In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob

from bokeh.models import NumeralTickFormatter, DatetimeTickFormatter, ColumnDataSource, HoverTool, CrosshairTool, SaveTool, PanTool, Range1d
from bokeh.plotting import figure, show, output_file
from bokeh.models import Line,TapTool
from bokeh.io import output_notebook, show
output_notebook()

import seaborn as sns

import holoviews.plotting.bokeh
import holoviews as hv
from holoviews import opts
from holoviews import streams
hv.extension('bokeh', width=90)

import xgboost as xgb
from xgboost import plot_importance
import os

import sklearn
```

Loading BokehJS ...

US Daily Reports - EDA

Data source: "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University" url: https://github.com/CSSEGISandData/COVID-19.

There are several datasets provided by Johns Hopkins University. Each has slightly different information The time series files contain all data to-date for each state and county, but only total cases for each location. Daily US files contain a richer dataset, including testing ratio, hospitalizaiton rates etc. But each file is for only one day and the data starts only in April, and no county level data is provided. The daily global files have similar data as the daily US files, but testing ratio and hospitalization rates are not available. Also, the table formats change over time. Not all the columns are avilable in earlier parts of the data.

First we will review the US Daily Reports.

In [2]:
```
# import data

'''
There is a dataformatting issue on April 23. Prior to April 23, the file name matches the date indicated in Last_Update.
However, on April 23, the file's name is 04-23-2020.csv but the Last_Update column is 4/24.

It makes sense because data is compiled in arrears. For instance, 4/24's data would be assembled on 4/25.
This change in the labeling scheme creates a gap in the time series.

Therefore while importing the data, we'll need a new column called Data_Date and append the file-name's listed date.
'''

covid_data = pd.DataFrame()

#this filepath assumes you've downloaded the entire Hopkins repository from Github. Adjust accordingly.

path = "COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports_us/*.csv"

for file in glob.glob(path):
    tmp = pd.read_csv(file)
    file_date = file[-14:-4] #this extracts the date from the filename
    tmp['Data_Date'] = file_date # this adds the filename date as a new column
    covid_data = pd.concat([covid_data,tmp])
```

In [3]:
```
'''
The Data_Date column may be interpreted as text.
Meaning, when we sort by Data_Date, the data may be erroneously sorted as follows: 4/19/20, 4/2/20, 4/20/20, 4/21/20...
This converts Data_Date into date-time format.
'''

covid_data['Data_Date'] = pd.to_datetime(covid_data['Data_Date'], infer_datetime_format = True)
```

In [4]:
```
# Now sort the data by State then Date

covid_data = covid_data.sort_values(by = ['Province_State', 'Data_Date'])
```

In [5]:
```
covid_data.head()
```

Out[5]:

	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	...	People_Tested	People_Hospitalized	Mortality_Rate	UID	ISO3	Testing_Rate	Hospitalization_Rate	Data_Date	Total_Test_Results	C
0	Alabama	US	2020-04-12 23:18:15	32.3182	-86.9023	3667	93	NaN	3470.0	1.0	...	21583.0	437.0	2.610160	84000001.0	USA	460.300152	12.264945	2020-04-12	NaN	
0	Alabama	US	2020-04-13 23:07:54	32.3182	-86.9023	3870	99	NaN	3635.0	1.0	...	29182.0	457.0	2.651312	84000001.0	USA	622.363852	12.238886	2020-04-13	NaN	
0	Alabama	US	2020-04-14 23:33:31	32.3182	-86.9023	4041	114	NaN	3839.0	1.0	...	33117.0	493.0	2.883886	84000001.0	USA	706.285508	12.471541	2020-04-14	NaN	
0	Alabama	US	2020-04-15 22:56:51	32.3182	-86.9023	4307	118	NaN	3957.0	1.0	...	34077.0	525.0	2.895706	84000001.0	USA	726.759406	12.883436	2020-04-15	NaN	
0	Alabama	US	2020-04-16 23:30:51	32.3182	-86.9023	4465	133	NaN	4212.0	1.0	...	36391.0	553.0	3.060990	84000001.0	USA	776.110032	12.727273	2020-04-16	NaN	

5 rows × 21 columns

In [6]:
```
covid_data.columns
```

Out[6]:
```
Index(['Province_State', 'Country_Region', 'Last_Update', 'Lat', 'Long_',
       'Confirmed', 'Deaths', 'Recovered', 'Active', 'FIPS', 'Incident_Rate',
       'People_Tested', 'People_Hospitalized', 'Mortality_Rate', 'UID', 'ISO3',
       'Testing_Rate', 'Hospitalization_Rate', 'Data_Date',
       'Total_Test_Results', 'Case_Fatality_Ratio'],
      dtype='object')
```

In [7]:
```
'''
Create a DataFrame that will hold Incident_Rate for each US State/Territory and date.
'''

incident_df = pd.DataFrame(covid_data['Data_Date'].unique(), columns = ['Data_Date'])

states = covid_data['Province_State'].unique()

