450078	6045680.0	
28	27	Utah	385641	514.0	2122	4	373856.0	9663.0	120289	662	4235928	1321267	3205958.0	\n[1]
29	28	Iowa	379388	571.0	5743	14	332958.0	40687.0	120247	1820	4447633	1409678	3155070.0	\n
30	29	Washington	367228	1094.0	5300	10	179778.0	182150.0	48225	696	5912472	776435	7614893.0	\n[view by cou
31	30	Arkansas	330398	212.0	5626	10	323032.0	1740.0	109483	1864	3383520	1121186	3017804.0	\n[1] [2] [3]
32	31	Connecticut	310888	832.0	7886	1	271294.0	31708.0	87199	2212	7696039	2158603	3565287.0	\n
33	32	Mississippi	305146	288.0	7032	19	292872.0	5242.0	102530	2363	2610281	877067	2976149.0	\n[1]
34	33	Nevada	303762	268.0	5249	10	280217.0	18296.0	98619	1704	2950952	958053	3080156.0	\n
35	34	Kansas	303675	0.0	4913	11	290550.0	8212.0	104237	1686	1318593	452609	2913314.0	\n[1] [2] [3] [4] [5] [6] [7] [8] |

	0	State	Total_Cases	New_Cases	Total_Deaths	New_Deaths	Total_Recovered	Active_Cases	Tot_Cases_per_1m_Pop	Tot_Deaths_per_1m_Pop	Total_Tests	Tot_Tests_per_1m_Pop	Population	
36	35	Nebraska	209346	0.0	2180	0	162560.0	44606.0	108222	1127	1002240	518112	1934408.0	\r
37	36	New Mexico	191652	275.0	3937	5	173701.0	14014.0	91401	1878	3000869	1431146	2096829.0	\r
38	37	Idaho	180536	460.0	1962	5	100936.0	77638.0	101024	1098	1186846	664131	1787065.0	\n[1]
39	38	Oregon	165012	442.0	2383	2	147935.0	14694.0	39123	565	4191940	993884	4217737.0	\n[1]
40	39	West Virginia	141738	416.0	2676	36	132784.0	6278.0	79088	1493	2447894	1365900	1792147.0	
41	40	Rhode Island	137329	564.0	2619	1	8300.0	126410.0	129634	2472	3518593	3321430	1059361.0	\r
42	41	South Dakota	117759	264.0	1935	0	113302.0	2522.0	133112	2187	444576	502539	884659.0	
43	42	Montana	104552	231.0	1437	0	102203.0	912.0	97824	1345	1197763	1120685	1068778.0	\n[1] [2]
44	43	North Dakota	103091	211.0	1466	0	100485.0	1140.0	135279	1924	415765	545579	762062.0	\r
45	44	Delaware	94802	229.0	1559	0	86673.0	6570.0	97356	1601	660836	678641	973764.0	\r
46	45	New Hampshire	84176	462.0	1238	1	79825.0	3113.0	61907	910	1901623	1398549	1359711.0	\r
47	46	Alaska	60333	0.0	309	0	52369.0	7655.0	82473	422	1896375	2592288	731545.0	\r
48	47	Wyoming	56310	74.0	700	5	55132.0	478.0	97294	1209	685954	1185215	578759.0	\r
49	48	Maine	50504	251.0	743	5	12963.0	36798.0	37571	553	2090730	1555357	1344212.0	\r
50	49	District of Columbia	44513	100.0	1064	3	31512.0	11937.0	63072	1508	1375475	1948958	705749.0	
51	50	Hawaii	29681	100.0	463	1	0.0	0.0	20963	327	1268077	895616	1415872.0	\n[1] [2] [3]
52	51	Vermont	19275	166.0	227	2	15866.0	3182.0	30890	364	1332173	2134930	623989.0	

Cleans state vaccination totals by dropping undesired columns and renaming undesired column names to desired ones.

In [14]:
```
df4=df4.drop('Total COVID-19 Vaccines Delivered', axis=1)
df4=df4.drop('Share of Delivered Vaccines That Have Been Administered', axis=1)
df4=df4.drop('Number of People Who Have Been Vaccinated', axis=1)
df4=df4.drop('Share of Population Vaccinated', axis=1)
df4=df4.rename(columns={'Location':'State'})
df4=df4.rename(columns={'Total COVID-19 Vaccines Administered':'Vaccines_Administered'})
df4
```

Out[14]:

	State	Vaccines_Administered
0	United States	118313818.0
1	Alabama	1426696.0
2	Alaska	356055.0
3	Arizona	2760907.0
4	Arkansas	952919.0
...	...	...
69	*Vaccines delivered* are cumulative counts of ...	NaN
70	NaN	NaN
71	*Vaccines administered* are cumulative counts ...	NaN
72	NaN	NaN
73	*Number of people who have been vaccinated* re...	NaN

74 rows × 2 columns

Makes a new dataframe df_usa_stats that the function usa_stats will populate with desired data.

In [15]:
```
df_usa_stats=pd.DataFrame(columns=['State','Population','Tests', 'Tests/Population','Cases', 'Active_Cases', 'Cases/Tests','Recoveries',
                          'Recoveries/Cases','Deaths','Deaths/Cases','Vaccinations','Vaccinations/Population'])
df_usa_stats
```

Out[15]:

State	Population	Tests	Tests/Population	Cases	Active_Cases	Cases/Tests	Recoveries	Recoveries/Cases	Deaths	Deaths/Cases	Vaccinations	Vaccinations/Population

In [16]:
```
states=sorted(df2['State'].unique())
stats(states, df2, 'State', df4, df_usa_stats)
```

Cleans country cases time-series dataframe by renaming undesired name to desired one.

In [42]:
```
df5=df5.rename(columns={'Date_reported':'Date'})
df5
```

Out[42]:

	Date	Country_code	Country	WHO_region	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
0	2020-01-03	AF	Afghanistan	EMRO	0	0	0	0
1	2020-01-04	AF	Afghanistan	EMRO	0	0	0	0
2	2020-01-05	AF	Afghanistan	EMRO	0	0	0	0
3	2020-01-06	AF	Afghanistan	EMRO	0	0	0	0
4	2020-01-07	AF	Afghanistan	EMRO	0	0	0	0
...	...	...	...	...	...	...	...	...
107139	2021-03-27	ZW	Zimbabwe	AFRO	27	36805	0	1518
107140	2021-03-28	ZW	Zimbabwe	AFRO	13	36818	1	1519
107141	2021-03-29	ZW	Zimbabwe	AFRO	4	36822	1	1520
107142	2021-03-30	ZW	Zimbabwe	AFRO	17	36839	0	1520
107143	2021-03-31	ZW	Zimbabwe	AFRO	19	36858	1	1521

107144 rows × 8 columns

Cleans USA state cases time-series dataframe by renaming state name values with with initials so that coloropleth function recognizes them using map method, dropping undesired column, and renaming column to desired name.

In [18]:
```
state = {
        'Alabama': 'AL',
        'Alaska': 'AK',
        'American Samoa': 'AS',
        'Arizona': 'AZ',
        'Arkansas': 'AR',
        'California': 'CA',
        'Colorado': 'CO',
        'Connecticut': 'CT',
        'Delaware': 'DE',
        'District of Columbia': 'DC',
        'Florida': 'FL',
        'Georgia': 'GA',
        'Guam': 'GU',
        'Hawaii': 'HI',
        'Idaho': 'ID',
        'Illinois': 'IL',
        'Indiana': 'IN',
        'Iowa': 'IA',
        'Kansas': 'KS',
        'Kentucky': 'KY',
        'Louisiana': 'LA',
        'Maine': 'ME',
        'Maryland': 'MD',
        'Massachusetts': 'MA',
        'Michigan': 'MI',
        'Minnesota': 'MN',
        'Mississippi': 'MS',
        'Missouri': 'MO',
        'Montana': 'MT',
        'Nebraska': 'NE',
        'Nevada': 'NV',
        'New Hampshire': 'NH',
        'New Jersey': 'NJ',
        'New Mexico': 'NM',
        'New York': 'NY',
        'North Carolina': 'NC',
        'North Dakota': 'ND',
        'Northern Mariana Islands':'MP',
        'Ohio': 'OH',
        'Oklahoma': 'OK',
        'Oregon': 'OR',
        'Pennsylvania': 'PA',
        'Puerto Rico': 'PR',
        'Rhode Island': 'RI',
        'South Carolina': 'SC',
        'South Dakota': 'SD',
        'Tennessee': 'TN',
        'Texas': 'TX',
        'Utah': 'UT',
        'Vermont': 'VT',
        'Virgin Islands': 'VI',
        'Virginia': 'VA',
        'Washington': 'WA',
        'West Virginia': 'WV',
        'Wisconsin': 'WI',
        'Wyoming': 'WY'}
```

```
df6['State'] = df6['state'].map(state)
df6=df6.drop('fips', axis=1)
df6=df6.rename(columns={'state':'states'})
df6
```

Out[18]:

	date	states	cases	deaths	State
0	2020-01-21	Washington	1	0	WA
1	2020-01-22	Washington	1	0	WA
2	2020-01-23	Washington	1	0	WA
3	2020-01-24	Illinois	1	0	IL
4	2020-01-24	Washington	1	0	WA
...	...	...	...	...	...
21624	2021-03-30	Virginia	617941	10242	VA
21625	2021-03-30	Washington	365816	5302	WA
21626	2021-03-30	West Virginia	141322	2640	WV
21627	2021-03-30	Wisconsin	635523	7293	WI
21628	2021-03-30	Wyoming	56236	695	WY

21629 rows × 5 columns

Set index to date column for time series data.

In [43]:
```
df5['Date'] = pd.to_datetime(df5['Date'])
df5.set_index('Date', inplace=True)
df3['date'] = pd.to_datetime(df3['date'])
df3.set_index('date', inplace=True)
```

Makes a dataframe that holds the total virus cases time series.

In [44]:
```
global_cases_df=df5.loc[df5['Country']=='Afghanistan']['Cumulative_cases']
for i in df5['Country'].unique():
    if i=='Afganistan':
        continue
    else:
        dd=df5.loc[df5['Country']==i]['Cumulative_cases']
        global_cases_df+=dd
global_cases_df
```

Out[44]:
```
Date
2020-01-03            0
2020-01-04            1
2020-01-05            1
2020-01-06            4
2020-01-07            4
                   ...
2021-03-27    125863438
2021-03-28    126456023
2021-03-29    126987247
2021-03-30    127424100
2021-03-31    127933916
Name: Cumulative_cases, Length: 454, dtype: int64
```

Makes a dataframe that holds the total virus vaccination time series.

In [45]:
```
global_vaccinations_df=df3.loc[df3['Country']=='World']['people_fully_vaccinated']
global_vaccinations_df
```

Out[45]:
```
date
2020-12-13            0.0
2020-12-14            0.0
2020-12-15            0.0
2020-12-16            0.0
2020-12-17            0.0
                    ...
2021-03-26    116309735.0
2021-03-27    118768359.0
2021-03-28    120832732.0
2021-03-29    124654695.0
2021-03-30    126779833.0
Name: people_fully_vaccinated, Length: 108, dtype: float64
```

Makes a sql database that contains cleaned dataframes as tables.

In [35]:
```
conn = sql.connect('covid19.db')
```

In [93]:
```
df_stats.to_sql('country_stats', conn)
```

In [94]:
```
df_usa_stats.to_sql('usa_state_stats', conn)
```

In [95]:
```
pd.read_sql('''SELECT name from sqlite_master where type="table";''', conn)
```

Out[95]:

	name
0	country_stats
1	usa_state_stats

Data Exploration

24.1% of the global population has been tested for the virus. 6.8% of the tested global tested have been diagnosed with the virus. 80.6% of global cases have recovered and 2.2% have died from the virus. 7.2% of the global population has been vaccinated.

In [97]:
```
world_df
```

Out[97]:

	Location	Population	Tests	Tests/Population	Cases	Active_Cases	Cases/Tests	Recoveries	Recoveries/Cases	Deaths	Deaths/Cases	Vaccinations	Vaccinations/Population
218	World	7815306159	1886790963	24.142253	128734212	22154126	6.822919	103766203	80.604993	2813883	2.185808	564561593	7.223794

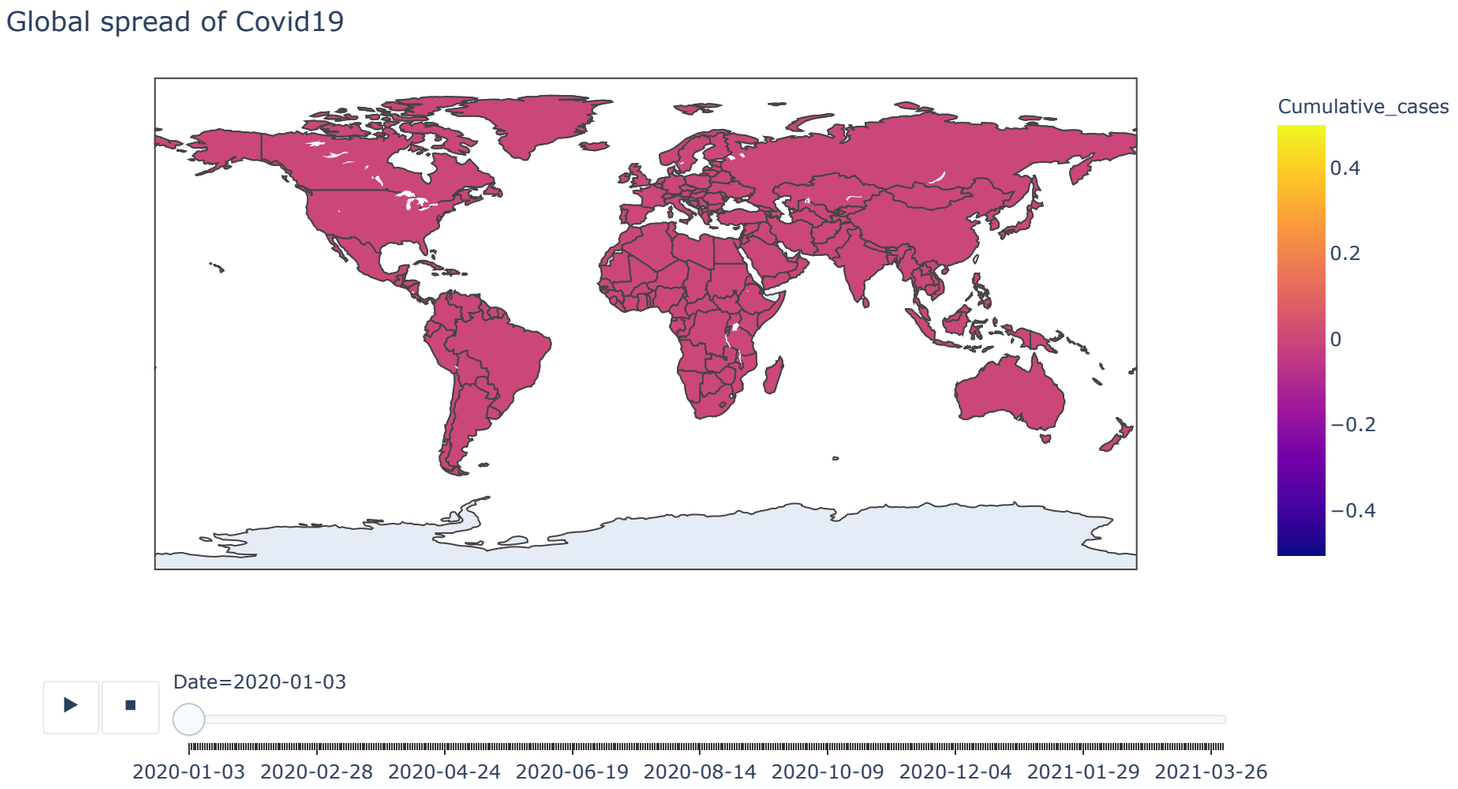
In [ ]:

In [ ]:

In [ ]:

The United States has the most cases of the virus.

In [33]:
```
graph1=graph()
graph1.choropleth(df5, 'Country', 'country names', 'Cumulative_cases', df5.index, 'world', 'Global spread of Covid19')
```



In [107]:
```
desc(df_stats['Population'])
```

```
Population
Count: 221
Mean: 35363376.28506787
Standard Deviation: 139675340.05062267
Min: 0
Q1 quantile:  628125.0
Q2 quantile:  6510483.0
Q3 quantile:  23848794.0
Max: 1439323776
```

Selects countries with larger populations than average country and fewer cases tha average country.

In [233]:
```
pd.read_sql('''SELECT Country, Population, Cases FROM country_stats
               WHERE Population > (SELECT AVG(Population) FROM country_stats) AND Cases<(SELECT AVG(Cases) FROM country_stats)
               ORDER BY Population DESC;''', conn)
```

Out[233]:

	Country	Population	Cases
0	China	1439323776	90190
1	Nigeria	209896709	162641
2	Japan	126188355	470175
3	Ethiopia	117057673	204521
4	Egypt	103757682	200739
5	Vietnam	97987233	2594
6	Democratic Republic of the Congo	91558337	28076
7	Thailand	69930037	28821
8	Tanzania	60983198	509
9	Myanmar	54680205	142412
10	Kenya	54643416	132646
11	South Korea	51301970	102582
12	Uganda	46799838	40839
13	Sudan	44602643	29825
14	Algeria	44435162	116946
15	Afghanistan	39573734	56454
16	Morocco	37234187	495421

In [109]:
```
desc(df_stats['Tests'])
```

```
Tests
Count: 221
Mean: 8537515.669683257
Standard Deviation: 35658632.49301704
Min: 0
Q1 quantile:  65827.0
Q2 quantile:  620810.0
Q3 quantile:  3539320.0
Max: 401946739
```

In [205]:
```
desc(df_stats['Tests/Population'])
```

```
Tests/Population
Count: 221
Mean: 33.432348755114575
Standard Deviation: 35.11752363793047
Min: 0.0
Q1 quantile:  3.126226593925527
Q2 quantile:  18.86441799275805
Q3 quantile:  56.7955491303749
Max: 99.99
```

The United States likely has the most cases of the virus than other countries because it has tested more than other countries.