for state in states:
    tmp = covid_data[covid_data["Province_State"]==state][['Data_Date','Incident_Rate']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "Incident_Rate": state})
    incident_df = pd.merge(incident_df,tmp, on="Data_Date", how = 'left')
```

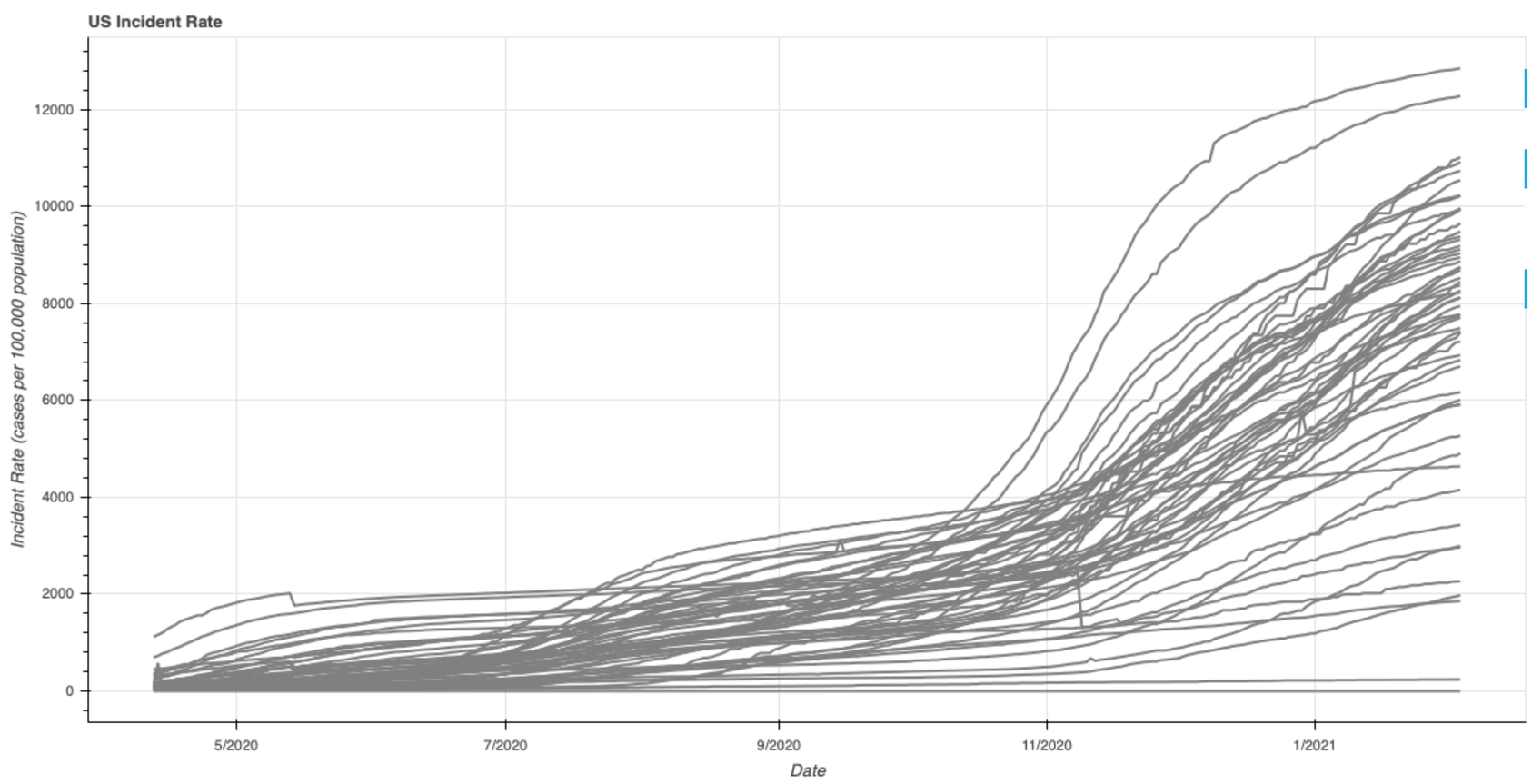
In [8]:
```
incident_df.head()
```

Out[8]:

	Data_Date	Alabama	Alaska	American Samoa	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	...	Tennessee	Texas	Utah	Vermont	Virgin Islands	Virginia	Washington	West Virginia	Wisconsin	Wyoming
0	2020-04-12	75.988020	45.504049	0.0	48.662422	49.439423	58.137726	128.943729	337.560482	166.878217	...	83.900374	59.505161	80.359205	119.064971	47.544468	66.698452	140.527668	44.822441	64.565739	54.299735
1	2020-04-13	79.634933	46.340521	0.0	50.901828	54.460614	61.035048	135.720025	375.313404	180.536557	...	85.454085	62.106907	82.452802	122.504262	47.544468	72.680319	140.872066	46.182987	66.247038	55.305285
2	2020-04-14	84.305541	47.678875	0.0	52.330651	57.859574	64.669453	140.290495	392.366730	197.789197	...	88.759528	65.287303	84.337038	123.159365	47.544468	78.181614	143.044423	48.374979	68.701347	56.713056
3	2020-04-15	86.907433	49.017230	0.0	54.460147	60.601917	68.061564	140.396374	413.851676	206.826295	...	88.759528	69.207326	88.908057	124.305796	47.544468	82.203249	144.938613	53.061305	71.909343	57.718607
4	2020-04-16	92.665716	50.188290	0.0	58.210808	62.571769	70.589069	146.219754	445.518131	212.577175	...	97.106915	73.423199	93.618649	126.762432	47.544468	87.122798	146.461912	55.026538	74.885435	59.528598

5 rows × 60 columns

In [9]:
```
'''
Plot a spaghetti-chart with all locations.
Clicking on a line will highlight it in red for easier viewing.
This works as expected except the hovertool interpolates between points. This is due to the use of $y in source.
'''

source = ColumnDataSource(incident_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="US Incident Rate", x_axis_type="datetime")
renderer = p.line('Data_Date','Maryland', source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("Incident Rate", "$y{0,0}"), ("State", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for state in states:

    renderer = p.line('Data_Date', state, source = source, line_color = 'gray', line_width=2, name = state)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Incident Rate (cases per 100,000 population)'

show(p)
```



In [10]:
```
'''
There are some oddities. For instance, the top-most line in the May 2020 timeframe is New York.
There is a kink in the incident rate between 5/13/2020 and 5/14/2020. The raw csv files show this drop, so the chart is correct.
Perhaps a change in methodology or data source is the cause of this kink?
'''
```

Out[10]:
```
'\nThere are some oddities. For instance, the top-most line in the May 2020 timeframe is New York. \nThere is a kink in the incident rate between 5/13/2020 and 5/14/2020. The raw csv files show this drop, so th
e chart is correct.\nPerhaps a change in methodology or data source is the cause of this kink?\n'
```

In [11]:
```
# Now will contstuct a heatmap showing the change in incident rate by week. There is also a line chart which allows you to focus in on a particular State & week.

incident_hm_df = pd.melt(incident_df, id_vars = ['Data_Date'])
cols = ['Data_Date', 'State', 'Incident_Rate']
incident_hm_df.columns = cols
incident_hm_df
```

Out[11]:

	Data_Date	State	Incident_Rate
0	2020-04-12	Alabama	75.988020
1	2020-04-13	Alabama	79.634933
2	2020-04-14	Alabama	84.305541
3	2020-04-15	Alabama	86.907433
4	2020-04-16	Alabama	92.665716
...	...	...	...
17577	2021-01-30	Wyoming	8933.597577
17578	2021-01-31	Wyoming	8969.536543
17579	2021-02-01	Wyoming	8994.590149
17580	2021-02-02	Wyoming	9006.857777
17581	2021-02-03	Wyoming	9034.503135

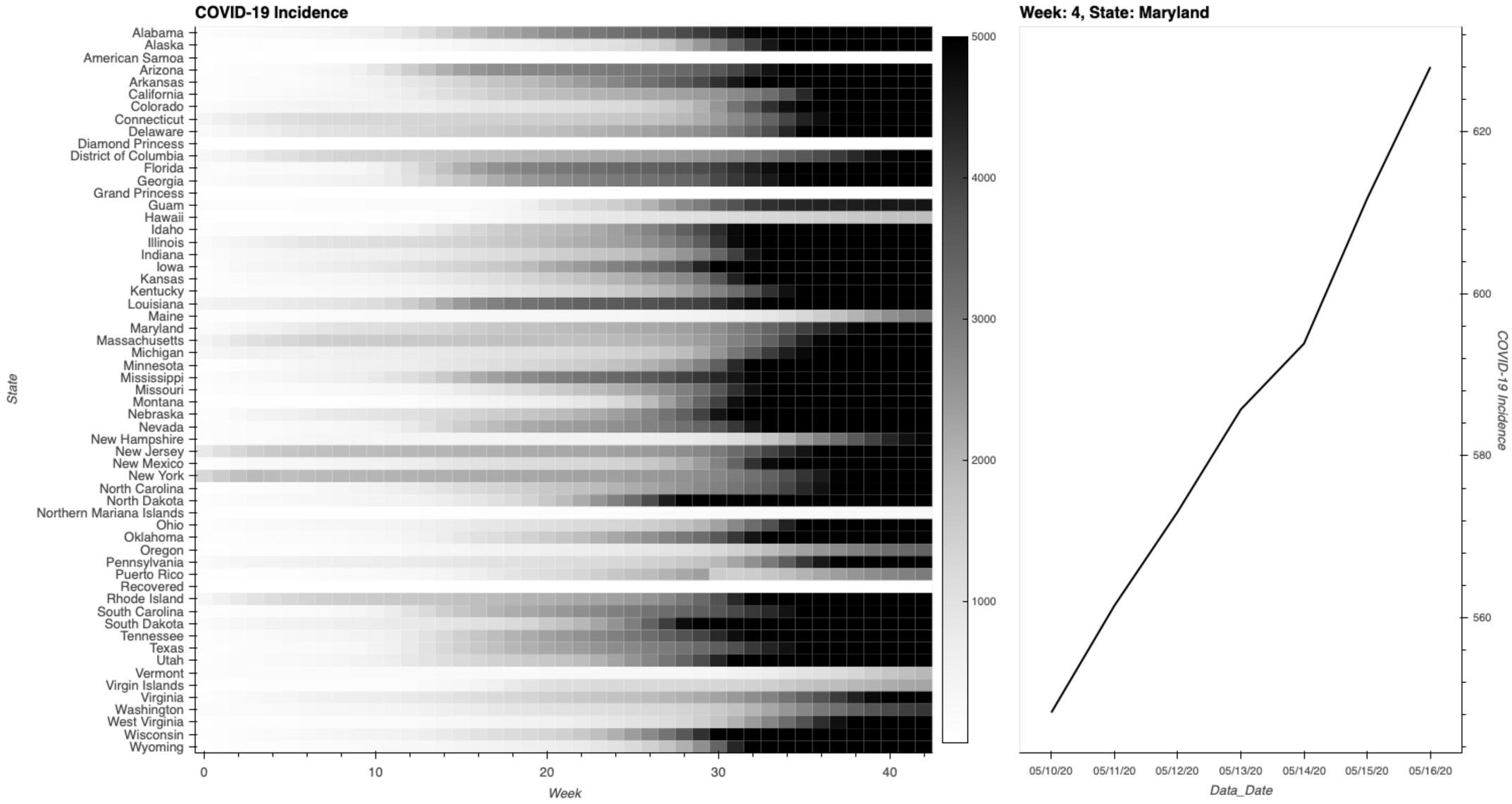
17582 rows × 3 columns

In [12]:
```
# Adding a Week column to the DataFrame for the heatmap
incident_hm_df['Start'] = '04/12/20'
incident_hm_df['Week'] = (pd.to_datetime(incident_hm_df['Data_Date']) - pd.to_datetime(incident_hm_df['Start']))//np.timedelta64(1,'W')
```

In [13]:
```
incident_hm_df.drop(['Start'], axis = 1, inplace = True)
```

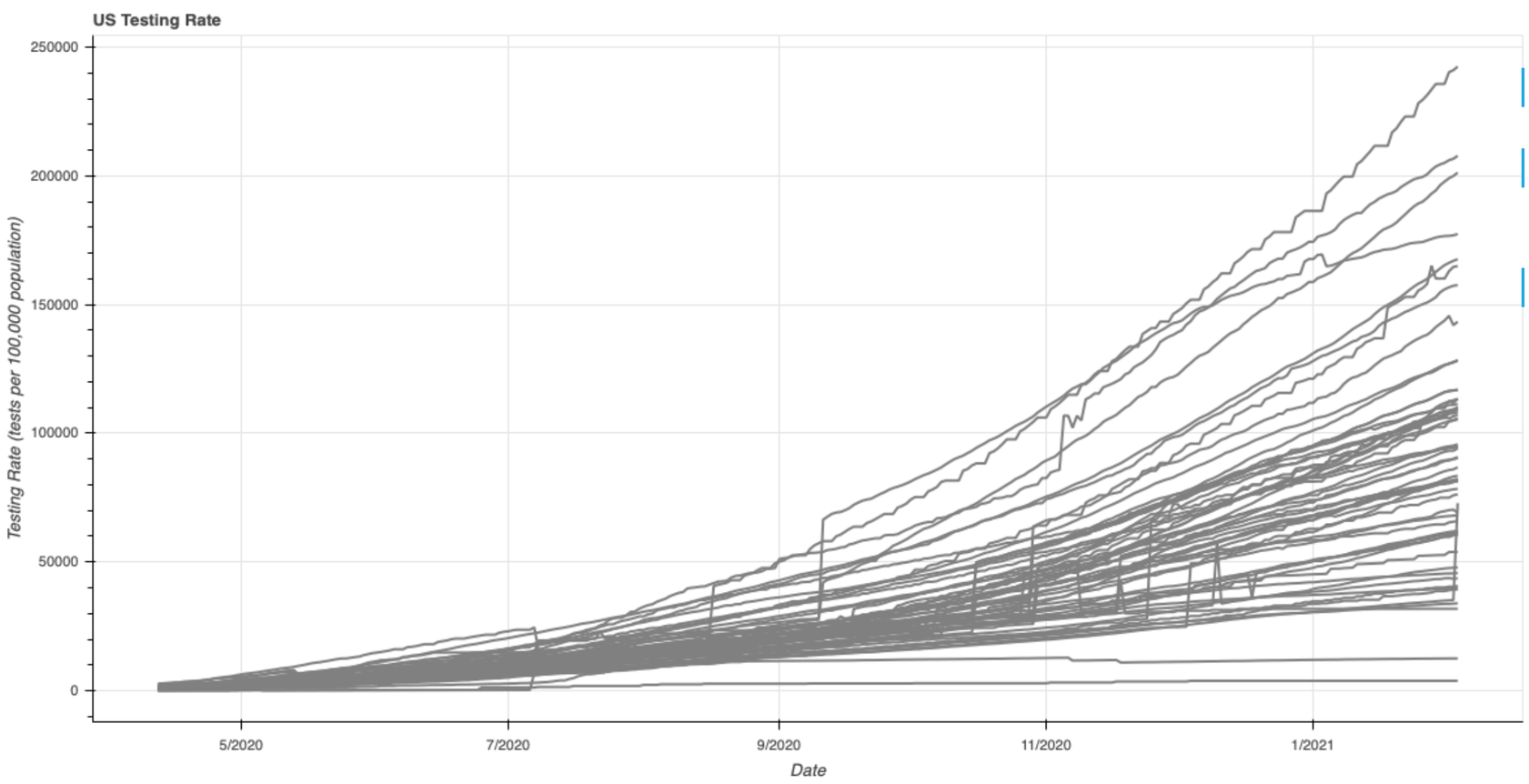
In [14]:
```
'''
This generates a heatmap for each week in the dataset and all locations.
Clicking on a cell in the heatmap will generate a line chart with daily incident rates for the selected week.
'''