In [36]:
```
pd.read_sql('''SELECT Country, Tests FROM country_stats
               ORDER BY Tests DESC
               LIMIT 10;''', conn)
```

Out[36]:

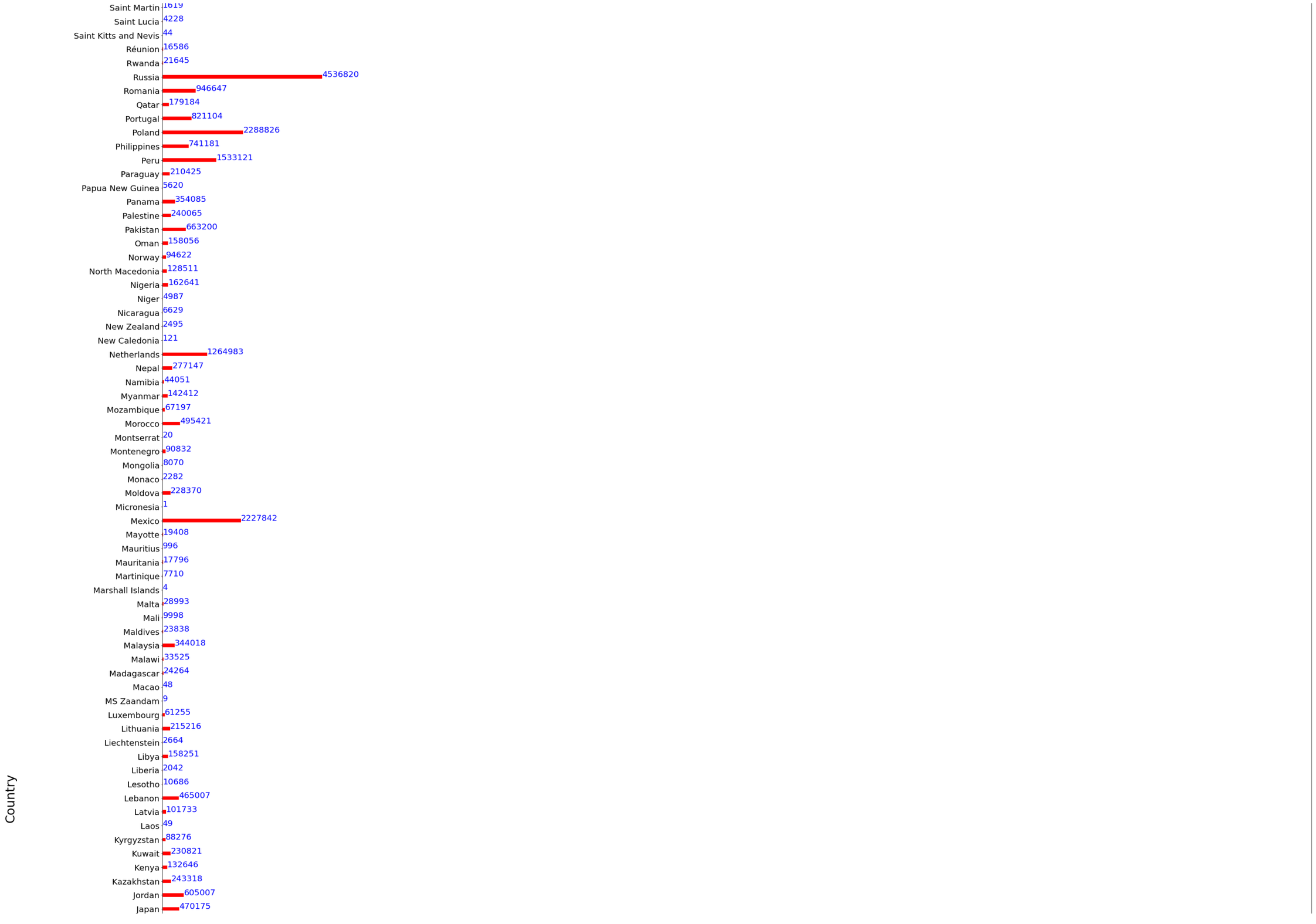
	Country	Tests
0	United States	401946739
1	India	242650025
2	China	160000000
3	United Kingdom	124452321
4	Russia	119900000
5	France	63999096
6	Italy	49551436
7	Germany	48979281
8	Spain	42707830
9	Turkey	38338045

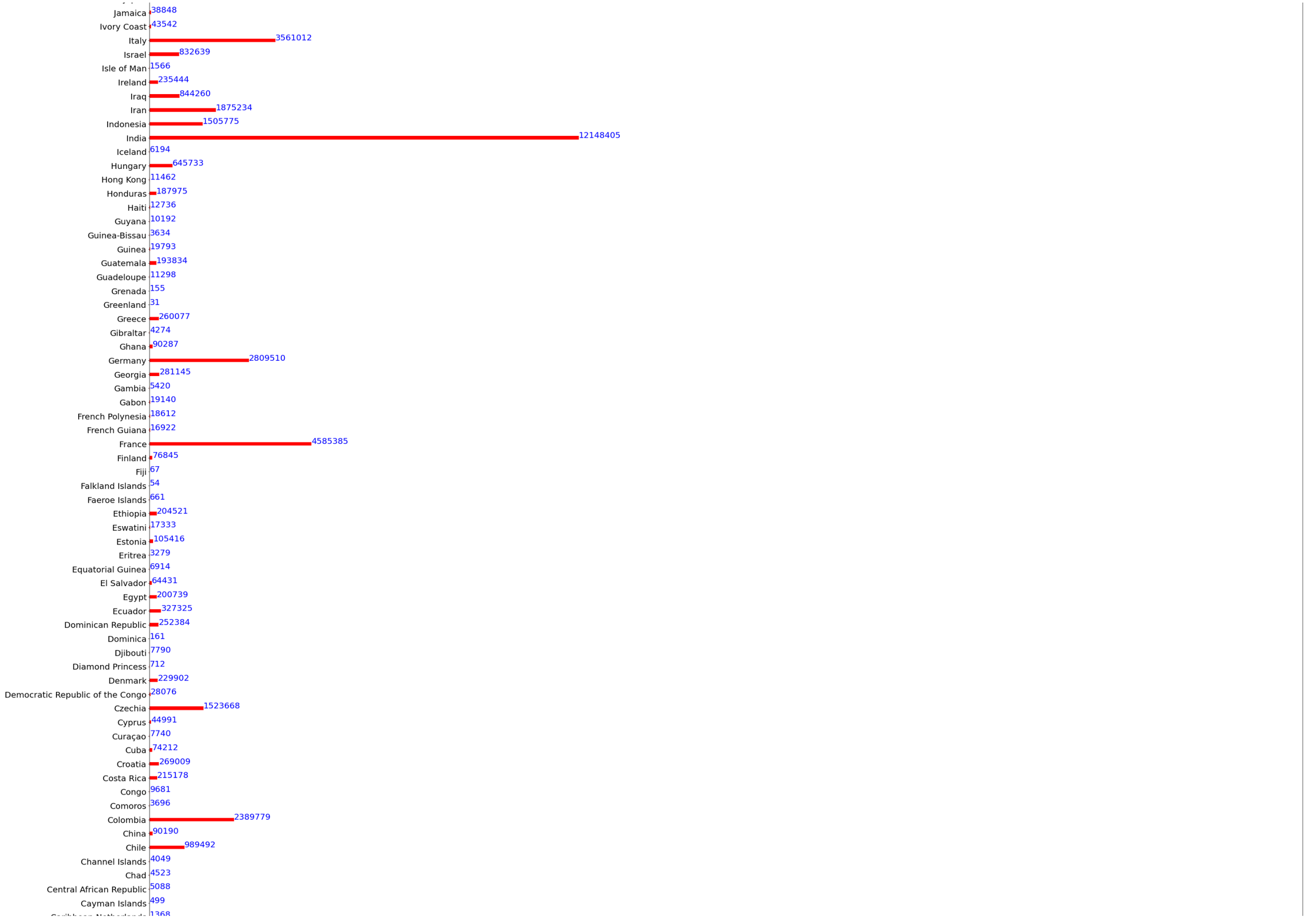
In [113]:
```
desc(df_stats['Cases'])
```

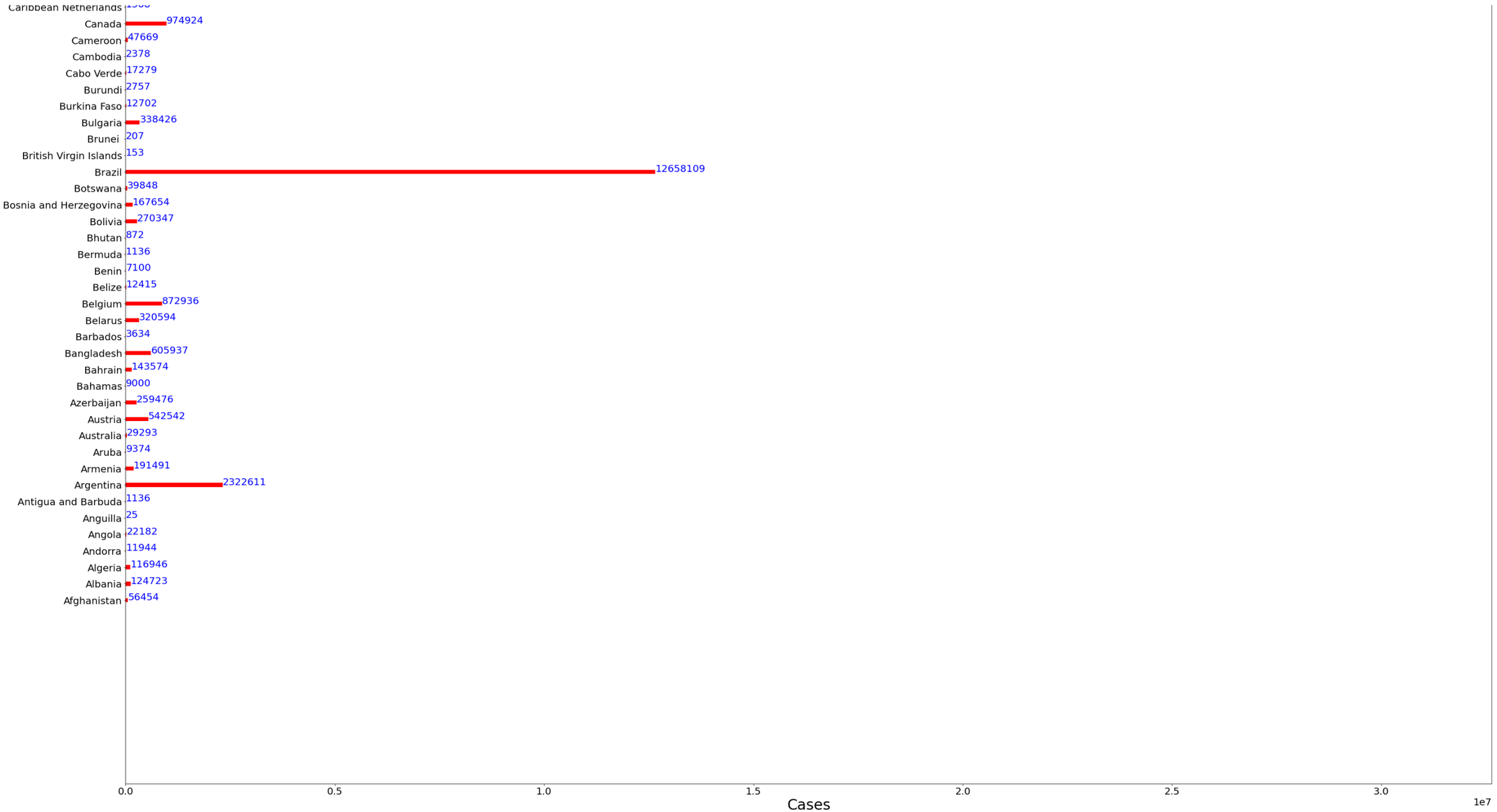
```
Cases
Count: 221
Mean: 582507.7466063348
Standard Deviation: 2480007.4734795145
Min: 1
Q1 quantile:  4228.0
Q2 quantile:  36839.0
Q3 quantile:  252171.0
Max: 31086179
```

In [114]:
```
graph5=graph()
graph5.bar_chart(df_stats['Country'], df_stats['Cases'], 'Country Cases', 'Cases', 'Country')
```


Country Cases
Zimbabwe 36839
Zambia 88199
Yemen 4247
Western Sahara 10
Wallis and Futuna 392
Vietnam 2594
Venezuela 157943
Vatican City 27
Vanuatu 3
Uzbekistan 82682
Uruguay 99584
United States 31086179
United Kingdom 4341736
United Arab Emirates 459360
Ukraine 1662942
Uganda 40839
Turks and Caicos 2327
Turkey 3277880
Tunisia 252171
Trinidad and Tobago 7986
Togo 10249
Timor-Leste 563
Thailand 28821
Tanzania 509
Tajikistan 13308
Taiwan 1024
Syria 18775
Switzerland 598713
Sweden 796445
Suriname 9104
Sudan 29825
St. Vincent Grenadines 1739
St. Barth 775
Sri Lanka 92442
Spain 3275819
South Sudan 10119
South Korea 102582
South Africa 1546735
Somalia 11292
Solomon Islands 19
Slovenia 214045
Slovakia 359330
Sint Maarten 2134
Singapore 60347
Sierra Leone 3970
Seychelles 4135
Serbia 595489
Senegal 38618
Saudi Arabia 389422
Sao Tome and Principe 2221
San Marino 4678
Samoa 3
Saint Pierre Miquelon 24


Country
Japan 470175
Jordan 605007
Kazakhstan 243318
Kenya 132646
Kuwait 230821
Kyrgyzstan 88276
Laos 49
Latvia 101733
Lebanon 465007
Lesotho 10686
Liberia 2042
Libya 158251
Liechtenstein 2664
Lithuania 215216
Luxembourg 61255
MS Zaandam 9
Macao 48
Madagascar 24264
Malawi 33525
Malaysia 344018
Maldives 23838
Mali 9998
Malta 28993
Marshall Islands 4
Martinique 7710
Mauritania 17796
Mauritius 996
Mayotte 19408
Mexico 2227842
Micronesia 1
Moldova 228370
Monaco 2282
Mongolia 8070
Montenegro 90832
Montserrat 20
Morocco 495421
Mozambique 67197
Myanmar 142412
Namibia 44051
Nepal 277147
Netherlands 1264983
New Caledonia 121
New Zealand 2495
Nicaragua 6629
Niger 4987
Nigeria 162641
North Macedonia 128511
Norway 94622
Oman 158056
Pakistan 663200
Palestine 240065
Panama 354085
Papua New Guinea 5620
Paraguay 210425
Peru 1533121
Philippines 741181
Poland 2288826
Portugal 821104
Qatar 179184
Romania 946647
Russia 4536820
Rwanda 21645
Réunion 16586
Saint Kitts and Nevis 44
Saint Lucia 4228
Saint Martin 1619


Jamaica 38848
Ivory Coast 43542
Italy 3561012
Israel 832639
Isle of Man 1566
Ireland 235444
Iraq 844260
Iran 1875234
Indonesia 1505775
India 12148405
Iceland 6194
Hungary 645733
Hong Kong 11462
Honduras 187975
Haiti 12736
Guyana 10192
Guinea-Bissau 3634
Guinea 19793
Guatemala 193834
Guadeloupe 11298
Grenada 155
Greenland 31
Greece 260077
Gibraltar 4274
Ghana 90287
Germany 2809510
Georgia 281145
Gambia 5420
Gabon 19140
French Polynesia 18612
French Guiana 16922
France 4585385
Finland 76845
Fiji 67
Falkland Islands 54
Faeroe Islands 661
Ethiopia 204521
Eswatini 17333
Estonia 105416
Eritrea 3279
Equatorial Guinea 6914
El Salvador 64431
Egypt 200739
Ecuador 327325
Dominican Republic 252384
Dominica 161
Djibouti 7790
Diamond Princess 712
Denmark 229902
Democratic Republic of the Congo 28076
Czechia 1523668
Cyprus 44991
Curaçao 7740
Cuba 74212
Croatia 269009
Costa Rica 215178
Congo 9681
Comoros 3696
Colombia 2389779
China 90190
Chile 989492
Channel Islands 4049
Chad 4523
Central African Republic 5088
Cayman Islands 499



In [115]:
```
desc(df_stats['Active_Cases'])
```

```
Active_Cases
Count: 221
Mean: 91154.82805429865
Standard Deviation: 556770.0224245826
Min: 0
Q1 quantile:  216.0
Q2 quantile:  2548.0
Q3 quantile:  20787.0
Max: 6948028
```

In [216]:
```
pd.read_sql('''SELECT Country, Active_Cases FROM country_stats
                ORDER BY Active_Cases DESC
                LIMIT 10;''', conn)
```

Out[216]:

	Country	Active_Cases
0	United States	6948028
1	France	4197252
2	Brazil	1371216
3	Belgium	793295
4	Italy	562832
5	India	553874
6	Poland	388235
7	United Kingdom	379848
8	Ukraine	323448
9	Russia	282382

In [117]:
```
desc(df_stats['Cases/Tests'])
```

```
Cases/Tests
Count: 221
Mean: 9.311129392984794
Standard Deviation: 10.57910265675415
Min: 0.0
Q1 quantile:  1.90079008585828
Q2 quantile:  7.312139686363121
Q3 quantile:  12.840935999827781
Max: 79.51923076923076
```

In [119]:
```
desc(df_stats['Recoveries'])
```

```
Recoveries
Count: 221
Mean: 469428.1583710407
Standard Deviation: 1977647.8892793
Min: 0
Q1 quantile:  2801.0
Q2 quantile:  25444.0
Q3 quantile:  213590.0
Max: 23574225
```

In [121]:
```
desc(df_stats['Recoveries/Cases'])
```

```
Recoveries/Cases
Count: 221
Mean: 81.70323256192705
Standard Deviation: 22.465763813300406
Min: 0.0
Q1 quantile:  79.23560108840752
Q2 quantile:  90.94063222821896
Q3 quantile:  94.77160154100164
Max: 100.0
```

In [123]:
```
desc(df_stats['Deaths'])
```

```
Deaths
Count: 221
Mean: 12732.502262443439
Standard Deviation: 49453.05612650376
Min: 0
Q1 quantile:  56.0
Q2 quantile:  536.0
Q3 quantile:  4585.0
Max: 563926
```

In [125]:
```
desc(df_stats['Deaths/Cases'])
```

```
Deaths/Cases
Count: 221
Mean: 1.940678362827644
Standard Deviation: 2.384017192118266
Min: 0.0
Q1 quantile:  0.7595365375777073
Q2 quantile:  1.4604354753053639
Q3 quantile:  2.394331207421303
Max: 22.22222222222222
```

In [127]:
```
desc(df_stats['Vaccinations'])
```

```
Vaccinations
Count: 221
Mean: 2553901.040723982
Standard Deviation: 13281198.043055674
Min: 0
Q1 quantile:  0.0
Q2 quantile:  37780.0
Q3 quantile:  475100.0
Max: 145812835
```

Countries that have more vaccinations than active cases.

In [128]:
```
pd.read_sql('''SELECT Country, Active_Cases, Vaccinations FROM country_stats
               WHERE Vaccinations>Active_Cases
               ORDER BY Vaccinations DESC;''', conn)
```

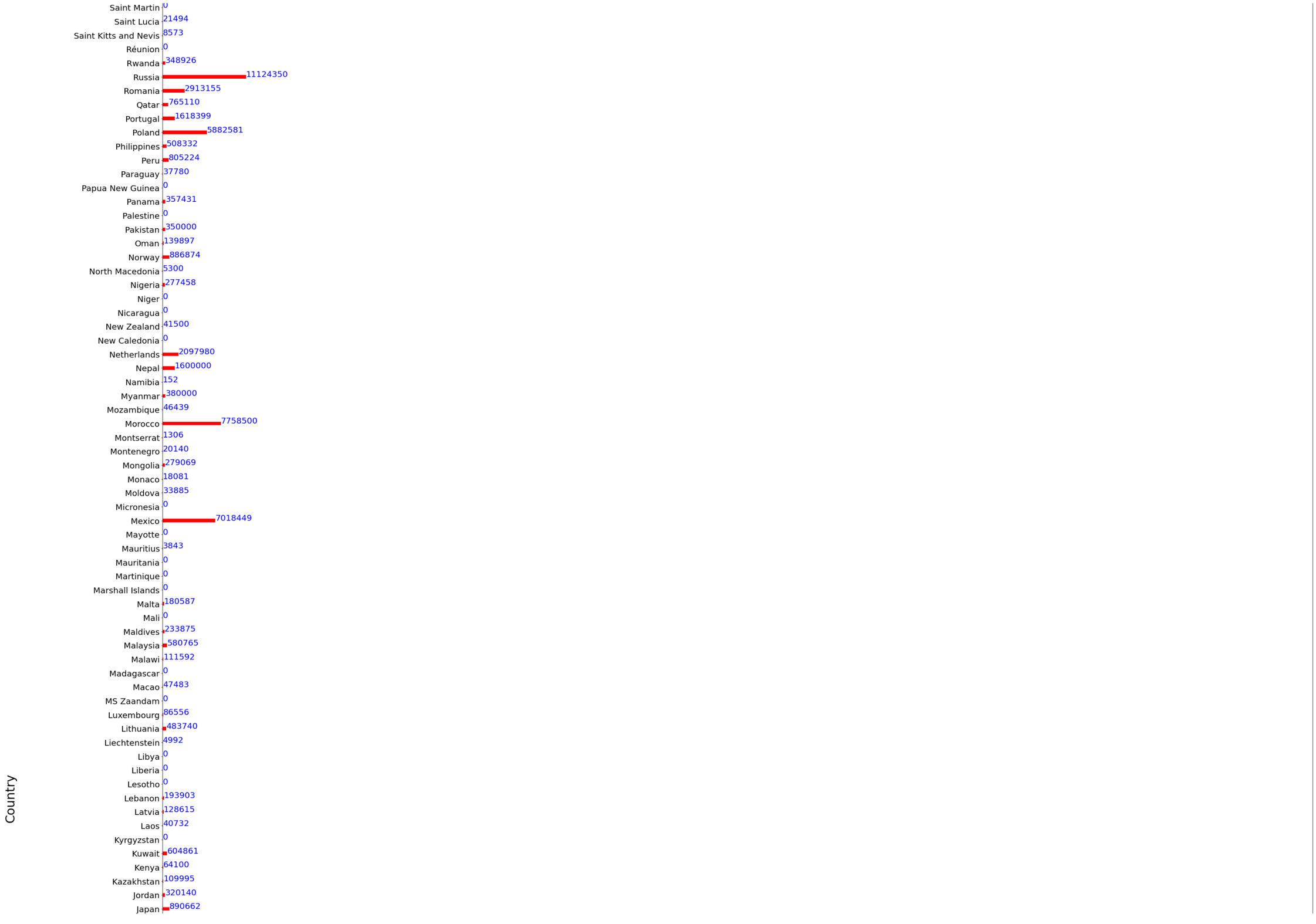
Out[128]:

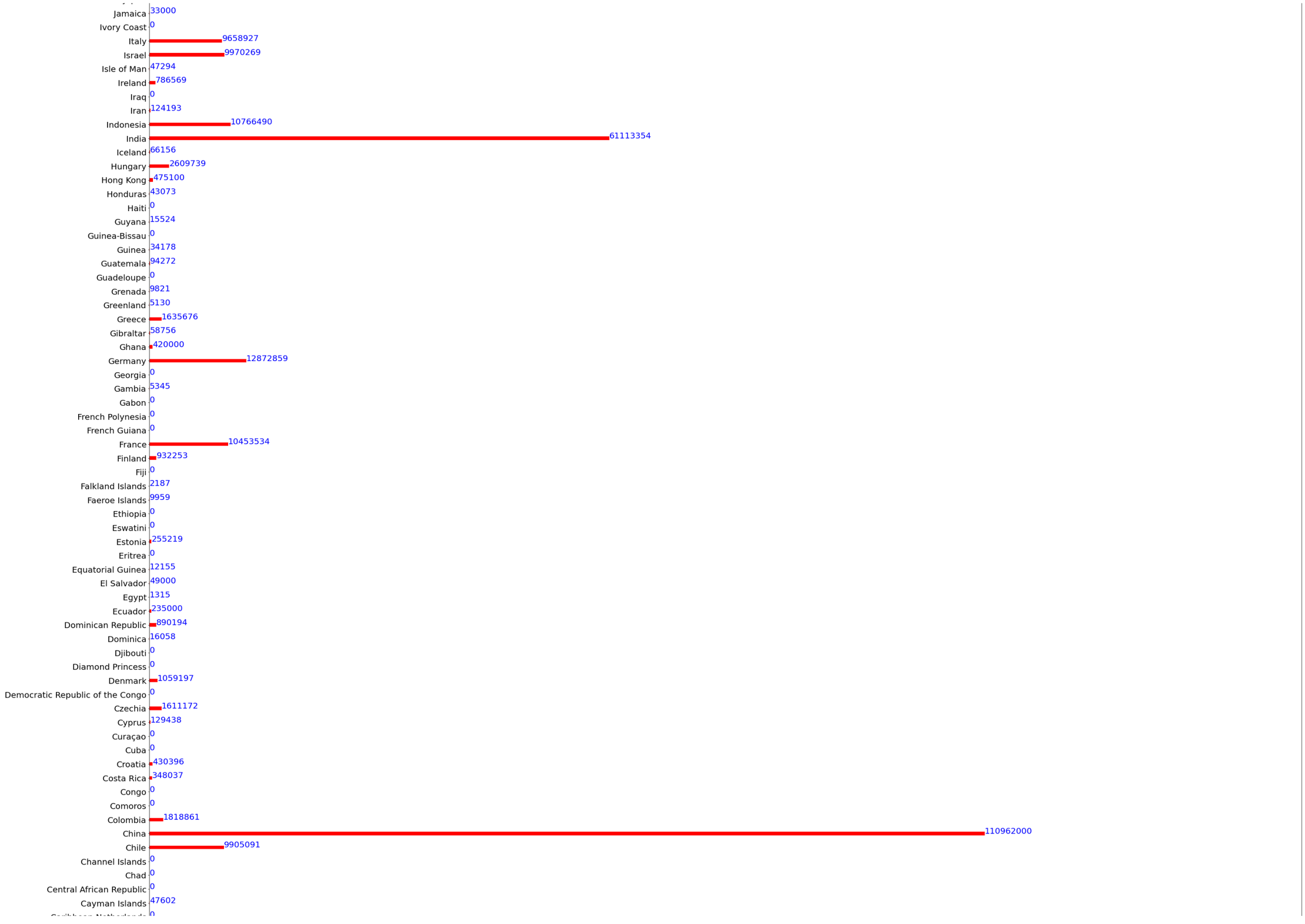
	Country	Active_Cases	Vaccinations
0	United States	6948028	145812835
1	China	173	110962000
2	India	553874	61113354
3	United Kingdom	379848	34119095
4	Brazil	1371216	18082153
...	...	...	...
133	Liechtenstein	46	4992
134	Mauritius	343	3843
135	Falkland Islands	0	2187
136	Montserrat	0	1306
137	Trinidad and Tobago	245	991

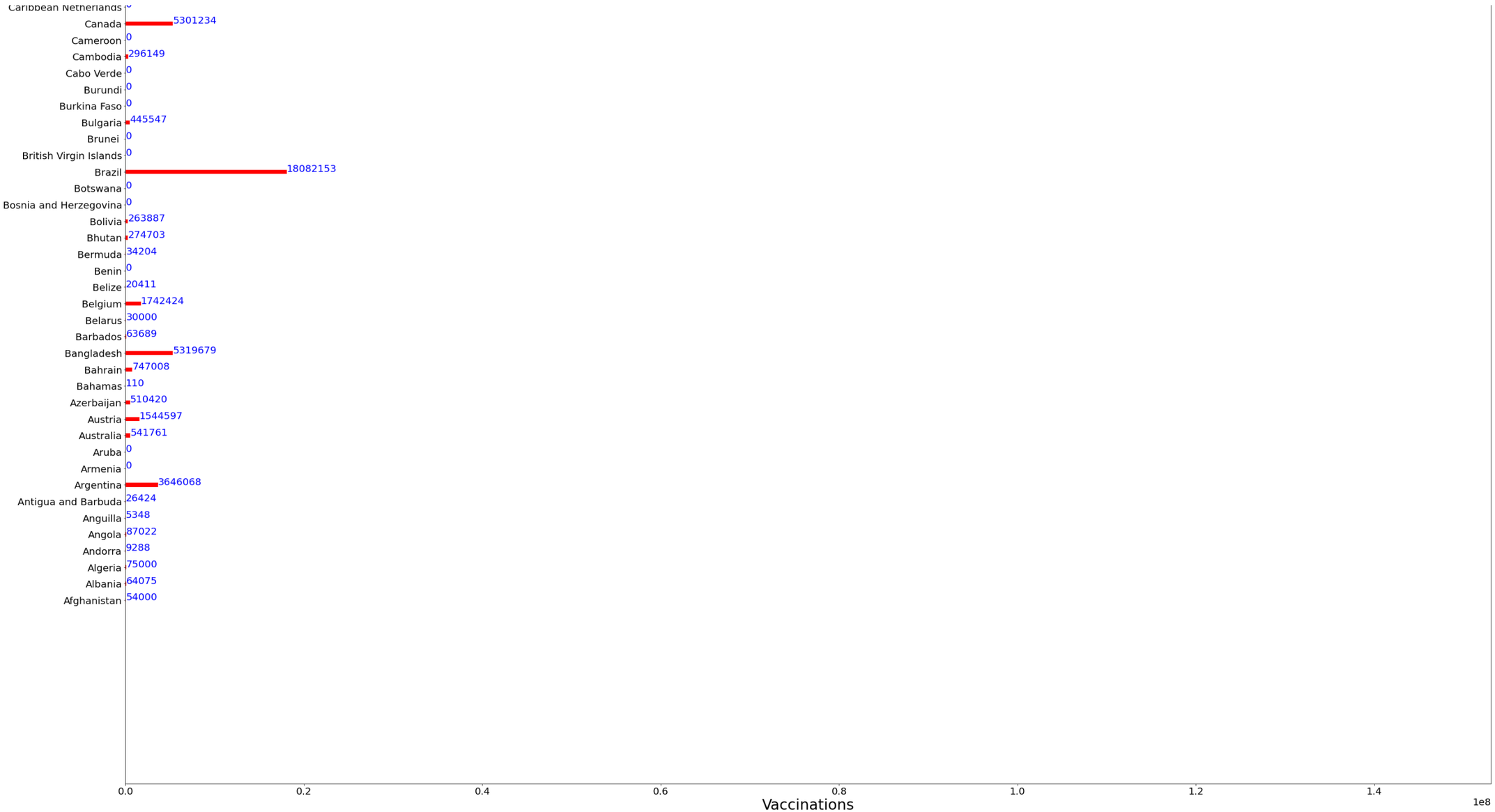
138 rows × 3 columns

In [129]:
```
graph12=graph()
graph12.bar_chart(df_stats['Country'], df_stats['Vaccinations'], 'Country Vaccinations', 'Vaccinations', 'Country')
```


Country Vaccinations
Zimbabwe 69751
Zambia 0
Yemen 0
Western Sahara 0
Wallis and Futuna 0
Vietnam 44000
Venezuela 12194
Vatican City 0
Vanuatu 0
Uzbekistan 0
Uruguay 563281
United States 145812835
United Kingdom 34119095
United Arab Emirates 8133841
Ukraine 212898
Uganda 60139
Turks and Caicos 0
Turkey 14949260
Tunisia 44311
Trinidad and Tobago 991
Togo 42092
Timor-Leste 0
Thailand 102050
Tanzania 0
Tajikistan 0
Taiwan 9412
Syria 0
Switzerland 1342748
Sweden 1482153
Suriname 23304
Sudan 0
St. Vincent Grenadines 0
St. Barth 0
Sri Lanka 894365
Spain 7571439
South Sudan 0
South Korea 828599
South Africa 239665
Somalia 0
Solomon Islands 0
Slovenia 295780
Slovakia 899660
Sint Maarten 0
Singapore 1109000
Sierra Leone 10673
Seychelles 96523
Serbia 2323689
Senegal 234099
Saudi Arabia 4241555
Sao Tome and Principe 0
San Marino 10308
Samoa 0
Saint Pierre Miquelon 0


Country
Saint Martin 0
Saint Lucia 21494
Saint Kitts and Nevis 8573
Réunion 0
Rwanda 348926
Russia 11124350
Romania 2913155
Qatar 765110
Portugal 1618399
Poland 5882581
Philippines 508332
Peru 805224
Paraguay 37780
Papua New Guinea 0
Panama 357431
Palestine 0
Pakistan 350000
Oman 139897
Norway 886874
North Macedonia 5300
Nigeria 277458
Niger 0
Nicaragua 0
New Zealand 41500
New Caledonia 0
Netherlands 2097980
Nepal 1600000
Namibia 152
Myanmar 380000
Mozambique 46439
Morocco 7758500
Montserrat 1306
Montenegro 20140
Mongolia 279069
Monaco 18081
Moldova 33885
Micronesia 0
Mexico 7018449
Mayotte 0
Mauritius 3843
Mauritania 0
Martinique 0
Marshall Islands 0
Malta 180587
Mali 0
Maldives 233875
Malaysia 580765
Malawi 111592
Madagascar 0
Macao 47483
MS Zaandam 0
Luxembourg 86556
Lithuania 483740
Liechtenstein 4992
Libya 0
Liberia 0
Lesotho 0
Lebanon 193903
Latvia 128615
Laos 40732
Kyrgyzstan 0
Kuwait 604861
Kenya 64100
Kazakhstan 109995
Jordan 320140
Japan 890662

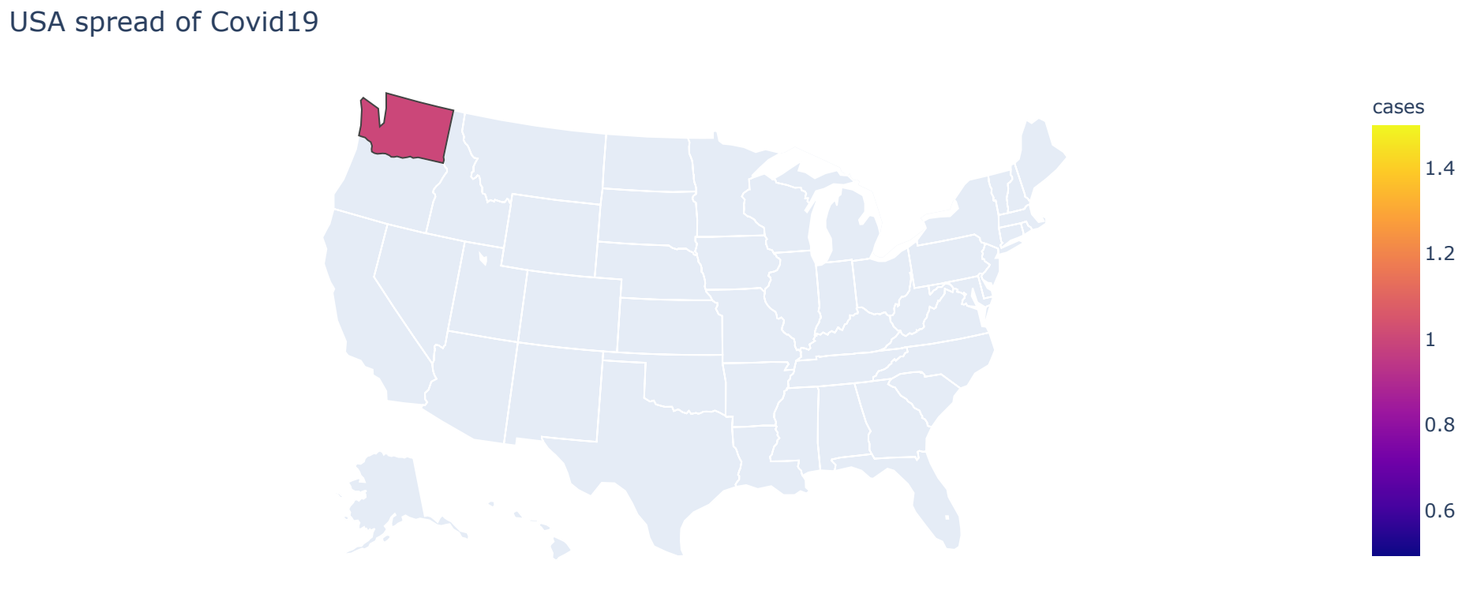

Jamaica 33000
Ivory Coast 0
Italy 9658927
Israel 9970269
Isle of Man 47294
Ireland 786569
Iraq 0
Iran 124193
Indonesia 10766490
India 61113354
Iceland 66156
Hungary 2609739
Hong Kong 475100
Honduras 43073
Haiti 0
Guyana 15524
Guinea-Bissau 0
Guinea 34178
Guatemala 94272
Guadeloupe 0
Grenada 9821
Greenland 5130
Greece 1635676
Gibraltar 58756
Ghana 420000
Germany 12872859
Georgia 0
Gambia 5345
Gabon 0
French Polynesia 0
French Guiana 0
France 10453534
Finland 932253
Fiji 0
Falkland Islands 2187
Faeroe Islands 9959
Ethiopia 0
Eswatini 0
Estonia 255219
Eritrea 0
Equatorial Guinea 12155
El Salvador 49000
Egypt 1315
Ecuador 235000
Dominican Republic 890194
Dominica 16058
Djibouti 0
Diamond Princess 0
Denmark 1059197
Democratic Republic of the Congo 0
Czechia 1611172
Cyprus 129438
Curaçao 0
Cuba 0
Croatia 430396
Costa Rica 348037
Congo 0
Comoros 0
Colombia 1818861
China 110962000
Chile 9905091
Channel Islands 0
Chad 0
Central African Republic 0
Cayman Islands 47602



In [130]:
```
desc(df_stats['Vaccinations/Population'])
```

```
Vaccinations/Population
Count: 221
Mean: 9.106300982497189
Standard Deviation: 17.315025255455687
Min: 0.0
Q1 quantile:  0.0
Q2 quantile:  0.5806679769595254
Q3 quantile:  13.663205982033064
Max: 99.99
```

In [132]:
```
graph14=graph()
graph14.choropleth(df6, 'State', 'USA-states', 'cases', 'date', 'usa', 'USA spread of Covid19')
```



In [133]:
```
desc(df_usa_stats['Population'])
```

```
Population
Count: 51
Mean: 6436069.0784313725
Standard Deviation: 7360660.467814472
Min: 578759
Q1 quantile:  1789606.0
Q2 quantile:  4467673.0
Q3 quantile:  7446805.0
Max: 39512223
```

In [135]:
```
desc(df_usa_stats['Tests'])
```

```
Tests
Count: 51
Mean: 7799215.019607843
Standard Deviation: 10344328.926277911
Min: 415165
Q1 quantile:  1974464.0
Q2 quantile:  4219331.0
Q3 quantile:  8555160.0
Max: 53684932
```

In [137]:
```
desc(df_usa_stats['Tests/Population'])
```

```
Tests/Population
Count: 51
Mean: 89.09704823192001
Standard Deviation: 17.95355830734586
Min: 45.12071132737494
Q1 quantile:  87.88821399793724
Q2 quantile:  99.99
Q3 quantile:  99.99
Max: 99.99
```

In [139]:
```
desc(df_usa_stats['Cases'])
```

```
Cases
Count: 51
Mean: 593752.9215686275
Standard Deviation: 696554.5233106406
Min: 19109
Q1 quantile:  152743.0
Q2 quantile:  409978.0
Q3 quantile:  712348.5
Max: 3665712
```

USA states that have more cases than the global average.

In [140]:
```
pd.read_sql('''SELECT State, us.Cases, AVG(c.Cases) AS AVG_Country_Cases FROM usa_state_stats AS us
               INNER JOIN country_stats AS c
               WHERE us.Cases>(SELECT AVG(Cases) FROM country_stats)
               GROUP BY State
               ORDER BY us.Cases DESC;''', conn)
```

Out[140]:

	State	Cases	AVG_Country_Cases
0	California	3665712	582507.746606
1	Texas	2792526	582507.746606
2	Florida	2052441	582507.746606
3	New York	1905416	582507.746606
4	Illinois	1241993	582507.746606
5	Georgia	1057741	582507.746606
6	Pennsylvania	1026364	582507.746606
7	Ohio	1015577	582507.746606
8	North Carolina	912203	582507.746606
9	New Jersey	905144	582507.746606
10	Arizona	841078	582507.746606
11	Tennessee	809692	582507.746606
12	Michigan	739244	582507.746606
13	Indiana	685453	582507.746606
14	Massachusetts	633081	582507.746606
15	Virginia	617941	582507.746606

In [142]:
```
desc(df_usa_stats['Active_Cases'])
```

```
Active_Cases
Count: 51
Mean: 127061.0
Standard Deviation: 273073.6189250071
Min: 0
Q1 quantile:  6728.0
Q2 quantile:  22385.0
Q3 quantile:  127955.0
Max: 1668426
```

In [144]:
```
desc(df_usa_stats['Cases/Tests'])
```

```
Cases/Tests
Count: 51
Mean: 9.538101156508498
Standard Deviation: 5.989973810731985
Min: 1.4486194563337016
Q1 quantile:  5.96619370943851
Q2 quantile:  8.170816627620518
Q3 quantile:  11.449936925406222
Max: 26.48687885625917
```

In [146]:
```
desc(df_usa_stats['Recoveries'])
```

```
Recoveries
Count: 51
Mean: 444745.0
Standard Deviation: 520181.4595664478
Min: 0
Q1 quantile:  93407.5
Q2 quantile:  292872.0
Q3 quantile:  619436.0
Max: 2645990
```

In [148]:
```
desc(df_usa_stats['Recoveries/Cases'])
```

```
Recoveries/Cases
Count: 51
Mean: 74.96929470760719
Standard Deviation: 30.26098848716949
Min: 0.0
Q1 quantile:  67.4453636906641
Q2 quantile:  90.46235058906976
Q3 quantile:  95.74451352367961
Max: 98.01744082576971
```

In [150]:
```
desc(df_usa_stats['Deaths'])
```

```
Deaths
Count: 51
Mean: 10758.843137254902
Standard Deviation: 12998.423240335806
Min: 225
Q1 quantile:  2147.5
Q2 quantile:  6208.0
Q3 quantile:  12562.5
Max: 58770
```

In [152]:
```
desc(df_usa_stats['Deaths/Cases'])
```

```
Deaths/Cases
Count: 51
Mean: 1.6933454018414336
Standard Deviation: 0.4916554344871608
Min: 0.5175012560710098
Q1 quantile:  1.4242130564196027
Q2 quantile:  1.6468785905783225
Q3 quantile:  1.9642166328700568
Max: 2.708658133793306
```

In [154]:
```
desc(df_usa_stats['Vaccinations'])
```

```
Vaccinations
Count: 51
Mean: 2289585.5490196077
Standard Deviation: 2534229.140847499
Min: 220735
Q1 quantile:  667738.0
Q2 quantile:  1585046.0
Q3 quantile:  2830693.5
Max: 13882984
```

In [156]:
```
desc(df_usa_stats['Vaccinations/Population'])
```

```
Vaccinations/Population
Count: 51
Mean: 37.3155827527672
Standard Deviation: 4.4378244287272155
Min: 27.415645020453645
Q1 quantile:  34.570600742724004
Q2 quantile:  36.95886329202409
Q3 quantile:  39.81495585780323
Max: 49.17911761044892
```

Feature Engineering

In [394]:
```
df=df_stats.copy()
```

Encode Countries.

In [395]:
```
df['Country']=df['Country'].astype('category')
df['Country']=df['Country'].cat.codes
```

Scale the data.

$$\frac{x - min(x)}{max(x) - min(x)}$$

In [396]:
```
pipeline = Pipeline([
    ('min_max_scaler',MinMaxScaler())
])

scaled = pipeline.fit_transform(df)
scaled_df = pd.DataFrame(scaled,columns=df.columns)
```

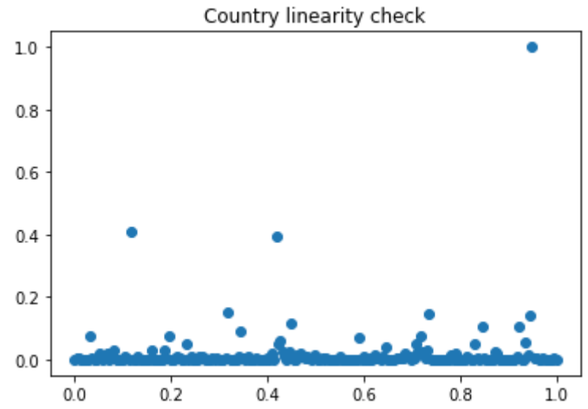
Define dependent and independent variables.

In [397]:
```
y=scaled_df['Cases']
X=scaled_df.drop(['Cases'], axis=1)
```

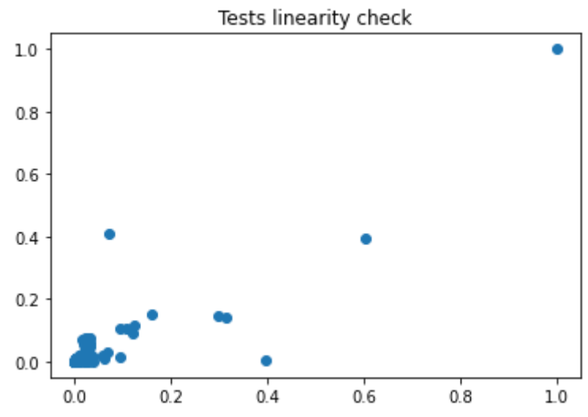
Ordinary least squares has the assumtions of linearity between dependent and indepependent variables, no multicolinearity between independent variables, normality of residuals, and homoskedasticity.

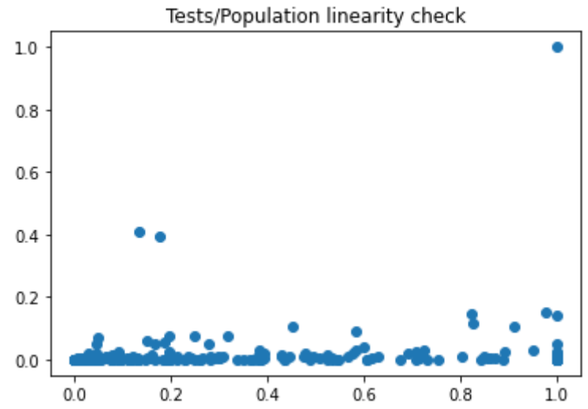
Check the linearity assumtion.

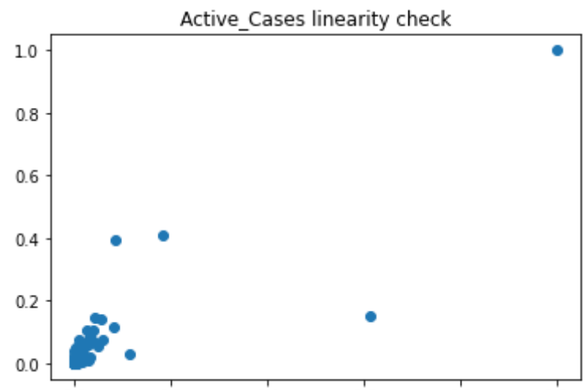
In [398]:
```
for columnName, columnData in X.iteritems():
    plt.scatter(columnData, y)
    plt.title(f"{columnName} linearity check")
    plt.show()
```

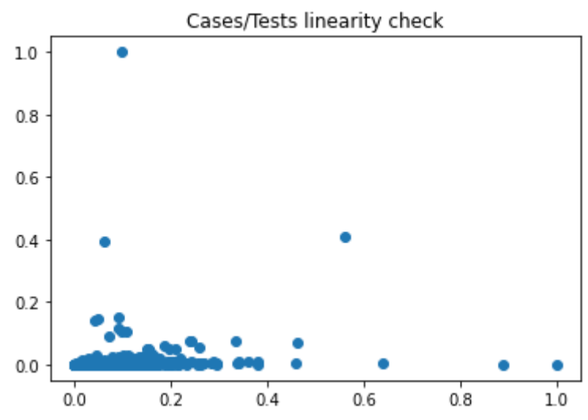

Country linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

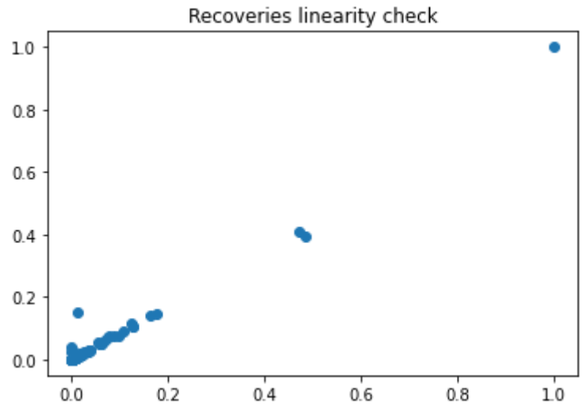
Population linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

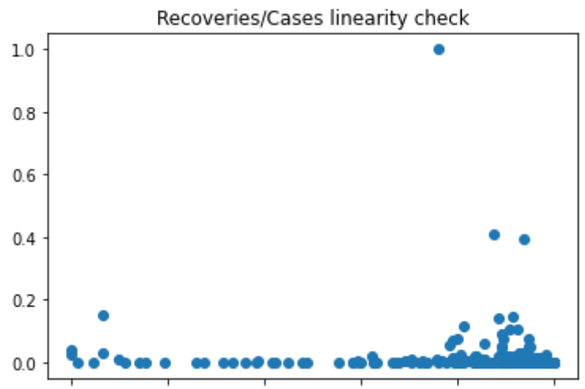

Tests linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

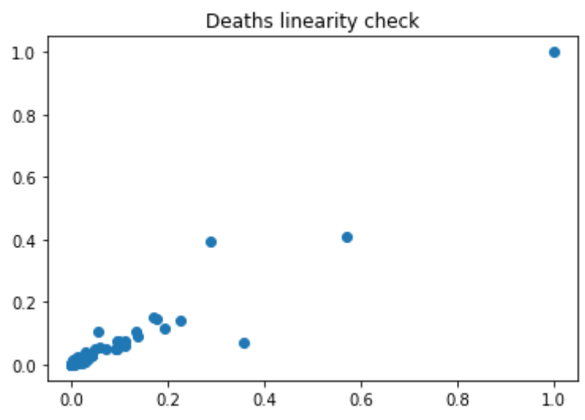

Tests/Population linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

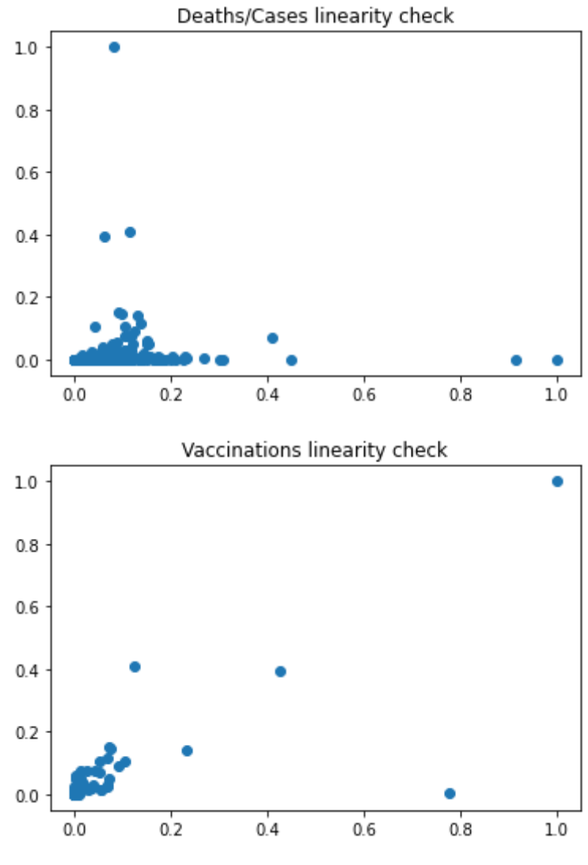

Active_Cases linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

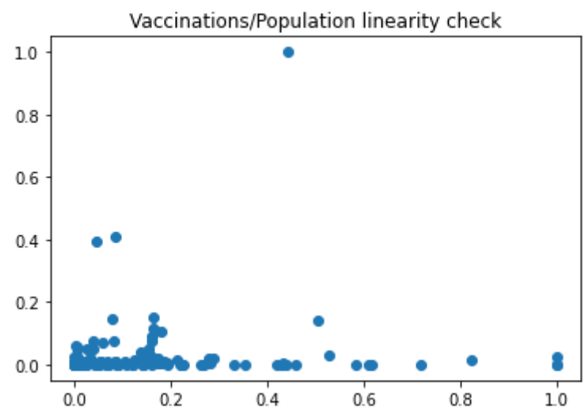

Cases/Tests linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Recoveries linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Recoveries/Cases linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Deaths linearity check
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0





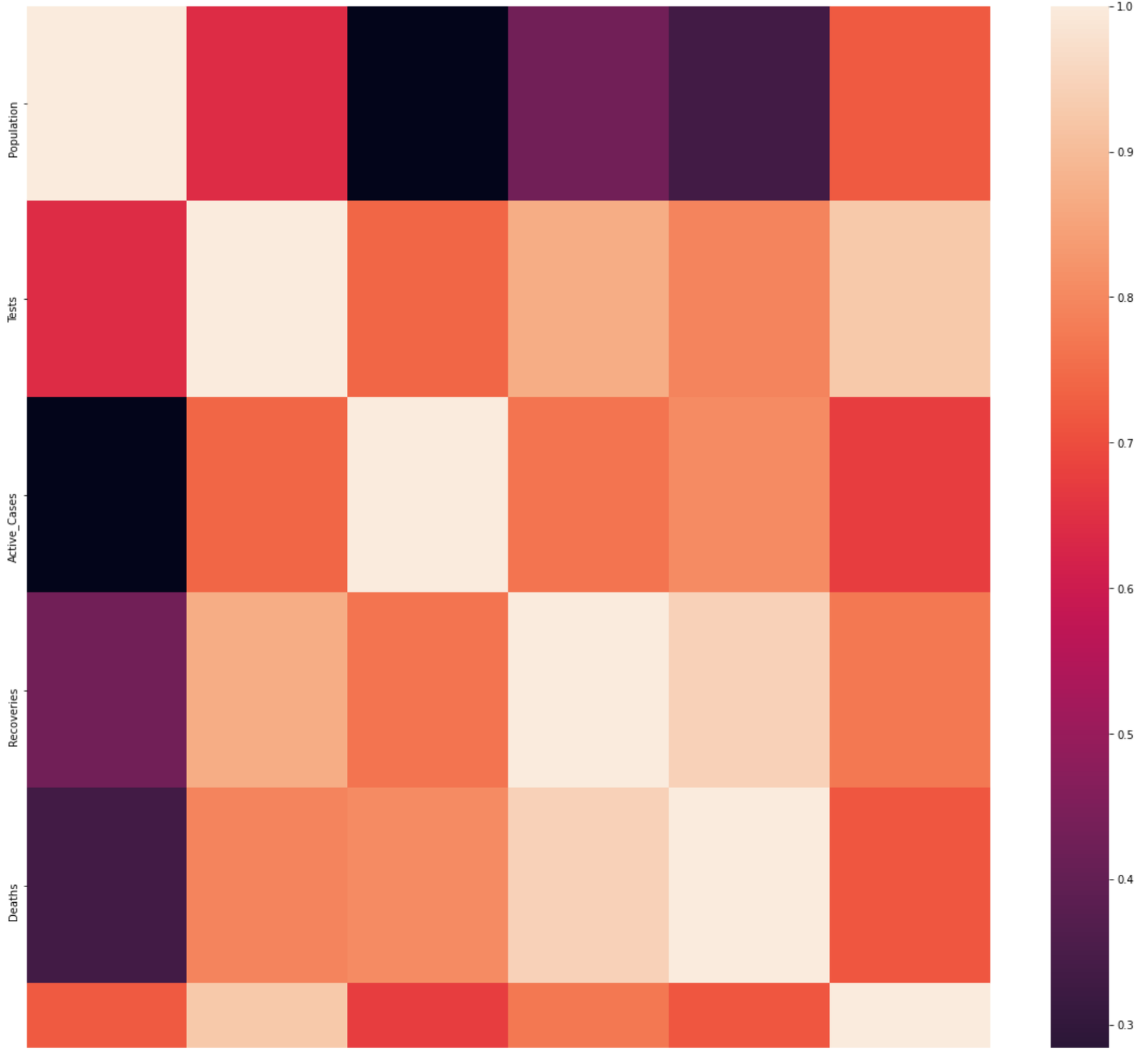
Drop independent variables that are not linear with dependent variable.

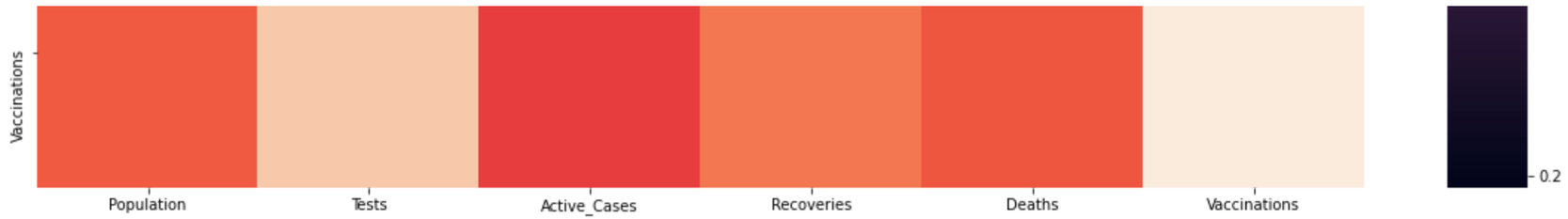
In [399]:
```
X=X.drop('Country', axis=1)
X=X.drop('Tests/Population', axis=1)
X=X.drop('Cases/Tests', axis=1)
X=X.drop('Recoveries/Cases', axis=1)
X=X.drop('Deaths/Cases', axis=1)
X=X.drop('Vaccinations/Population', axis=1)
```

Check multicolinearity assumption.

In [400]:
```
x_corr=X.corr(method='pearson')
plt.figure(figsize=[20, 20])
sns.heatmap(x_corr)
```

Out[400]: <AxesSubplot:>


Population
Tests
Active_Cases
Recoveries
Deaths
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3



In [401]:
```
correlation=[]
for columnName1, columnData1 in X.iteritems():
    for columnName2, columnData2 in X.iteritems():
        if abs(columnData1.corr(columnData2)) > .7:
            correlation.append((columnName1, columnName2, abs(columnData1.corr(columnData2))))
correlation
```

Out[401]:
```
[('Population', 'Population', 1.0),
 ('Population', 'Vaccinations', 0.7215571327372571),
 ('Tests', 'Tests', 1.0),
 ('Tests', 'Active_Cases', 0.7411923216342989),
 ('Tests', 'Recoveries', 0.8705100738823964),
 ('Tests', 'Deaths', 0.7940466335532776),
 ('Tests', 'Vaccinations', 0.9249638935973086),
 ('Active_Cases', 'Tests', 0.7411923216342989),
 ('Active_Cases', 'Active_Cases', 1.0),
 ('Active_Cases', 'Recoveries', 0.7654552243546718),
 ('Active_Cases', 'Deaths', 0.8063702879454626),
 ('Recoveries', 'Tests', 0.8705100738823964),
 ('Recoveries', 'Active_Cases', 0.7654552243546718),
 ('Recoveries', 'Recoveries', 1.0),
 ('Recoveries', 'Deaths', 0.9464553163215914),
 ('Recoveries', 'Vaccinations', 0.7711057055311876),
 ('Deaths', 'Tests', 0.7940466335532776),
 ('Deaths', 'Active_Cases', 0.8063702879454625),
 ('Deaths', 'Recoveries', 0.9464553163215914),
 ('Deaths', 'Deaths', 1.0),
 ('Deaths', 'Vaccinations', 0.7159606713334994),
 ('Vaccinations', 'Population', 0.7215571327372571),
 ('Vaccinations', 'Tests', 0.9249638935973086),
 ('Vaccinations', 'Recoveries', 0.7711057055311875),
 ('Vaccinations', 'Deaths', 0.7159606713334995),
 ('Vaccinations', 'Vaccinations', 1.0)]
```

All independent variables are multicolinear with cases independent variable. Drop all independent variables other than cases.

In [402]:
```
X=X.drop('Population', axis=1)
X=X.drop('Tests', axis=1)
X=X.drop('Recoveries', axis=1)
X=X.drop('Deaths', axis=1)
X=X.drop('Active_Cases', axis=1)
```

Make a train test split.

In [403]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

OLS Model

Linear regression has a y intercept, α, plus the coeficients proportionalized with their independent variables, x, plus an error term, ε.

$$\hat{y} = \alpha + \beta x + \varepsilon$$

Ordinary least squared error, OLS, is used to find the parameters α and β for which the error term is minimized.

$$OLS(\hat{\alpha}, \hat{\beta}) = \sum (y - \alpha - \beta x)^2 = \frac{\sum (y - \bar{y})(x - \bar{x})}{\sum (x - \bar{x})^2}$$

The residual sum of squares, the explained sum of squares, and the total sum of squares are the following:

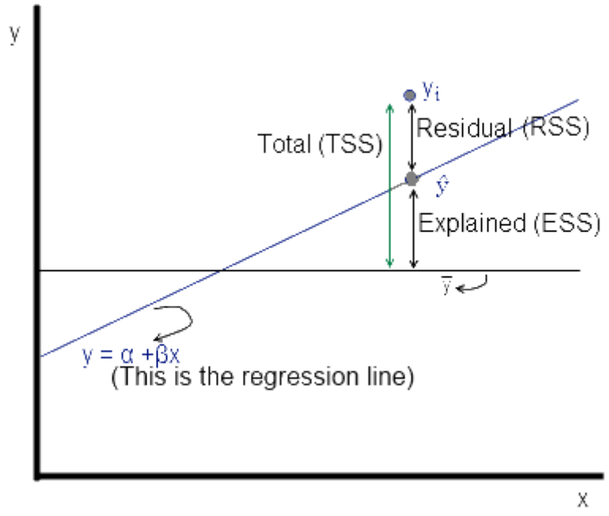
$$RSS = \sum (y - \hat{y})^2$$

$$ESS = \sum (\hat{y} - \bar{y})^2$$

$$TSS = RSS + ESS = \sum (y - \bar{y})^2$$

In [423]:
```
Image(filename='regression.png')
```

Out[423]:



The metrics used to evaluate an OLS model are R squared, which measures how much model explains the variance of the dependent variable, and mean squared error, which measures the average error of the model. n is the number of samples.

$$R^2 = \frac{ESS}{TSS}$$

$$MSE = \frac{RSS}{n}$$

In [404]:
```
predictors_int = sm.add_constant(X_train)
ols = sm.OLS(y_train,predictors_int.astype(float)).fit()
ols.summary()
```

Out[404]:

OLS Regression Results

Dep. Variable:	Cases	**R-squared:**	0.215
Model:	OLS	**Adj. R-squared:**	0.210
Method:	Least Squares	**F-statistic:**	47.62
Date:	Thu, 15 Apr 2021	**Prob (F-statistic):**	9.25e-11
Time:	16:16:36	**Log-Likelihood:**	301.29
No. Observations:	176	**AIC:**	-598.6
Df Residuals:	174	**BIC:**	-592.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0114	0.003	3.356	0.001	0.005	0.018
Vaccinations	0.3239	0.047	6.901	0.000	0.231	0.417

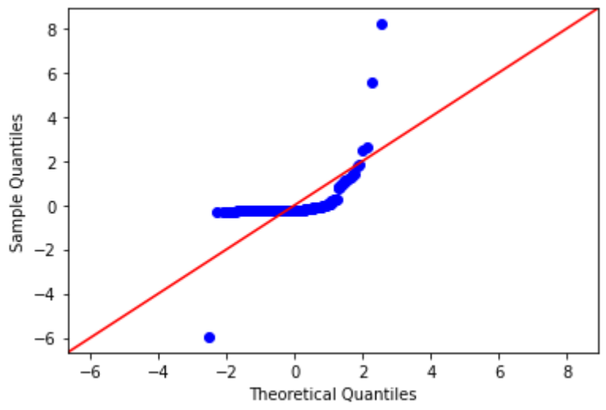
Omnibus:	173.710	**Durbin-Watson:**	2.072
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9739.553
Skew:	3.243	**Prob(JB):**	0.00
Kurtosis:	38.862	**Cond. No.**	14.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

The model only explains 21.5% of the the dependent variable variance.