%opts HeatMap [width=1000 height=800 logz=False fontsize={'xticks': '10pt', 'yticks': '10pt'}, tools=['hover'], xrotation=0, invert_yaxis=True,yaxis='left', colorbar=True, clim=(1, 5000)] (cmap='Greys')
%opts Curve [width=500, height=800, yaxis='right', tools=['hover']] (line_color='black') {+framewise}

dataset = incident_hm_df.groupby(['State','Data_Date','Week']).sum().reset_index()
dataset['Data_Date'] = dataset['Data_Date'].dt.strftime('%m/%d/%y')
dateset = dataset[['Week','Data_Date','State','Incident_Rate']]

dataset = hv.Dataset(dataset, vdims=[('Incident_Rate','COVID-19 Incidence')])

heatmap = hv.HeatMap(dataset.aggregate(['Week', 'State'], np.mean),
                     label='COVID-19 Incidence')

posxy = hv.streams.Tap(source=heatmap, x=4, y='Maryland')

def tap_histogram(x, y):
    x = int(x)
    return hv.Curve(dataset.select(State=y, Week=x), kdims=['Data_Date'],
                    label='Week: %s, State: %s' % (x, y))

heatmap + hv.DynamicMap(tap_histogram, kdims=[], streams=[posxy])
```

Out[14]:



In [15]:

```
# now let's look at other metrics, starting with testing rate.
testing_df = pd.DataFrame(covid_data['Data_Date'].unique(), columns = ['Data_Date'])

states = covid_data['Province_State'].unique()

for state in states:
    tmp = covid_data[covid_data["Province_State"]==state][['Data_Date','Testing_Rate']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "Testing_Rate": state})
    testing_df = pd.merge(testing_df,tmp, on="Data_Date", how = 'left')

source = ColumnDataSource(testing_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="US Testing Rate", x_axis_type="datetime")
renderer = p.line('Data_Date','Maryland' , source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("Testing Rate", "$y{0,0}"), ("State", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for state in states:

    renderer = p.line('Data_Date', state, source = source, line_color = 'gray', line_width=2, name = state)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Testing Rate (tests per 100,000 population)'

p.yaxis.formatter.use_scientific=False

show(p)
```



In [16]:

```
'''
There are some oddities here as well. For instance, there is a large jump in one line between 9/10/20 and 9/11/20.
This line corresponds to North Dakota. Looking at the raw data confirms this jump.
Might be a change in methodology or data source?
'''
```

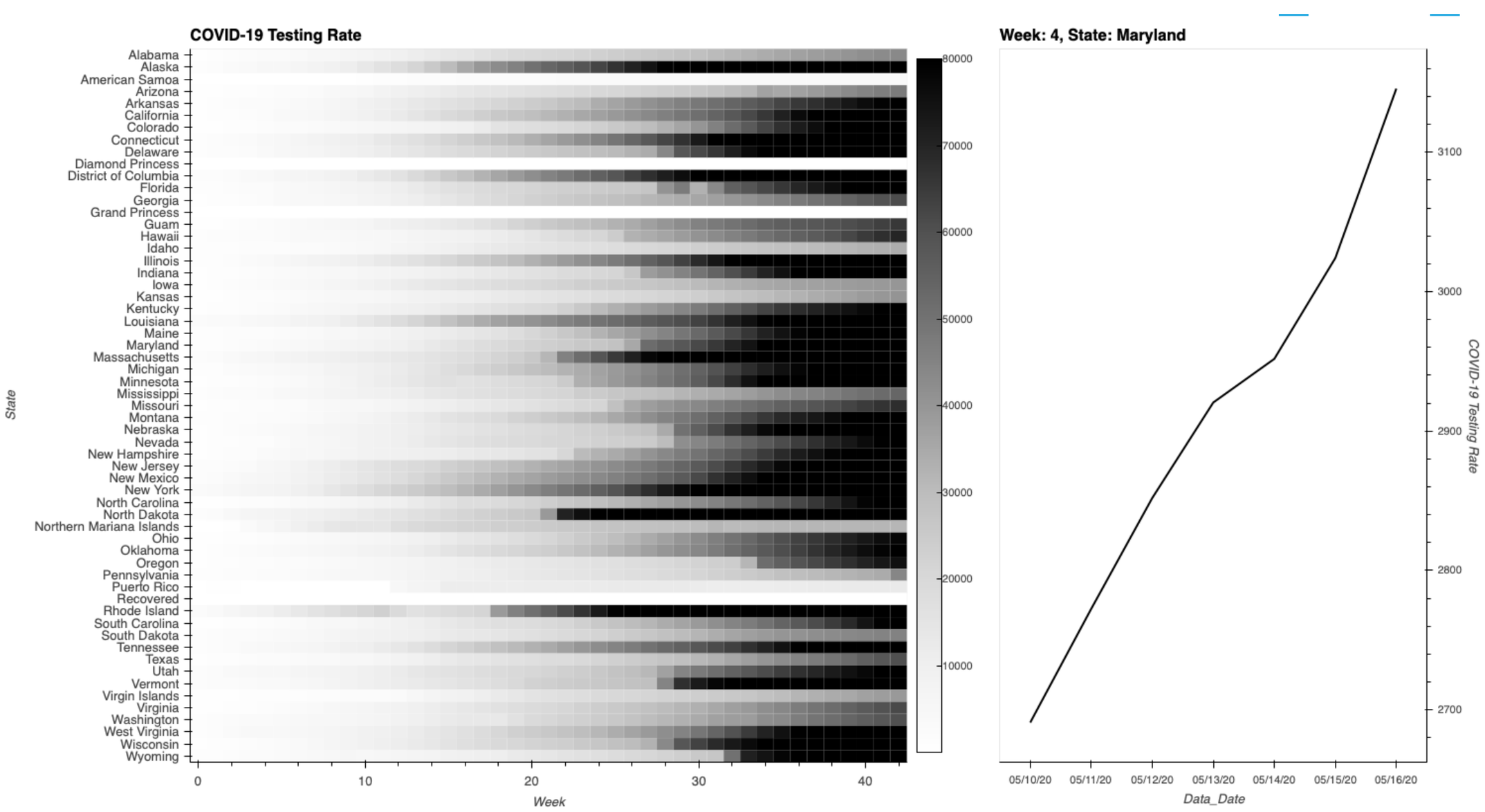
Out[16]:

```
'\nThere are some oddities here as well. For instance, there is a large jump in one line between 9/10/20 and 9/11/20.\nThis line corresponds to North Dakota. Looking at the raw data confirms this jump. \nMight be a change in methodology or data source?\n'
```

In [17]:

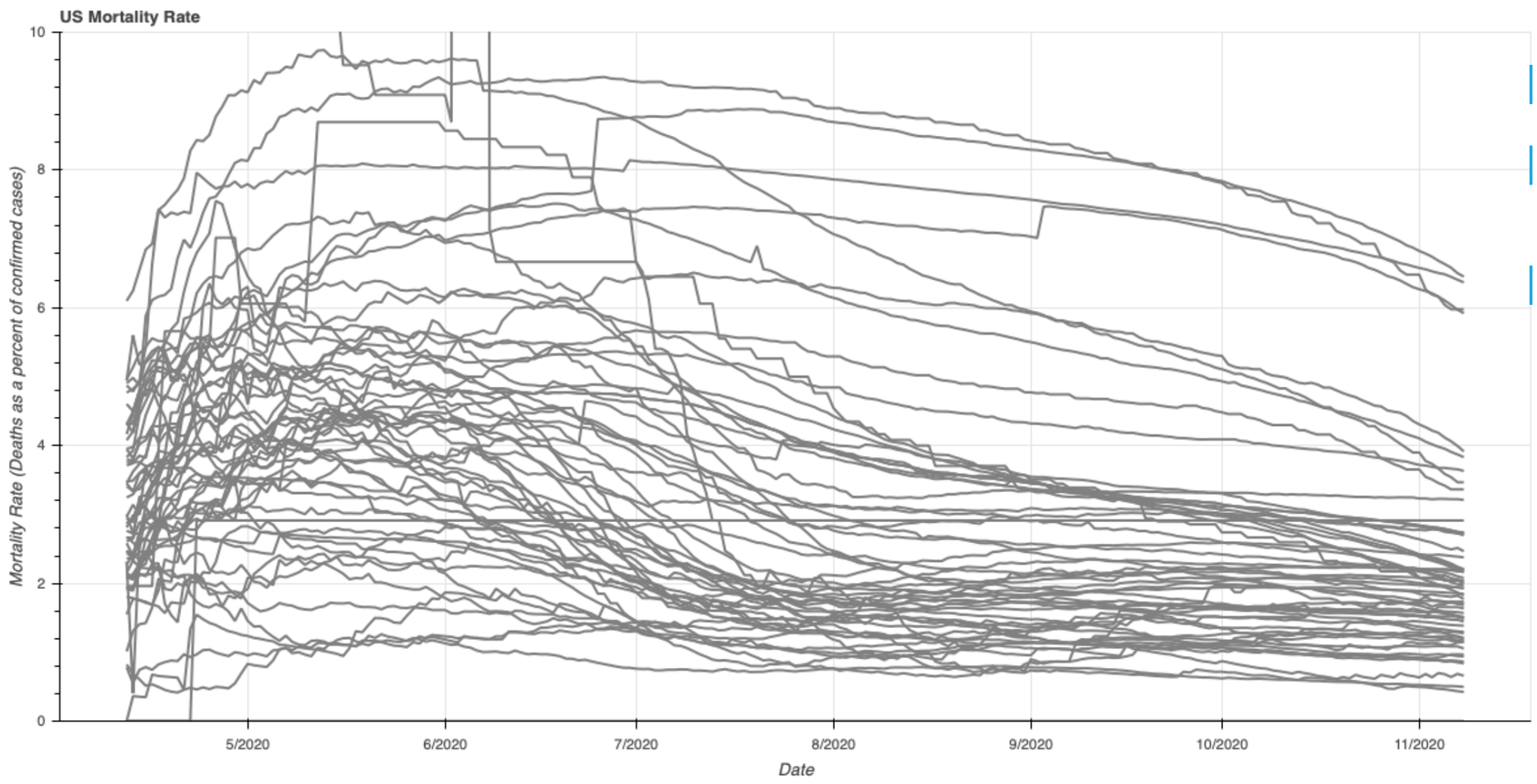
```
# now let's make a heatmap of this same chart.

testing_hm_df = pd.melt(testing_df, id_vars = ['Data_Date'])
cols = ['Data_Date', 'State', 'Testing_Rate']
testing_hm_df.columns = cols
testing_hm_df

testing_hm_df['Start'] = '04/12/20'
testing_hm_df['Week'] = (pd.to_datetime(testing_hm_df['Data_Date']) - pd.to_datetime(testing_hm_df['Start']))//np.timedelta64(1,'W')
testing_hm_df.drop(['Start'], axis = 1, inplace = True)

%opts HeatMap [width=1000 height=800 logz=False fontsize={'xticks': '10pt', 'yticks': '10pt'}, tools=['hover'], xrotation=0, invert_yaxis=True,yaxis='left', colorbar=True, clim=(1, 80000)] (cmap='Greys')
%opts Curve [width=500, height=800, yaxis='right', tools=['hover']] (line_color='black') {+framewise}

dataset = testing_hm_df.groupby(['State','Data_Date','Week']).sum().reset_index()
dataset['Data_Date'] = dataset['Data_Date'].dt.strftime('%m/%d/%y')
dateset = dataset[['Week','Data_Date','State','Testing_Rate']]


dataset = hv.Dataset(dataset, vdims=[('Testing_Rate','COVID-19 Testing Rate')])

heatmap = hv.HeatMap(dataset.aggregate(['Week', 'State'], np.mean),
                     label='COVID-19 Testing Rate')

posxy = hv.streams.Tap(source=heatmap, x=4, y='Maryland')

def tap_histogram(x, y):
    x = int(x)
    return hv.Curve(dataset.select(State=y, Week=x), kdims=['Data_Date'],
                    label='Week: %s, State: %s' % (x, y))