Check residual normality assumtion.

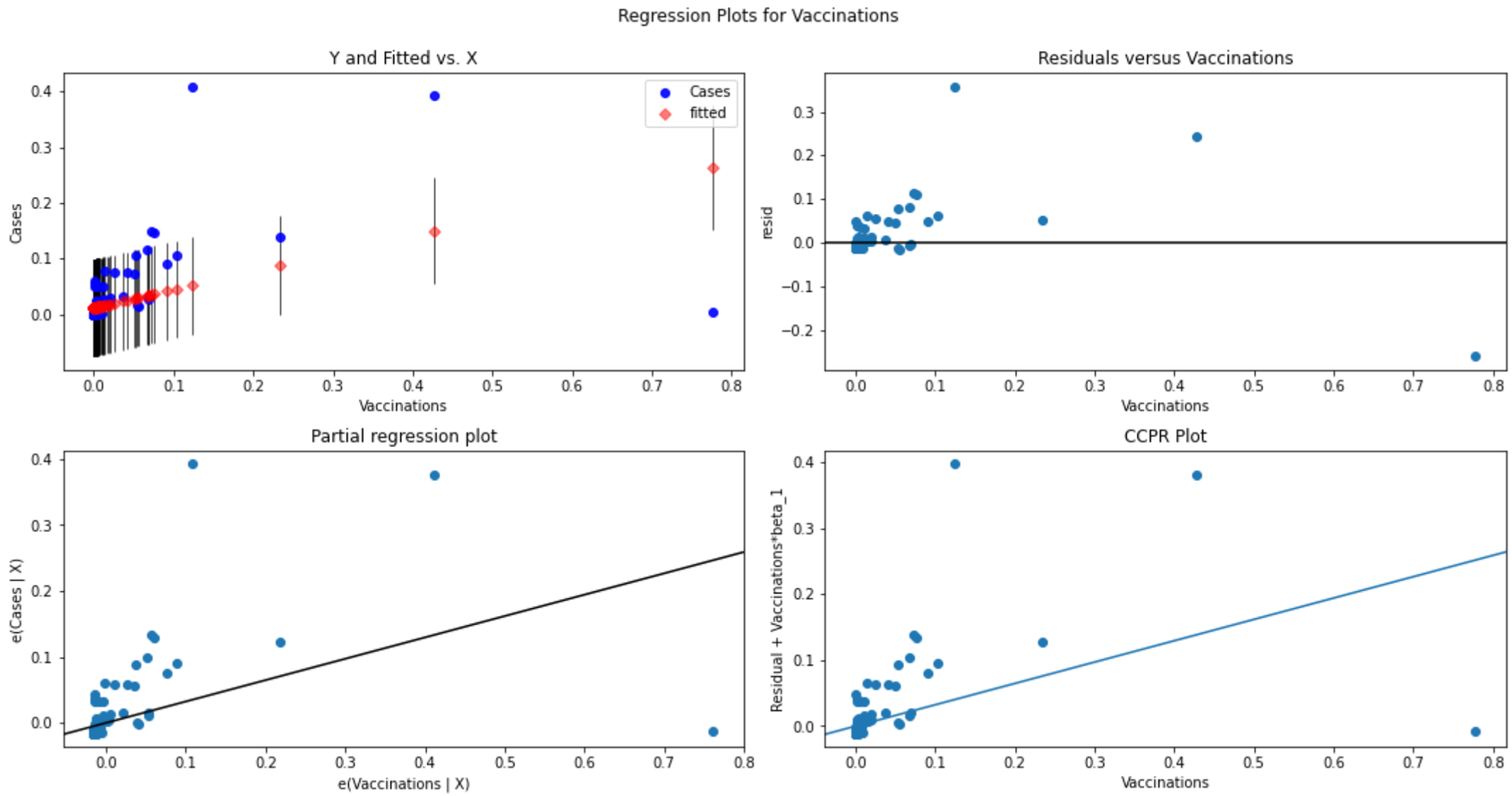
In [405]:
```
residuals = ols.resid
fig = sm.graphics.qqplot(residuals, line='45', fit=True)
fig.show()
```



Residuals don't completly fit normal distribution red line, but are normaly distributed enough with a few outliers.

Check homoskidastisity assumption.

In [406]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(ols, 'Vaccinations', fig=fig)
plt.show()
```



The Goldfeld Quandt test is used to test for homoskedasticity. The test is caculated by taking the mean square residual, RSS, for two subsets of data, and then contructing an F-test to determine wheter to fail to reject or reject the null hypothesis of homoskedasticity. n is the sample size.

$$RSS_1 = \sum (y_1 - \hat{y}_1)^2$$

$$RSS_2 = \sum (y_2 - \hat{y}_2)^2$$

$$F = \frac{(RSS_1/n)}{(RSS_2/n)}$$

$$H0 : homoskedasticity$$

$$H1 : heteroskedasticity$$

```
In [407]: name = ['F statistic', 'p-value']
          test = sms.het_goldfeldquandt(ols.resid, ols.model.exog)
          list(zip(name, test))
```

```
Out[407]: [('F statistic', 0.0933723809981886), ('p-value', 0.9999999999999999)]
```

The Goldfeld Quandt test confirms that residuals are dispersed the same by failing to reject the null hypothesis of homoskedasticity.

Polynomial Regression with Lasso Regularization Model

Polynomial regression uses a higher degree to increase model complxity to explain independent variable variance.

$$\hat{y} = \alpha + \beta_1 x + \ldots + \beta_i x^k + \varepsilon$$

Lasso regression is a regularization technique that uses a penalty term, λ, to shrink the coeficients when computing OLS.

$$OLS(\hat{\alpha}, \hat{\beta}) = \sum (y - \alpha - \beta x)^2 + \lambda \sum |\beta|$$

```
In [606]: poly_features = PolynomialFeatures(degree=3)
          X_poly_train = poly_features.fit_transform(X_train)
          reg = linear_model.Lasso(alpha=1e-05)
          reg.fit(X_poly_train,y_train)
```

```
Out[606]: Lasso(alpha=1e-05)
```

```
In [607]: y_hat_train = reg.predict(X_poly_train)
```

The statistically significant coeficients are negative. Vaccinations and cases are inversely related.

```
In [608]: p=stats.ttest_ind(X_poly_train, y_train)
          print(f'Feature: {X.columns[0]}, Coeficients: {reg.coef_[1:]}, P values: {p[1][1:]}')
```

```
Feature: Vaccinations, Coeficients: [ 1.36026527 -0.64983114 -1.40271111], P values: [0.87816556 0.03213291 0.00459772]
```

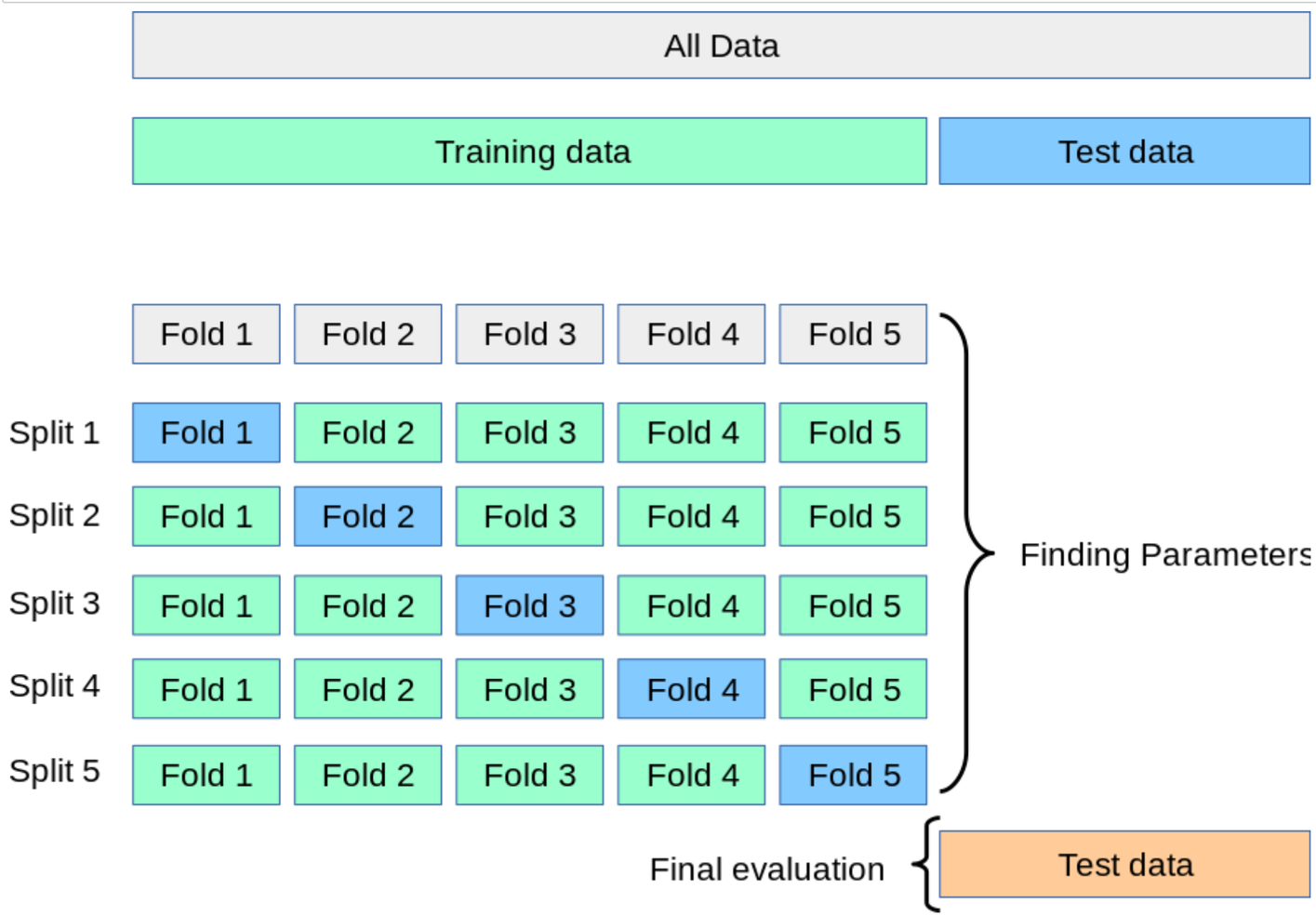
In [609]:
```
r2 = r2_score(y_train,y_hat_train)
mse = mean_squared_error(y_train, y_hat_train)
print('R Squared:',r2)
print('Root Mean Squared Error:', np.sqrt(mse))
```

```
R Squared: 0.7148041182634517
Root Mean Squared Error: 0.026326975968062563
```

Cross validation trains models on k fold subsets of data and evaluates the models on the complementary subset of the data.

In [529]:
```
Image(filename='cross_validation.png')
```

Out[529]:



Cross validate model.

In [616]:
```
cv_results = cross_val_score(reg, X_poly_train, y_train, cv=5, scoring='neg_mean_squared_error')
print('Mean Cross Validation RMSE:',np.mean(np.sqrt(abs(cv_results))))
```

```
Mean Cross Validation RMSE: 0.03800104663543885
```

Test model.

In [611]:
```
X_poly_test = poly_features.fit_transform(X_test)
reg.fit(X_poly_test,y_test)
```

Out[611]:
```
Lasso(alpha=1e-05)
```

In [612]:
```
y_hat_test = reg.predict(X_poly_test)
```

In [614]:
```
r2 = r2_score(y_test,y_hat_test)
mse = mean_squared_error(y_test, y_hat_test)
print('R Squared:',r2)
print('Root Mean Squared Error:', np.sqrt(mse))
```

```
R Squared: 0.9977960692289732
Root Mean Squared Error: 0.0068950582250556286
```

Time Series Models

Time series assumes stationarity, a probability distribution that stays the same over time. The rolling mean, MA, and rolling standard deviation, MSD, are used to visually determine whether the data is stationary.

$$MA = \frac{1}{N}\sum_{i}^{N} p_i$$

$$MSD = \sqrt{\frac{1}{N}\sum_{i}^{N}(p_i - MA)^2}$$

The Dickey-Fuller test is a statistical test used to determine whether the data is stationary. The null hypothesis is that the data is non-stationary. If γ=0, then the data follows a random walk and if γ≠0, then the data is stationary. ε is white noise.

$$\Delta y_i = \alpha + \beta_i + \gamma y_{i-1} + \varepsilon_i$$

Autocorrelation function is used to determine lag correlation for an Auto Regressive model, p, and partial autocrrelation function is used to determine lag correlation for a Moving Average model, q. ACF solves for the correlation of series values with their lagged values. PACF solves for the correlation of the values after removing the effects of previous lags. ACF is covariance of residuals devided by standard deviation multiplied by standard deviation of lag.

$$cov(y_i, y_{i-k}) = \frac{\sum_i^{N-k}(y_{i+1} - \bar{y})(y_{i-k} - \bar{y})}{N-1}$$

$$\sigma y_i = \sqrt{\frac{\sum_i^{N}(y_i - \bar{y})^2}{N-1}}$$

$$ACF(y_i, y_{i-k}) = \frac{cov(y_i, y_{i-k})}{\sigma y_i * \sigma y_{i-k}}$$

$$PACF(y_i, y_{i-k}) = \frac{cov(y_i, y_{i-k}|y_{i+1} \dots y_{i-k-1})}{\sigma y_i|y_{i+1} \dots y_{i-k-1} * \sigma y_{i-k}|y_{i+1} \dots y_{i-k+1}}$$

AIC, akaike information criterion, estimates prediction error. k is the model parameters and $\hat{L}$ is the model maximum value of the likelihood function.

$$AIC = 2k - 2 * ln(\hat{L})$$

An AR model uses lagged values to make forecasts. c is a constant, p is the number of data lags, θ is the coeficients, y is the lags, and Θ is an order p polynomial function of L. MA model uses residuals of previous forecasts to make new forecasts. q is the number of residual lags, ϕ is the coefficients, ε is the residuals, and Φ is an order q polynomial function of L. ARMA model adds the two models.

$$AR : y_t = c + \sum_k^p \phi_k * y_{t-k} + \varepsilon_t = \Phi(L)^p y_t + \varepsilon_t$$

$$MA : y_t = c + \sum_k^q \theta_k * \varepsilon_{t-k} + \varepsilon_t = \Theta(L)^q \varepsilon_t + \varepsilon_t$$

$$ARMA : y_t = \Phi(L)^p y_t + \Theta(L)^q \varepsilon_t + \varepsilon_t$$

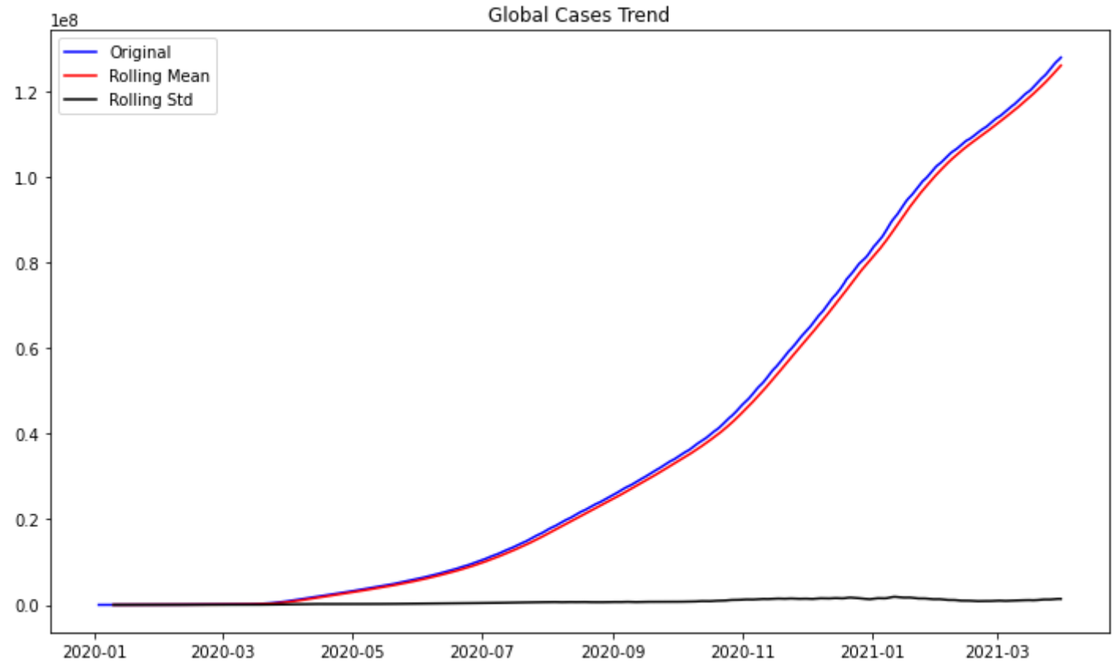
Remove trend by differencing data with lags. An ARIMA model I, integrated, component differences the data to make it stationary. A SARIMA model has a component to represent seasonality, s.

$$(1 - L)^d = y_t - y_{t-1}$$

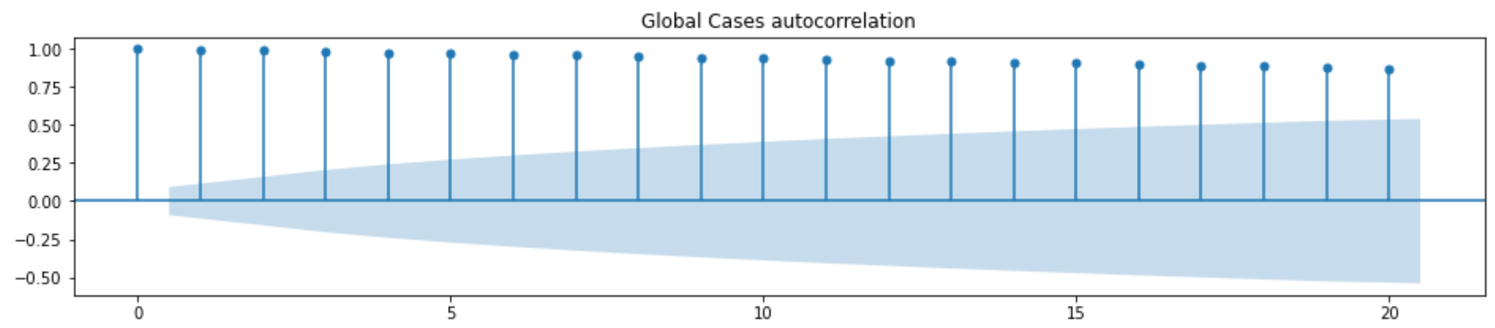
$$ARIMA : (1 - \Phi(L)^p)(1 - L)^d y_t = (1 + \Theta(L)^q)\varepsilon_t$$

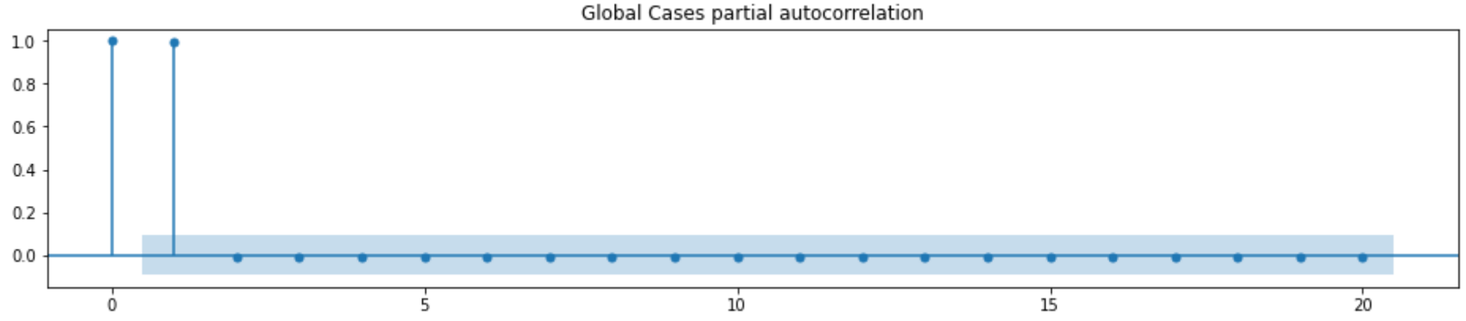
$$SARIMA : (1 - \Phi(L)^p)(1 - \Phi(L)^{ps})(1 - L)^{ds} y_t = (1 + \Theta(L)^q)(1 + \Theta(L)^{qs})\varepsilon_t$$

In [46]:
```
start_new_cases={}
end_new_cases={}
forecast(global_cases_df, 8, start_new_cases, end_new_cases, 30, 'Global Cases')
```



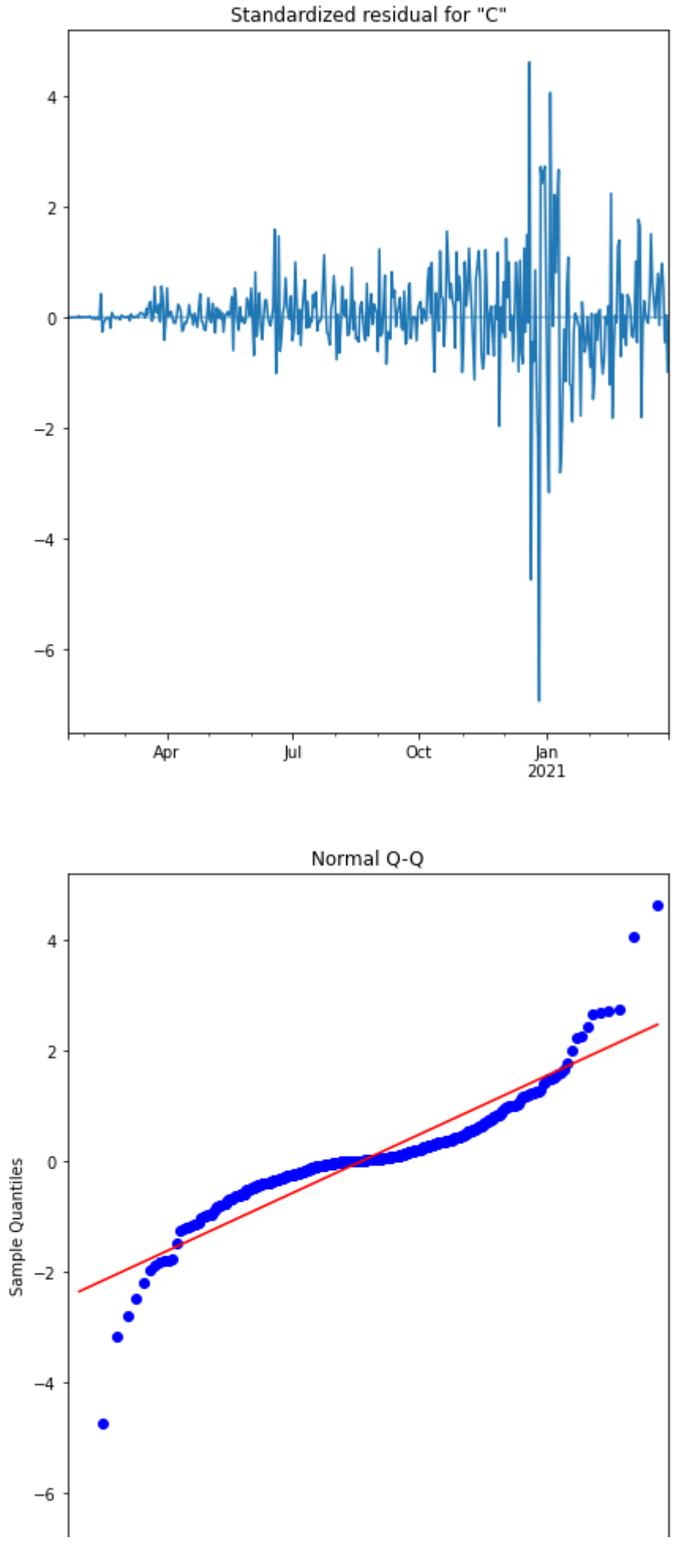

```
Global Cases Dickey-Fuller test results:

Test Statistic                   0.654123
p-value                          0.988876
#Lags Used                      17.000000
Number of Observations Used    436.000000
Critical Value (1%)             -3.445438
Critical Value (5%)             -2.868192
Critical Value (10%)            -2.570313
dtype: float64
```

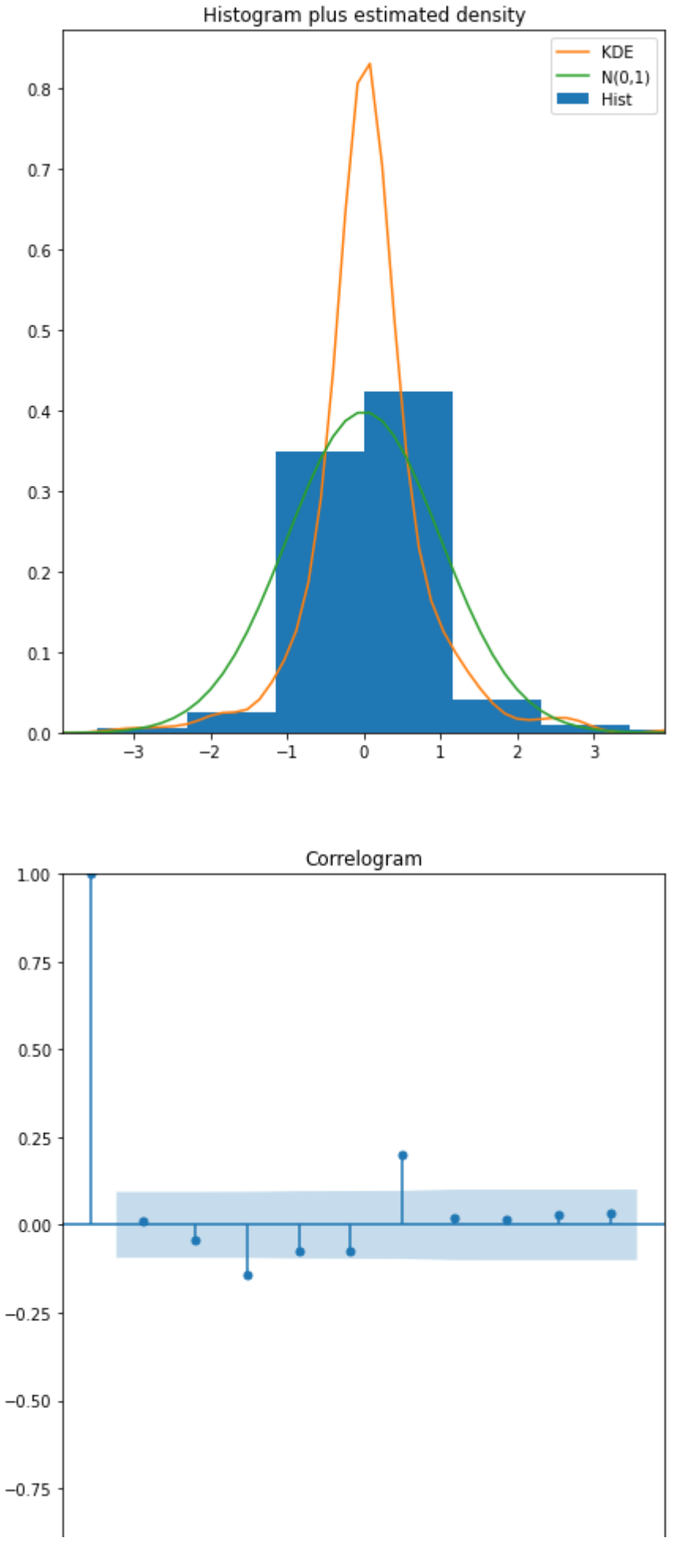



Global Cases partial autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20

```
Global Cases AIC Scores:
ARIMA (0, 0, 0) x (0, 0, 0, 7)12 : AIC Calculated =17446.65142441528
ARIMA (0, 0, 0) x (0, 0, 1, 7)12 : AIC Calculated =17631.638275274137
ARIMA (0, 0, 0) x (0, 1, 0, 7)12 : AIC Calculated =14410.918618753534
ARIMA (0, 0, 0) x (0, 1, 1, 7)12 : AIC Calculated =13882.868765690982
ARIMA (0, 0, 0) x (1, 0, 0, 7)12 : AIC Calculated =13681.224180158386
ARIMA (0, 0, 0) x (1, 0, 1, 7)12 : AIC Calculated =13346.54814411781
ARIMA (0, 0, 0) x (1, 1, 0, 7)12 : AIC Calculated =12183.894937634634
ARIMA (0, 0, 0) x (1, 1, 1, 7)12 : AIC Calculated =12073.270168384384
ARIMA (0, 0, 1) x (0, 0, 0, 7)12 : AIC Calculated =17086.09867987108
ARIMA (0, 0, 1) x (0, 0, 1, 7)12 : AIC Calculated =19227.15759604908
ARIMA (0, 0, 1) x (0, 1, 0, 7)12 : AIC Calculated =14050.654791231847
ARIMA (0, 0, 1) x (0, 1, 1, 7)12 : AIC Calculated =13932.864830200657
ARIMA (0, 0, 1) x (1, 0, 0, 7)12 : AIC Calculated =16775.592098663597
ARIMA (0, 0, 1) x (1, 0, 1, 7)12 : AIC Calculated =16702.533465115594
ARIMA (0, 0, 1) x (1, 1, 0, 7)12 : AIC Calculated =13785.203117612858
ARIMA (0, 0, 1) x (1, 1, 1, 7)12 : AIC Calculated =13724.439246266866
ARIMA (0, 1, 0) x (0, 0, 0, 7)12 : AIC Calculated =12852.669352502904
ARIMA (0, 1, 0) x (0, 0, 1, 7)12 : AIC Calculated =12354.282729150931
ARIMA (0, 1, 0) x (0, 1, 0, 7)12 : AIC Calculated =10800.992422975807
ARIMA (0, 1, 0) x (0, 1, 1, 7)12 : AIC Calculated =10627.962852564777
ARIMA (0, 1, 0) x (1, 0, 0, 7)12 : AIC Calculated =10822.340920318751
ARIMA (0, 1, 0) x (1, 0, 1, 7)12 : AIC Calculated =10790.756561032244
ARIMA (0, 1, 0) x (1, 1, 0, 7)12 : AIC Calculated =10651.80075959113
ARIMA (0, 1, 0) x (1, 1, 1, 7)12 : AIC Calculated =10631.034769745194
ARIMA (0, 1, 1) x (0, 0, 0, 7)12 : AIC Calculated =12496.017558580957
ARIMA (0, 1, 1) x (0, 0, 1, 7)12 : AIC Calculated =12227.649563478912
ARIMA (0, 1, 1) x (0, 1, 0, 7)12 : AIC Calculated =10603.185311075278
ARIMA (0, 1, 1) x (0, 1, 1, 7)12 : AIC Calculated =10443.33394874522
ARIMA (0, 1, 1) x (1, 0, 0, 7)12 : AIC Calculated =12238.233454322504
ARIMA (0, 1, 1) x (1, 0, 1, 7)12 : AIC Calculated =12185.395304082129
ARIMA (0, 1, 1) x (1, 1, 0, 7)12 : AIC Calculated =10489.566637728665
ARIMA (0, 1, 1) x (1, 1, 1, 7)12 : AIC Calculated =10445.104711970102
ARIMA (1, 0, 0) x (0, 0, 0, 7)12 : AIC Calculated =12123.215923471731
ARIMA (1, 0, 0) x (0, 0, 1, 7)12 : AIC Calculated =12414.065595671269
ARIMA (1, 0, 0) x (0, 1, 0, 7)12 : AIC Calculated =10816.808333981253
ARIMA (1, 0, 0) x (0, 1, 1, 7)12 : AIC Calculated =10786.676658251443
ARIMA (1, 0, 0) x (1, 0, 0, 7)12 : AIC Calculated =11527.986604803104
ARIMA (1, 0, 0) x (1, 0, 1, 7)12 : AIC Calculated =11528.908137081831
ARIMA (1, 0, 0) x (1, 1, 0, 7)12 : AIC Calculated =10646.924465736049
ARIMA (1, 0, 0) x (1, 1, 1, 7)12 : AIC Calculated =10658.173468218552
ARIMA (1, 0, 1) x (0, 0, 0, 7)12 : AIC Calculated =11760.58942683645
ARIMA (1, 0, 1) x (0, 0, 1, 7)12 : AIC Calculated =12095.460315410048
ARIMA (1, 0, 1) x (0, 1, 0, 7)12 : AIC Calculated =10620.711897824713
ARIMA (1, 0, 1) x (0, 1, 1, 7)12 : AIC Calculated =10663.723628760334
ARIMA (1, 0, 1) x (1, 0, 0, 7)12 : AIC Calculated =11469.562044349124
ARIMA (1, 0, 1) x (1, 0, 1, 7)12 : AIC Calculated =11445.929652278752
ARIMA (1, 0, 1) x (1, 1, 0, 7)12 : AIC Calculated =10483.931635570916
ARIMA (1, 0, 1) x (1, 1, 1, 7)12 : AIC Calculated =10459.31136679001
ARIMA (1, 1, 0) x (0, 0, 0, 7)12 : AIC Calculated =10930.531356796262
ARIMA (1, 1, 0) x (0, 0, 1, 7)12 : AIC Calculated =10601.957764509247
ARIMA (1, 1, 0) x (0, 1, 0, 7)12 : AIC Calculated =10453.21659309935
ARIMA (1, 1, 0) x (0, 1, 1, 7)12 : AIC Calculated =10177.15721444577
ARIMA (1, 1, 0) x (1, 0, 0, 7)12 : AIC Calculated =10537.399504448624
ARIMA (1, 1, 0) x (1, 0, 1, 7)12 : AIC Calculated =10481.810449301982
ARIMA (1, 1, 0) x (1, 1, 0, 7)12 : AIC Calculated =10261.52032244702
ARIMA (1, 1, 0) x (1, 1, 1, 7)12 : AIC Calculated =10177.325789282198
```

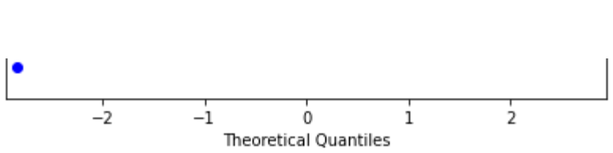
```
ARIMA (1, 1, 1) x (0, 0, 0, 7)12 : AIC Calculated =10868.164011926763
ARIMA (1, 1, 1) x (0, 0, 1, 7)12 : AIC Calculated =10566.804096910944
ARIMA (1, 1, 1) x (0, 1, 0, 7)12 : AIC Calculated =10419.820289635687
ARIMA (1, 1, 1) x (0, 1, 1, 7)12 : AIC Calculated =10144.570016513087
ARIMA (1, 1, 1) x (1, 0, 0, 7)12 : AIC Calculated =10507.904983660163
ARIMA (1, 1, 1) x (1, 0, 1, 7)12 : AIC Calculated =10430.74327723813
ARIMA (1, 1, 1) x (1, 1, 0, 7)12 : AIC Calculated =10249.641372417764
ARIMA (1, 1, 1) x (1, 1, 1, 7)12 : AIC Calculated =10143.465401504374
Global Cases ARIMA:                                SARIMAX Results
==========================================================================================
Dep. Variable:                 Cumulative_cases   No. Observations:                  454
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 7)   Log Likelihood               -5066.733
Date:                           Wed, 31 Mar 2021   AIC                           10143.465
Time:                                   21:12:27   BIC                           10163.865
Sample:                             01-03-2020   HQIC                          10151.515
                                  - 03-31-2021
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9662      0.014     69.718      0.000       0.939       0.993
ma.L1         -0.2192      0.028     -7.909      0.000      -0.274      -0.165
ar.S.L7        0.1211      0.038      3.218      0.001       0.047       0.195
ma.S.L7       -0.8014      0.033    -24.160      0.000      -0.866      -0.736
sigma2      9.041e+08   1.08e-11   8.35e+19      0.000    9.04e+08    9.04e+08
===================================================================================
Ljung-Box (L1) (Q):                   0.06   Jarque-Bera (JB):              4685.62
Prob(Q):                              0.80   Prob(JB):                         0.00
Heteroskedasticity (H):              43.30   Skew:                            -1.02
Prob(H) (two-sided):                  0.00   Kurtosis:                        18.91
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.67e+35. Standard errors may be unstable.
```

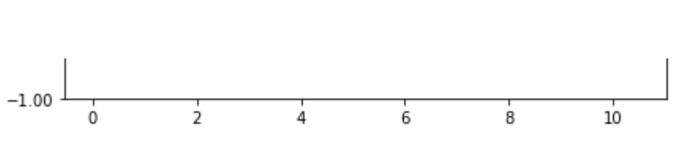

Standardized residual for "C"
4
2
0
−2
−4
−6
Apr
Jul
Oct
Jan
2021

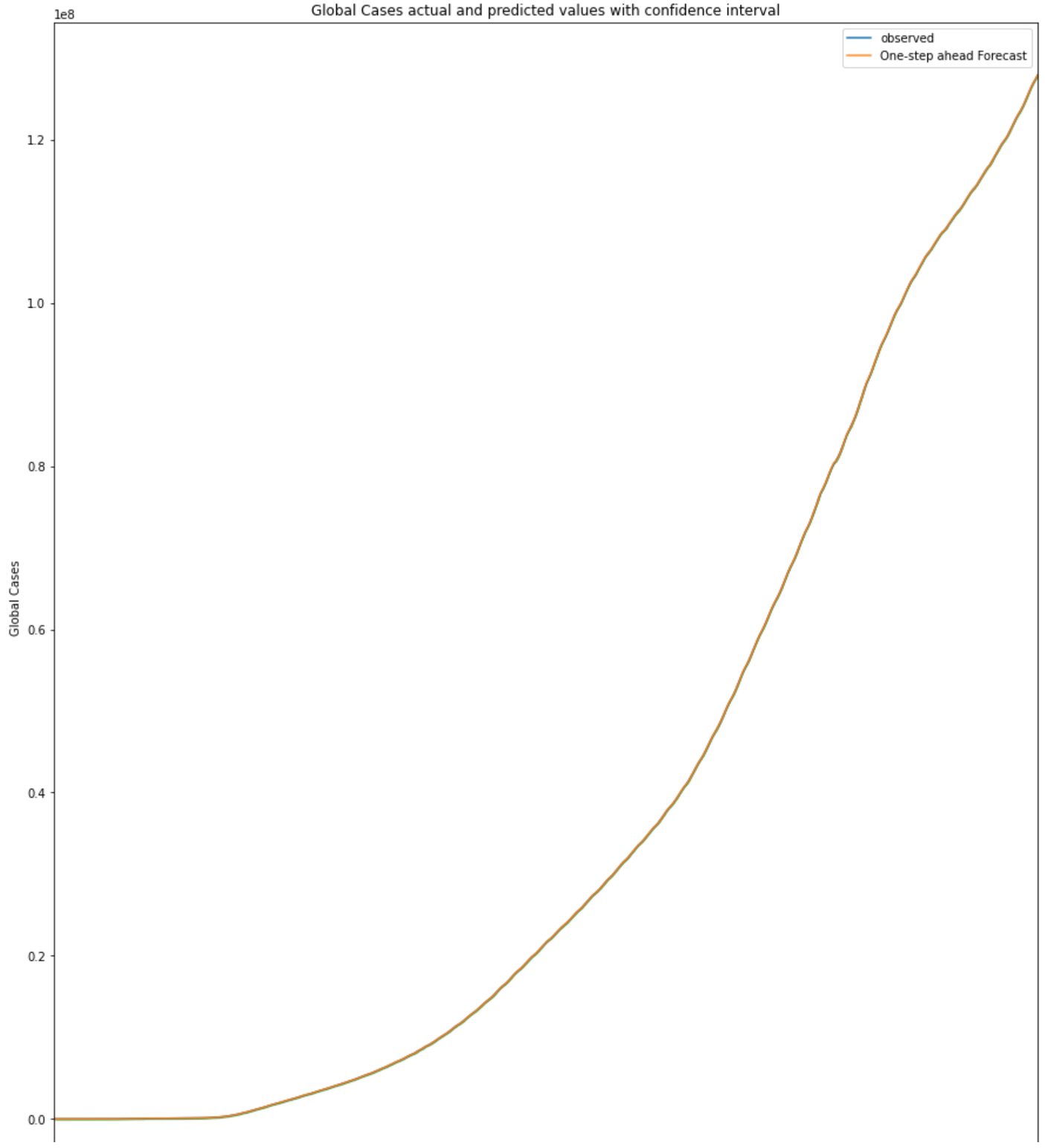

Histogram plus estimated density
KDE
N(0,1)
Hist
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−3
−2
−1
0
1
2
3

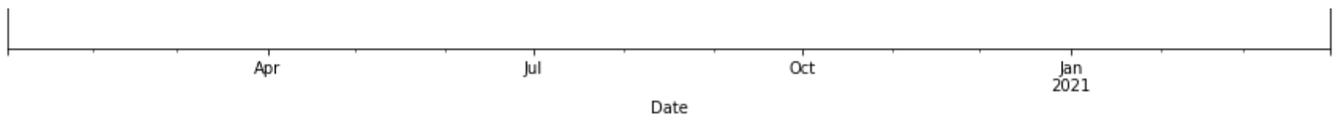
Normal Q-Q
Sample Quantiles
4
2
0
−2
−4
−6

Correlogram
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

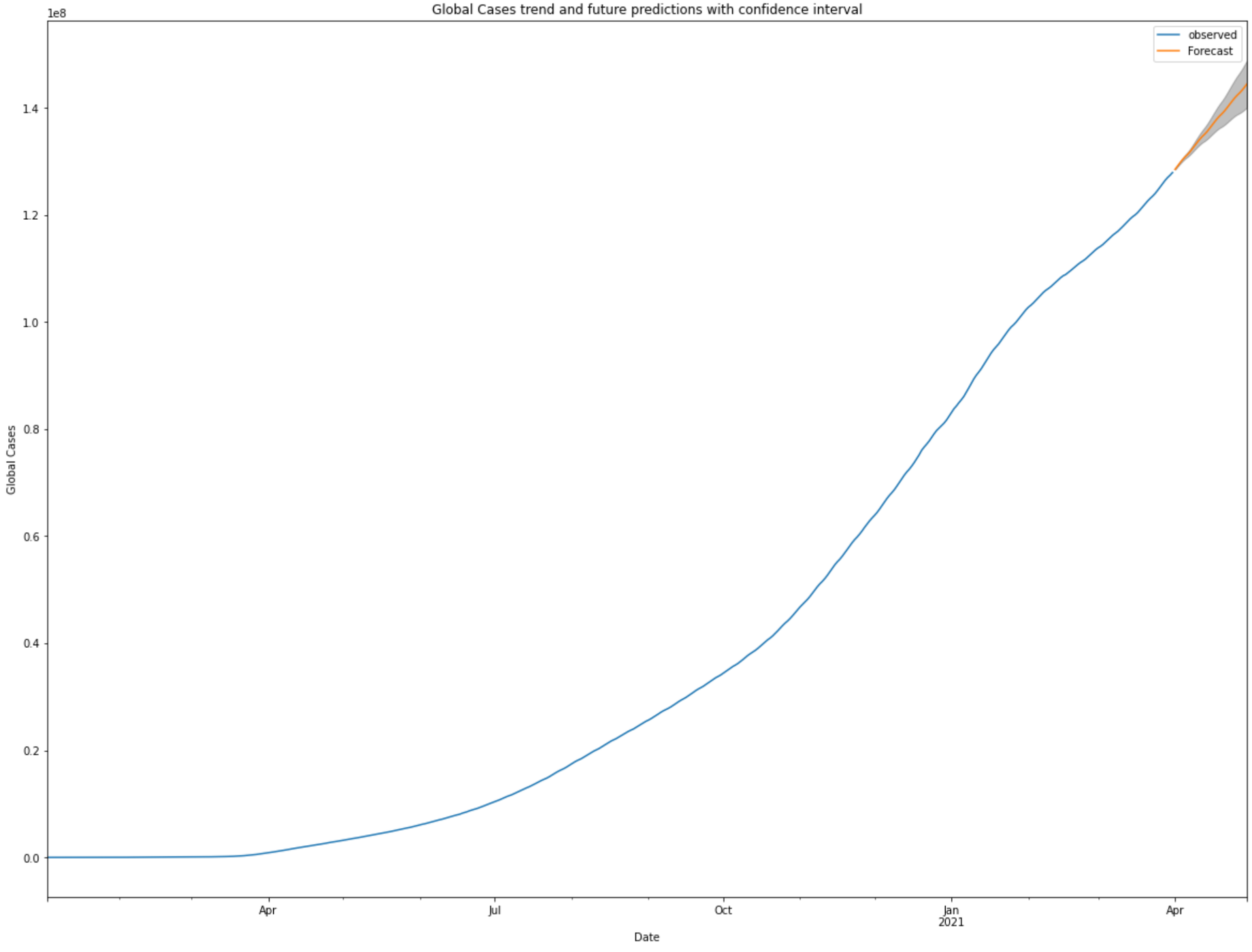

−2
−1
0
1
2
Theoretical Quantiles


−1.00
0
2
4
6
8
10


Global Cases actual and predicted values with confidence interval
1e8
observed
One-step ahead Forecast
Global Cases
1.2
1.0
0.8
0.6
0.4
0.2
0.0