heatmap + hv.DynamicMap(tap_histogram, kdims=[], streams=[posxy])
```

Out[17]:



In [18]:

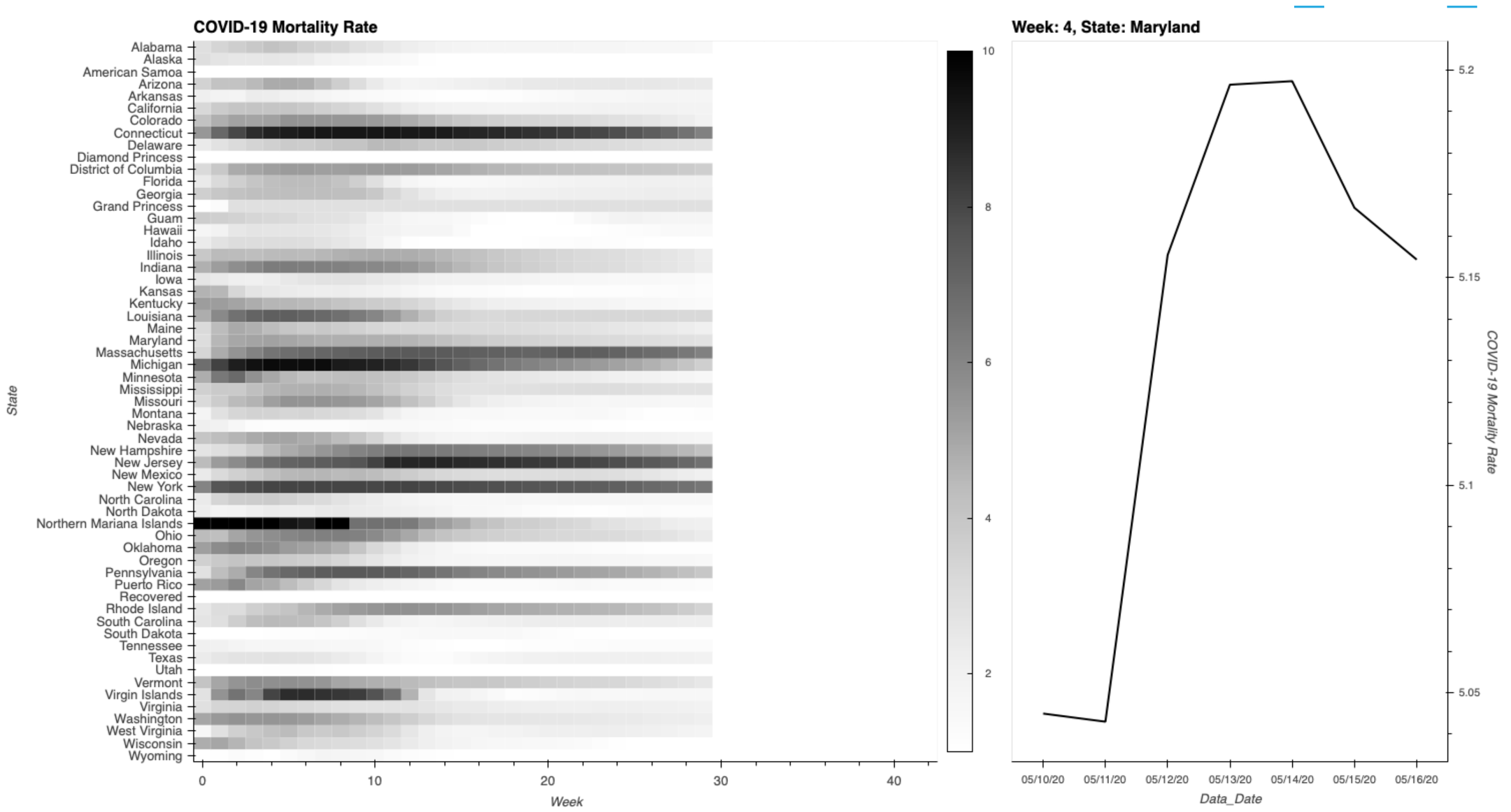
```
# now let's look at mortality
mortality_df = pd.DataFrame(covid_data['Data_Date'].unique(), columns = ['Data_Date'])
states = covid_data['Province_State'].unique()

for state in states:
    tmp = covid_data[covid_data["Province_State"]==state][['Data_Date','Mortality_Rate']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "Mortality_Rate": state})
    mortality_df = pd.merge(mortality_df,tmp, on="Data_Date", how = 'left')

source = ColumnDataSource(mortality_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="US Mortality Rate", x_axis_type="datetime")
renderer = p.line('Data_Date','Maryland' , source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("Mortality Rate", "$y{0,0.000}"), ("State", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for state in states:

    renderer = p.line('Data_Date', state, source = source, line_color = 'gray', line_width=2, name = state)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Mortality Rate (Deaths as a percent of confirmed cases)'

# North Mariana Islands is distorting the the chart, so we'll scale the y-axis range
p.y_range=Range1d(0, 10)

show(p)
```



In [19]:

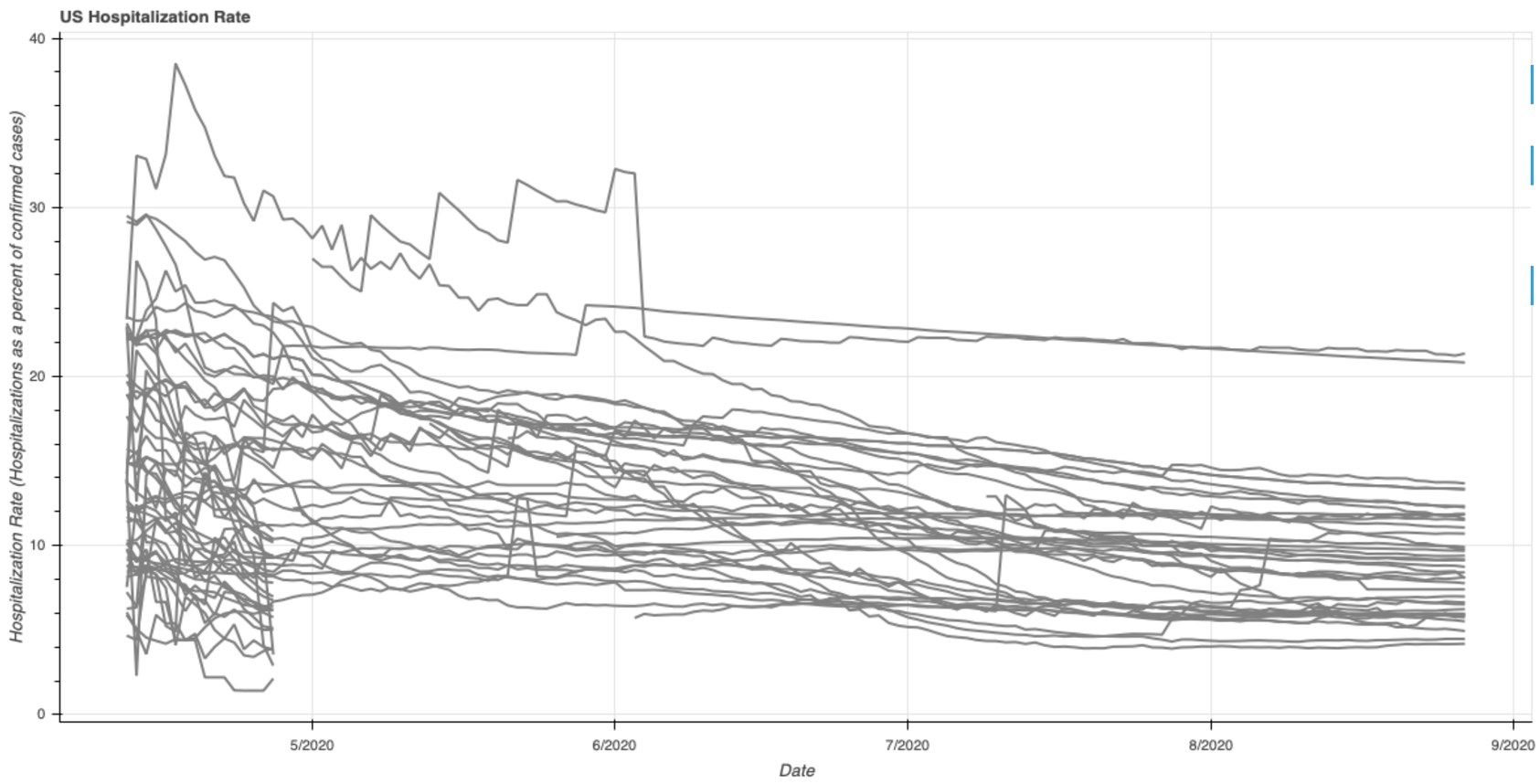
```
# now let's make a heatmap of this same chart.