```
Global Cases Root Mean Squared Error:25176.24
Global Cases Forecast:
2021-04-01    1.285216e+08
2021-04-02    1.291329e+08
2021-04-03    1.297402e+08
2021-04-04    1.303214e+08
2021-04-05    1.308377e+08
2021-04-06    1.312787e+08
2021-04-07    1.317902e+08
2021-04-08    1.323770e+08
2021-04-09    1.329853e+08
2021-04-10    1.335888e+08
2021-04-11    1.341678e+08
2021-04-12    1.346815e+08
2021-04-13    1.351222e+08
2021-04-14    1.356331e+08
2021-04-15    1.362191e+08
2021-04-16    1.368263e+08
2021-04-17    1.374287e+08
2021-04-18    1.380067e+08
2021-04-19    1.385195e+08
2021-04-20    1.389596e+08
2021-04-21    1.394698e+08
2021-04-22    1.400551e+08
2021-04-23    1.406616e+08
2021-04-24    1.412634e+08
2021-04-25    1.418408e+08
2021-04-26    1.423529e+08
2021-04-27    1.427924e+08
2021-04-28    1.433021e+08
2021-04-29    1.438869e+08
2021-04-30    1.444929e+08
Freq: D, Name: predicted_mean, dtype: float64
```


Global Cases trend and future predictions with confidence interval
1e8
observed
Forecast
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Global Cases
Apr
Jul
Oct
Jan
2021
Apr
Date

In [181]:
```
total_cases=round(sum(start_new_cases.values()))
print('Total Global Cases:', total_cases)
```

```
Total Global Cases: 127405632
```

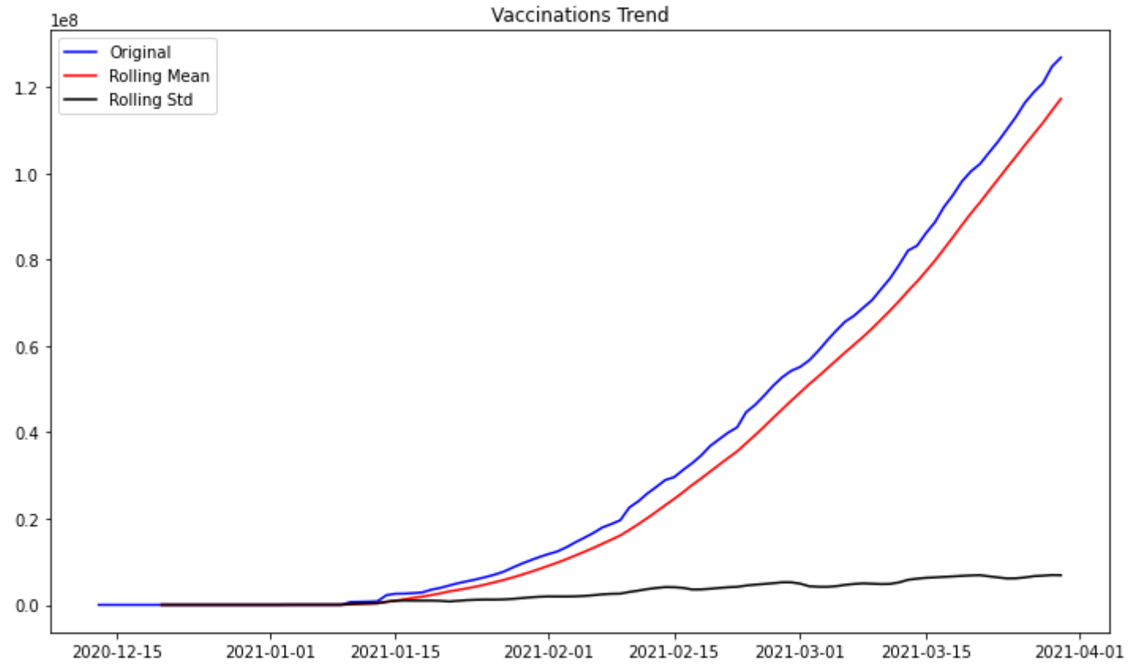
In [182]:
```
forecasted_cases=round(sum(end_new_cases.values()))
print('Forecasted Global Cases:', forecasted_cases)
```

```
Forecasted Global Cases: 143654082
```

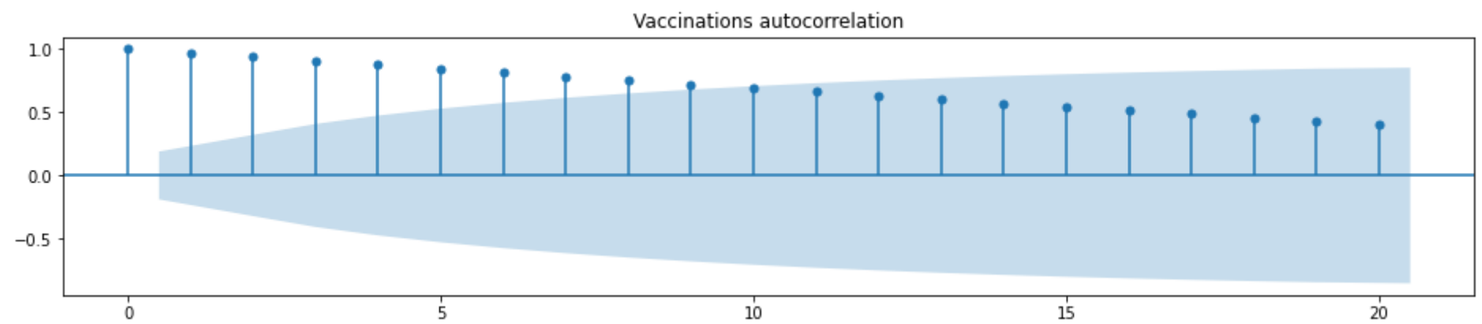
In [183]:
```
pecentage_change(total_cases,forecasted_cases)
```

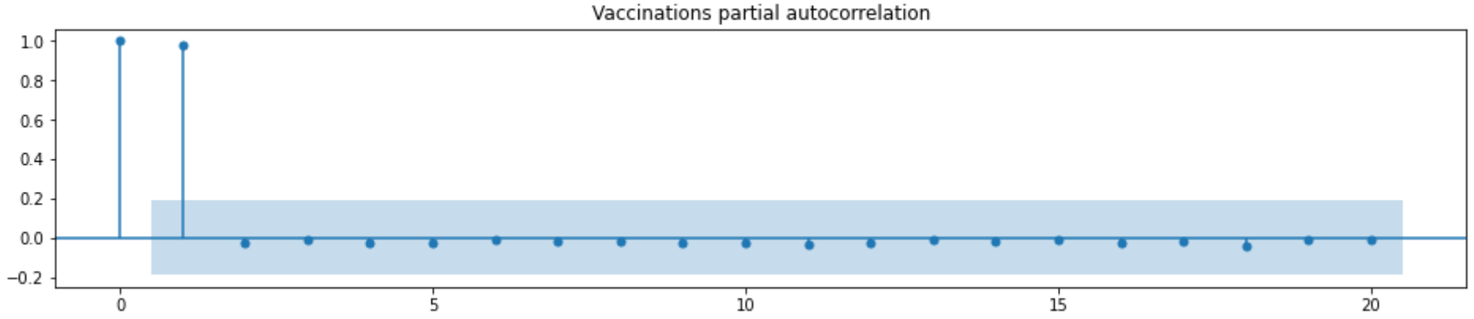
Out[183]:
```
'Pecentage Change: 12.75332161140255%'
```

In [47]:
```
start_new_vaccinations={}
end_new_vaccinations={}
forecast(global_vaccinations_df, 8, start_new_vaccinations, end_new_vaccinations, 30, 'Vaccinations')
```



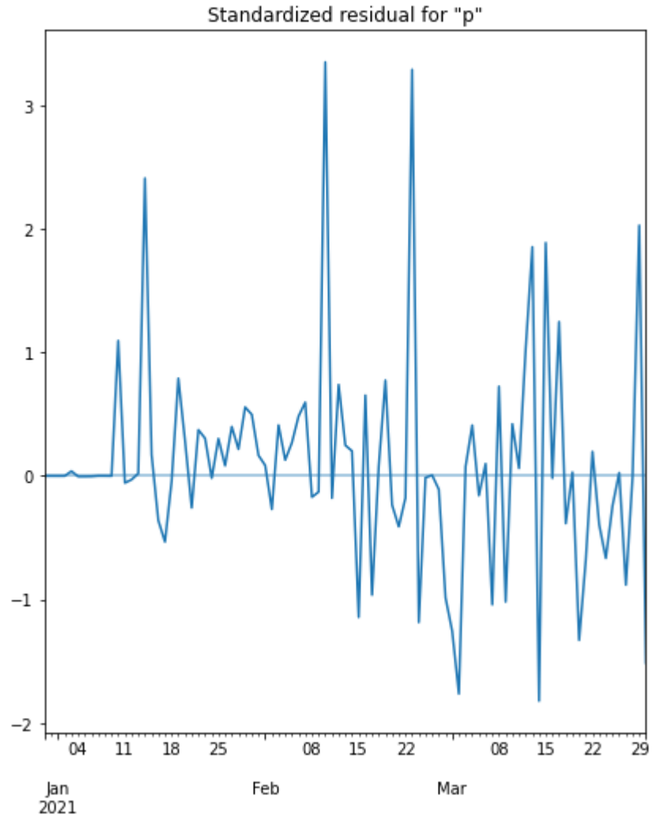

```
Vaccinations Dickey-Fuller test results:

Test Statistic                  -0.586827
p-value                          0.873931
#Lags Used                      13.000000
Number of Observations Used     94.000000
Critical Value (1%)             -3.501912
Critical Value (5%)             -2.892815
Critical Value (10%)            -2.583454
dtype: float64
```

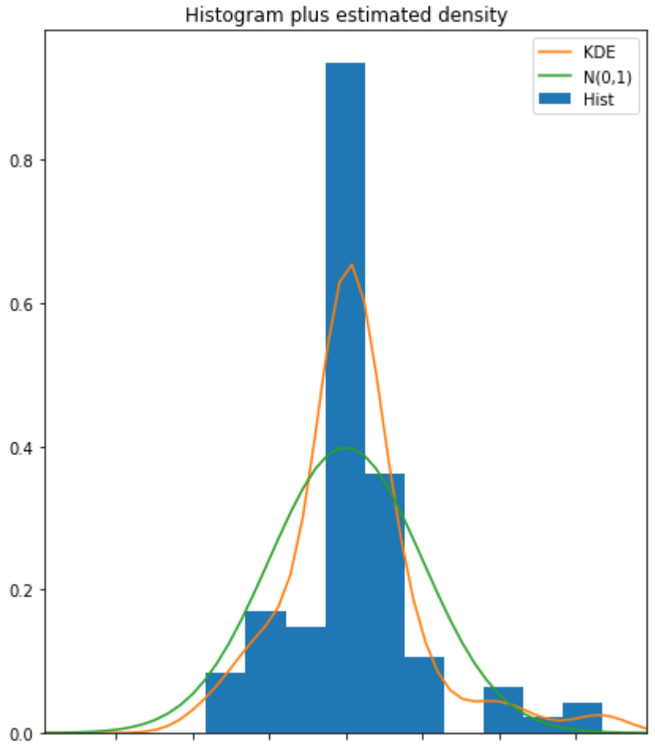


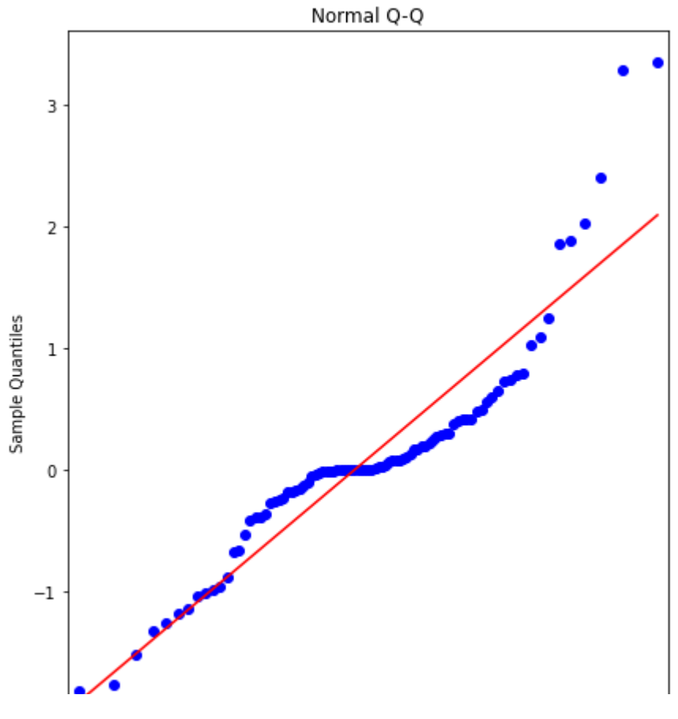

Vaccinations partial autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20

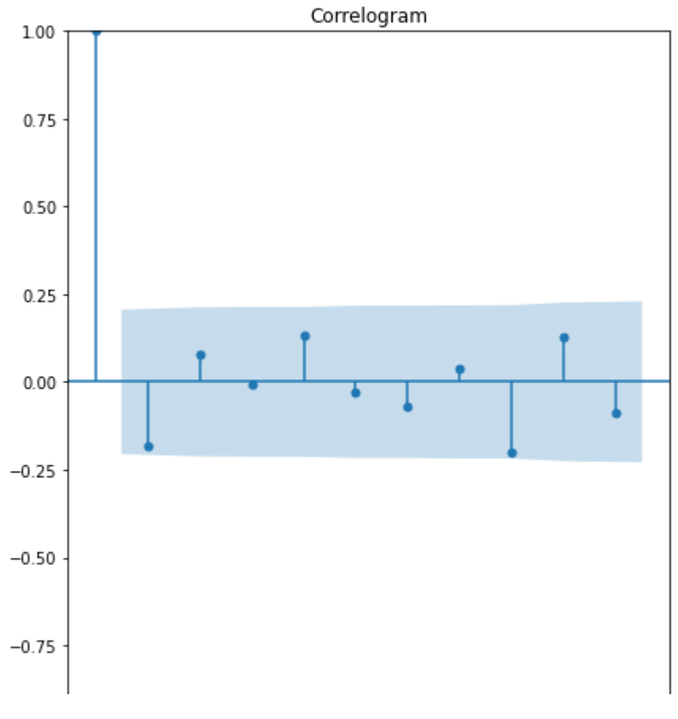
```
Vaccinations AIC Scores:
ARIMA (0, 0, 0) x (0, 0, 0, 7)12 : AIC Calculated =4105.1157827491315
ARIMA (0, 0, 0) x (0, 0, 1, 7)12 : AIC Calculated =3954.3448549453487
ARIMA (0, 0, 0) x (0, 1, 0, 7)12 : AIC Calculated =3523.473011617948
ARIMA (0, 0, 0) x (0, 1, 1, 7)12 : AIC Calculated =3419.5286218208353
ARIMA (0, 0, 0) x (1, 0, 0, 7)12 : AIC Calculated =3322.7453960040934
ARIMA (0, 0, 0) x (1, 0, 1, 7)12 : AIC Calculated =3256.2432483931007
ARIMA (0, 0, 0) x (1, 1, 0, 7)12 : AIC Calculated =2979.644211716834
ARIMA (0, 0, 0) x (1, 1, 1, 7)12 : AIC Calculated =2950.455600634846
ARIMA (0, 0, 1) x (0, 0, 0, 7)12 : AIC Calculated =3991.1885569417223
ARIMA (0, 0, 1) x (0, 0, 1, 7)12 : AIC Calculated =4275.722577713414
ARIMA (0, 0, 1) x (0, 1, 0, 7)12 : AIC Calculated =3419.066380701285
ARIMA (0, 0, 1) x (0, 1, 1, 7)12 : AIC Calculated =3439.462883682845
ARIMA (0, 0, 1) x (1, 0, 0, 7)12 : AIC Calculated =3780.2957026956574
ARIMA (0, 0, 1) x (1, 0, 1, 7)12 : AIC Calculated =3707.5149473118067
ARIMA (0, 0, 1) x (1, 1, 0, 7)12 : AIC Calculated =3225.4067930544475
ARIMA (0, 0, 1) x (1, 1, 1, 7)12 : AIC Calculated =3158.875429815208
ARIMA (0, 1, 0) x (0, 0, 0, 7)12 : AIC Calculated =3333.3406663817063
ARIMA (0, 1, 0) x (0, 0, 1, 7)12 : AIC Calculated =3072.13450025509
ARIMA (0, 1, 0) x (0, 1, 0, 7)12 : AIC Calculated =2906.306479485046
ARIMA (0, 1, 0) x (0, 1, 1, 7)12 : AIC Calculated =2706.897764931974
ARIMA (0, 1, 0) x (1, 0, 0, 7)12 : AIC Calculated =2933.9857624147808
ARIMA (0, 1, 0) x (1, 0, 1, 7)12 : AIC Calculated =2895.5060956351663
ARIMA (0, 1, 0) x (1, 1, 0, 7)12 : AIC Calculated =2734.56197841205
ARIMA (0, 1, 0) x (1, 1, 1, 7)12 : AIC Calculated =2708.4477315741156
ARIMA (0, 1, 1) x (0, 0, 0, 7)12 : AIC Calculated =3250.071190709681
ARIMA (0, 1, 1) x (0, 0, 1, 7)12 : AIC Calculated =3015.8098581131826
ARIMA (0, 1, 1) x (0, 1, 0, 7)12 : AIC Calculated =2878.179122348249
ARIMA (0, 1, 1) x (0, 1, 1, 7)12 : AIC Calculated =2678.1935218344565
ARIMA (0, 1, 1) x (1, 0, 0, 7)12 : AIC Calculated =3061.3688354899346
ARIMA (0, 1, 1) x (1, 0, 1, 7)12 : AIC Calculated =3001.150150844575
ARIMA (0, 1, 1) x (1, 1, 0, 7)12 : AIC Calculated =2734.8372644361816
ARIMA (0, 1, 1) x (1, 1, 1, 7)12 : AIC Calculated =2680.0777207488295
ARIMA (1, 0, 0) x (0, 0, 0, 7)12 : AIC Calculated =3164.506699178182
ARIMA (1, 0, 0) x (0, 0, 1, 7)12 : AIC Calculated =3116.222313855577
ARIMA (1, 0, 0) x (0, 1, 0, 7)12 : AIC Calculated =2927.2312932404798
ARIMA (1, 0, 0) x (0, 1, 1, 7)12 : AIC Calculated =2891.382160039508
ARIMA (1, 0, 0) x (1, 0, 0, 7)12 : AIC Calculated =2925.521400069446
ARIMA (1, 0, 0) x (1, 0, 1, 7)12 : AIC Calculated =2924.618596393868
ARIMA (1, 0, 0) x (1, 1, 0, 7)12 : AIC Calculated =2720.979430188925
ARIMA (1, 0, 0) x (1, 1, 1, 7)12 : AIC Calculated =2719.7598816354334
ARIMA (1, 0, 1) x (0, 0, 0, 7)12 : AIC Calculated =3126.244462236315
ARIMA (1, 0, 1) x (0, 0, 1, 7)12 : AIC Calculated =3086.470459773389
ARIMA (1, 0, 1) x (0, 1, 0, 7)12 : AIC Calculated =2900.9032567633894
ARIMA (1, 0, 1) x (0, 1, 1, 7)12 : AIC Calculated =2867.4627409882414
ARIMA (1, 0, 1) x (1, 0, 0, 7)12 : AIC Calculated =2925.6099131539722
ARIMA (1, 0, 1) x (1, 0, 1, 7)12 : AIC Calculated =2896.773747861594
ARIMA (1, 0, 1) x (1, 1, 0, 7)12 : AIC Calculated =2723.1031005798436
ARIMA (1, 0, 1) x (1, 1, 1, 7)12 : AIC Calculated =2693.6593738259976
ARIMA (1, 1, 0) x (0, 0, 0, 7)12 : AIC Calculated =3140.8496217966
ARIMA (1, 1, 0) x (0, 0, 1, 7)12 : AIC Calculated =2926.267218139881
ARIMA (1, 1, 0) x (0, 1, 0, 7)12 : AIC Calculated =2905.983471116158
ARIMA (1, 1, 0) x (0, 1, 1, 7)12 : AIC Calculated =2705.450544937997
ARIMA (1, 1, 0) x (1, 0, 0, 7)12 : AIC Calculated =2910.85709758607
ARIMA (1, 1, 0) x (1, 0, 1, 7)12 : AIC Calculated =2904.3996221967036
ARIMA (1, 1, 0) x (1, 1, 0, 7)12 : AIC Calculated =2705.701619763774
ARIMA (1, 1, 0) x (1, 1, 1, 7)12 : AIC Calculated =2707.431254445614
```

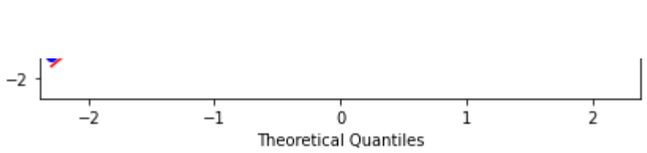
```
ARIMA (1, 1, 1) x (0, 0, 0, 7)12 : AIC Calculated =3060.9372148369175
ARIMA (1, 1, 1) x (0, 0, 1, 7)12 : AIC Calculated =2855.224875957011
ARIMA (1, 1, 1) x (0, 1, 0, 7)12 : AIC Calculated =2872.167831172917
ARIMA (1, 1, 1) x (0, 1, 1, 7)12 : AIC Calculated =2654.959693052119
ARIMA (1, 1, 1) x (1, 0, 0, 7)12 : AIC Calculated =2881.015719679422
ARIMA (1, 1, 1) x (1, 0, 1, 7)12 : AIC Calculated =2852.9882469297195
ARIMA (1, 1, 1) x (1, 1, 0, 7)12 : AIC Calculated =2692.0303251580226
ARIMA (1, 1, 1) x (1, 1, 1, 7)12 : AIC Calculated =2656.195091005778
Vaccinations ARIMA:                                SARIMAX Results
==========================================================================================
Dep. Variable:           people_fully_vaccinated   No. Observations:                  108
Model:             SARIMAX(1, 1, 1)x(0, 1, 1, 7)   Log Likelihood               -1323.480
Date:                           Wed, 31 Mar 2021   AIC                           2654.960
Time:                                   21:12:35   BIC                           2665.003
Sample:                             12-13-2020   HQIC                          2659.012
                                  - 03-30-2021
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.0037      0.014     74.126      0.000       0.977       1.030
ma.L1         -0.8904      0.087    -10.258      0.000      -1.060      -0.720
ma.S.L7       -0.6812      0.103     -6.605      0.000      -0.883      -0.479
sigma2      3.166e+11   2.96e-14   1.07e+25      0.000    3.17e+11    3.17e+11
===================================================================================
Ljung-Box (L1) (Q):                   3.06   Jarque-Bera (JB):                63.58
Prob(Q):                              0.08   Prob(JB):                         0.00
Heteroskedasticity (H):               3.53   Skew:                             1.13
Prob(H) (two-sided):                  0.00   Kurtosis:                         6.41
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.65e+41. Standard errors may be unstable.
```

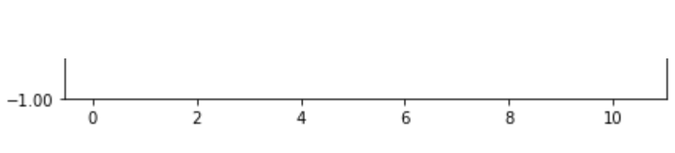

Standardized residual for "p"
3
2
1
0
−1
−2
04
11
18
25
08
15
22
08
15
22
29
Jan
2021
Feb
Mar

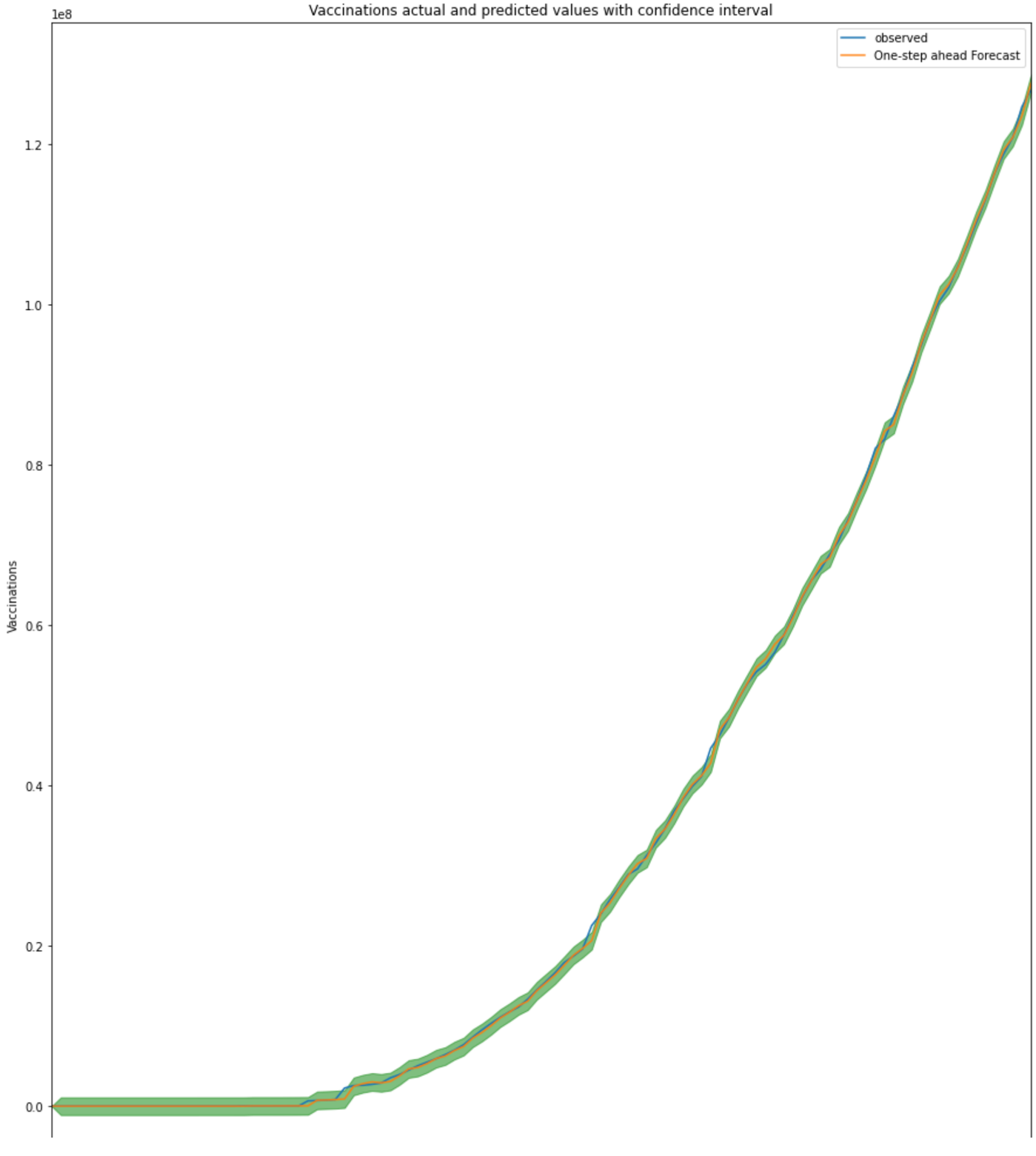

Histogram plus estimated density
KDE
N(0,1)
Hist
0.8
0.6
0.4
0.2
0.0
−3
−2
−1
0
1
2
3

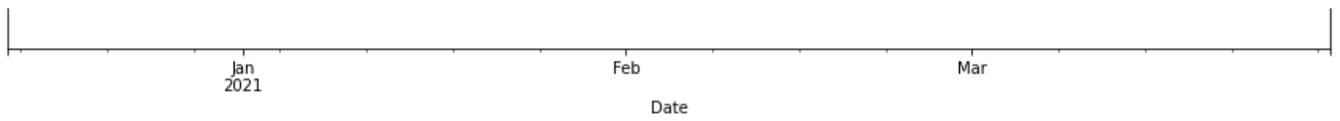

Normal Q-Q
Sample Quantiles
3
2
1
0
−1


Correlogram
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

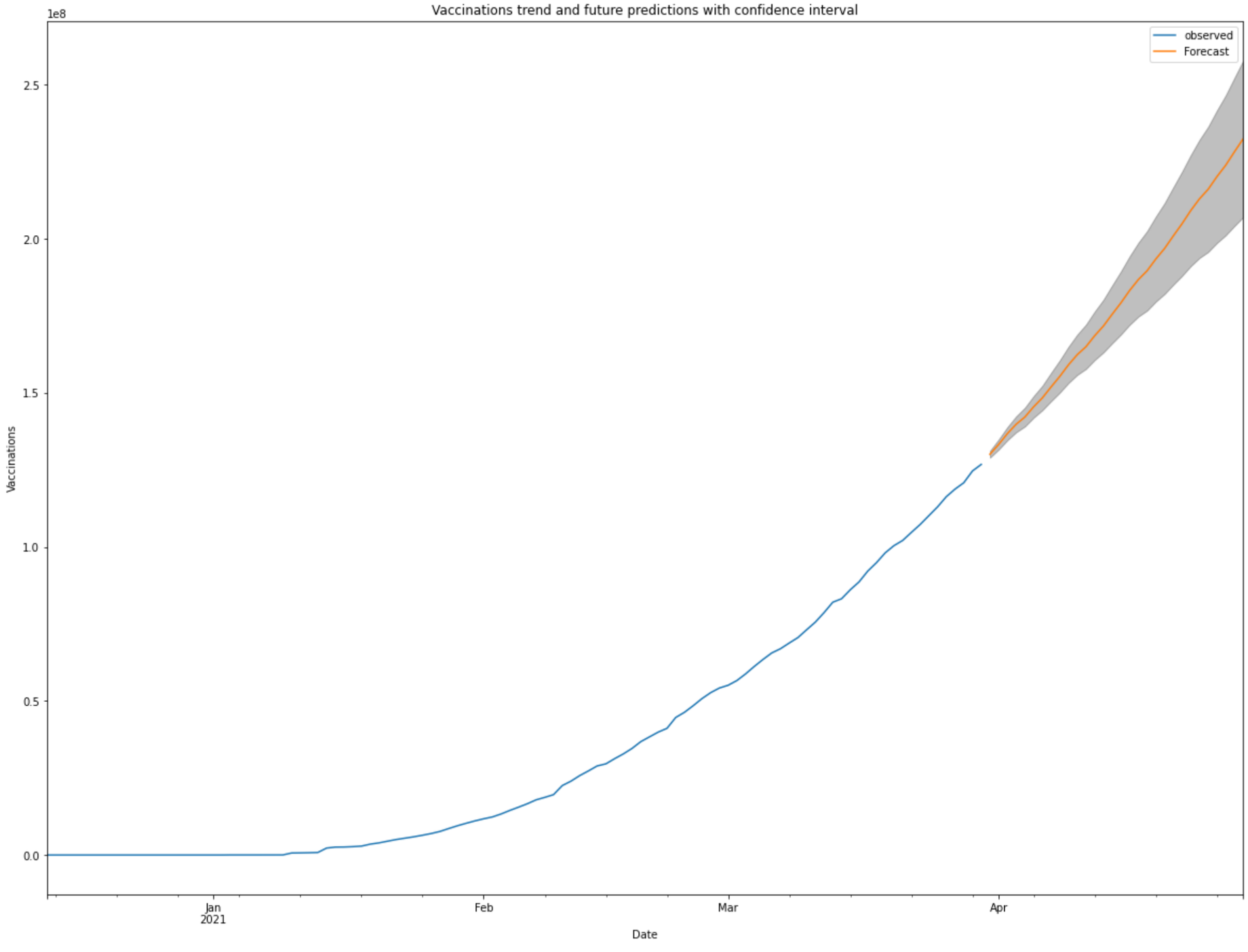

−2
−2
−1
0
1
2
Theoretical Quantiles


−1.00
0
2
4
6
8
10


Vaccinations actual and predicted values with confidence interval
1e8
observed
One-step ahead Forecast
Vaccinations
1.2
1.0
0.8
0.6
0.4
0.2
0.0




```
Vaccinations Root Mean Squared Error:452828.44
Vaccinations Forecast:
2021-03-31    1.300900e+08
2021-04-01    1.332706e+08
2021-04-02    1.367910e+08
2021-04-03    1.398174e+08
2021-04-04    1.421536e+08
2021-04-05    1.454676e+08
2021-04-06    1.483561e+08
2021-04-07    1.519156e+08
2021-04-08    1.553464e+08
2021-04-09    1.591180e+08
2021-04-10    1.623963e+08
2021-04-11    1.649856e+08
2021-04-12    1.685535e+08
2021-04-13    1.716968e+08
2021-04-14    1.755120e+08
2021-04-15    1.791994e+08
2021-04-16    1.832286e+08
2021-04-17    1.867656e+08
2021-04-18    1.896143e+08
2021-04-19    1.934426e+08
2021-04-20    1.968474e+08
2021-04-21    2.009250e+08
2021-04-22    2.048757e+08
2021-04-23    2.091692e+08
2021-04-24    2.129714e+08
2021-04-25    2.160863e+08
2021-04-26    2.201818e+08
2021-04-27    2.238547e+08
2021-04-28    2.282014e+08
2021-04-29    2.324222e+08
Freq: D, Name: predicted_mean, dtype: float64
```


Vaccinations trend and future predictions with confidence interval
1e8
2.5
2.0
1.5
1.0
0.5
0.0
Vaccinations
observed
Forecast
Jan
2021
Feb
Mar
Apr
Date

In [3]:
```
total_vaccinations=round(sum(start_new_vaccinations.values()))
print('Total Global Vaccinations:', total_vaccinations)
```

```
Total Global Vaccinations: 126779833
```

In [2]:
```
forecasted_vaccinations=round(sum(end_new_vaccinations.values()))
print('Forecated Global Vaccinations:', forecasted_vaccinations)
```

```
Forecasted Global Vaccinations: 232422250
```

In [50]:
```
pecentage_change(total_vaccinations,forecasted_vaccinations)
```

Out[50]:
```
'Pecentage Change: 83.32746186848188%'
```

Conclusion

The regression model suggests that cases of and vaccinations for the virus have an inverse relationship. The time-series models suggest that, over the next month, vaccinations will trend upward by 83.3% and cases of the virus will trend upward by 12.8%. The virus spread is slowing and will decrease more as more people get vaccinated.

In [ ]:
```
```