mortality_hm_df = pd.melt(mortality_df, id_vars = ['Data_Date'])
cols = ['Data_Date', 'State', 'Mortality_Rate']
mortality_hm_df.columns = cols

mortality_hm_df['Start'] = '04/12/20'
mortality_hm_df['Week'] = (pd.to_datetime(mortality_hm_df['Data_Date']) - pd.to_datetime(mortality_hm_df['Start']))//np.timedelta64(1,'W')
mortality_hm_df.drop(['Start'], axis = 1, inplace = True)

%opts HeatMap [width=1000 height=800 logz=False fontsize={'xticks': '10pt', 'yticks': '10pt'}, tools=['hover'], xrotation=0, invert_yaxis=True,yaxis='left', colorbar=True, clim=(1, 10)] (cmap='Greys')
%opts Curve [width=500, height=800, yaxis='right', tools=['hover']] (line_color='black') {+framewise}

dataset = mortality_hm_df.groupby(['State','Data_Date','Week']).sum().reset_index()
dataset['Data_Date'] = dataset['Data_Date'].dt.strftime('%m/%d/%y')
dateset = dataset[['Week','Data_Date','State','Mortality_Rate']]


dataset = hv.Dataset(dataset, vdims=[('Mortality_Rate','COVID-19 Mortality Rate')])

heatmap = hv.HeatMap(dataset.aggregate(['Week', 'State'], np.mean),
                     label='COVID-19 Mortality Rate')

posxy = hv.streams.Tap(source=heatmap, x=4, y='Maryland')

def tap_histogram(x, y):
    x = int(x)
    return hv.Curve(dataset.select(State=y, Week=x), kdims=['Data_Date'],
                    label='Week: %s, State: %s' % (x, y))

heatmap + hv.DynamicMap(tap_histogram, kdims=[], streams=[posxy])
```

Out[19]:



In [20]:

```
# now let's look at hospitalization
hospitalization_df = pd.DataFrame(covid_data['Data_Date'].unique(), columns = ['Data_Date'])
states = covid_data['Province_State'].unique()

for state in states:
    tmp = covid_data[covid_data["Province_State"]==state][['Data_Date','Hospitalization_Rate']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "Hospitalization_Rate": state})
    hospitalization_df = pd.merge(hospitalization_df,tmp, on="Data_Date", how = 'left')

source = ColumnDataSource(hospitalization_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="US Hospitalization Rate", x_axis_type="datetime")
renderer = p.line('Data_Date','Maryland' , source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("Hospitalization Rate", "$y{0,0}"), ("State", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for state in states:

    renderer = p.line('Data_Date', state, source = source, line_color = 'gray', line_width=2, name = state)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Hospitalization Rate (Hospitalizations as a percent of confirmed cases)'

show(p)
```

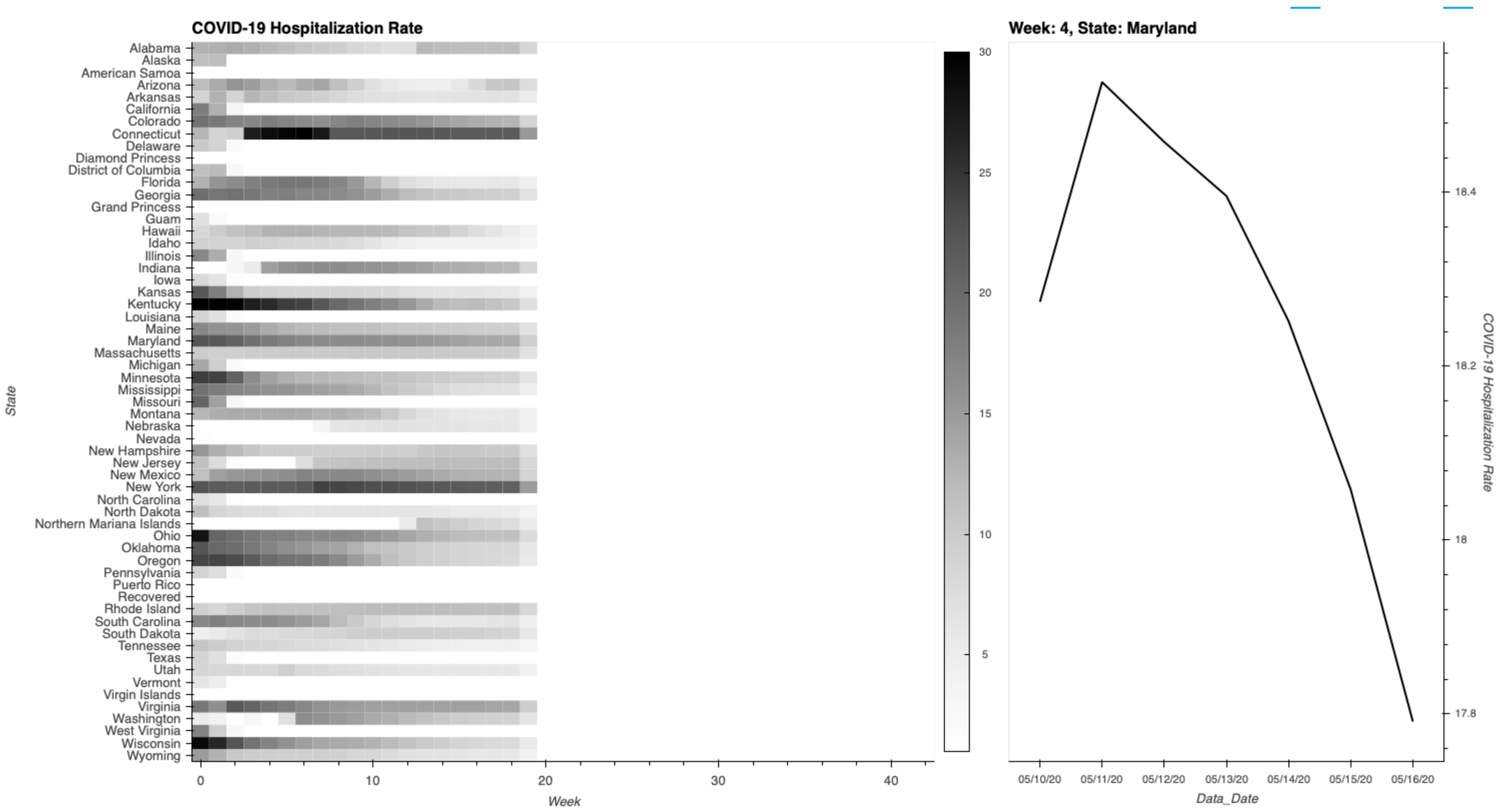


In [21]:
```
'''
Many lines end abruptly and the remaining lines end at 8/31/20.
The GitHub repository notes that Hospitalization Rate was dropped from the dataset.
'''
```

Out[21]:
```
'\nMany lines end abruptly and the remaining lines end at 8/31/20.\nThe GitHub repository notes that Hospitalization Rate was dropped from the dataset.\n'
```

In [22]:
```
# now let's make a heatmap of this same chart.

hospitalization_hm_df = pd.melt(hospitalization_df, id_vars = ['Data_Date'])
cols = ['Data_Date', 'State', 'Hospitalization_Rate']
hospitalization_hm_df.columns = cols

hospitalization_hm_df['Start'] = '04/12/20'
hospitalization_hm_df['Week'] = (pd.to_datetime(hospitalization_hm_df['Data_Date']) - pd.to_datetime(hospitalization_hm_df['Start']))//np.timedelta64(1,'W')
hospitalization_hm_df.drop(['Start'], axis = 1, inplace = True)

%opts HeatMap [width=1000 height=800 logz=False fontsize={'xticks': '10pt', 'yticks': '10pt'}, tools=['hover'], xrotation=0, invert_yaxis=True,yaxis='left', colorbar=True, clim=(1, 30)] (cmap='Greys')
%opts Curve [width=500, height=800, yaxis='right', tools=['hover']] (line_color='black') {+framewise}

dataset = hospitalization_hm_df.groupby(['State','Data_Date','Week']).sum().reset_index()
dataset['Data_Date'] = dataset['Data_Date'].dt.strftime('%m/%d/%y')
dateset = dataset[['Week','Data_Date','State','Hospitalization_Rate']]


dataset = hv.Dataset(dataset, vdims=[('Hospitalization_Rate','COVID-19 Hospitalization Rate')])

heatmap = hv.HeatMap(dataset.aggregate(['Week', 'State'], np.mean),
                     label='COVID-19 Hospitalization Rate')

posxy = hv.streams.Tap(source=heatmap, x=4, y='Maryland')

def tap_histogram(x, y):
    x = int(x)
    return hv.Curve(dataset.select(State=y, Week=x), kdims=['Data_Date'],
                    label='Week: %s, State: %s' % (x, y))

heatmap + hv.DynamicMap(tap_histogram, kdims=[], streams=[posxy])
```

Out[22]:



Daily Global Reports

In [23]:
```
'''
Although the daily global reports lack certain datapoints compared to the US Daily Reports, the global files include granular US county-level data.
'''
```

Out[23]:
```
'\nAlthough the daily global reports lack certain datapoints compared to the US Daily Reports, the global files include granular US county-level data.\n'
```

In [24]:
```
granular_covid_data = pd.read_csv("COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/10-03-2020.csv")
granular_covid_data.head()
```

Out[24]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_Rate	Case-Fatality_Ratio
0	NaN	NaN	NaN	Afghanistan	2020-10-04 04:23:44	33.93911	67.709953	39297	1464	32842	4993.0	Afghanistan	100.947020	3.720386
1	NaN	NaN	NaN	Albania	2020-10-04 04:23:44	41.15330	20.168300	14117	392	8536	5189.0	Albania	490.548336	2.776794
2	NaN	NaN	NaN	Algeria	2020-10-04 04:23:44	28.03390	1.659600	51995	1756	36482	13757.0	Algeria	118.571866	3.377248
3	NaN	NaN	NaN	Andorra	2020-10-04 04:23:44	42.50630	1.521800	2110	53	1540	517.0	Andorra	2730.861321	2.511848
4	NaN	NaN	NaN	Angola	2020-10-04 04:23:44	-11.20270	17.873900	5370	193	2436	2741.0	Angola	16.338941	3.594041

In [25]:

```python
# problem is that the columns and column names in the files are not consistent

path = "COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/*.csv"

for file in glob.glob(path):
    tmp = pd.read_csv(file)
    print(tmp.columns)
```

```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
```

```
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
```

```
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
```

```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
```

```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
```

```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
```

```
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
```

```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
```

```
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered', 'Latitude', 'Longitude'],
      dtype='object')
Index(['Province/State', 'Country/Region', 'Last Update', 'Confirmed',
       'Deaths', 'Recovered'],
      dtype='object')
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incident_Rate', 'Case_Fatality_Ratio'],
      dtype='object')
```

In [26]:

```
# this creates a placeholder DataFrame to which each daily data file is concatenated
cols = ['FIPS', 'Admin2','Province_State','Country_Region']
granular_covid_data = granular_covid_data[cols]
```

In [27]:

```
#this filepath assumes you've downloaded the entire Hopkins repository from Github. Adjust accordingly.

path = "COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/*.csv"

for file in glob.glob(path):
    tmp = pd.read_csv(file)

    # these try statements are needed to correct inconsistencies in column names across files
    try:
        tmp['Province_State']
    except KeyError:
        tmp = tmp.rename(columns={'Province/State': 'Province_State', 'Country/Region': 'Country_Region', 'Last Update': 'Last_Update'})
        tmp['Admin2'] = np.NaN
        tmp['FIPS'] = np.NaN
        tmp = tmp.astype({'Admin2': 'str', 'FIPS': 'float64'})

    try:
        tmp['Lat']
    except KeyError:
        tmp = tmp.rename(columns={'Latitude': 'Lat', 'Longitude': 'Long_'})

    try:
        tmp['Incidence_Rate']
    except KeyError:
        tmp = tmp.rename(columns={'Incident_Rate': 'Incidence_Rate'})

    try:
        tmp['Case_Fatality_Ratio']
    except KeyError:
        tmp = tmp.rename(columns={'Case-Fatality_Ratio': 'Case_Fatality_Ratio'})

    file_date = file[-14:-4] #this extracts the date from the filename
    tmp['Data_Date'] = file_date # this adds the filename date as a new column

    tmp['Data_Date'] = pd.to_datetime(tmp['Data_Date'])

    granular_covid_data = pd.concat([granular_covid_data,tmp])
```

In [28]:

```
granular_covid_data = granular_covid_data.sort_values(by = ['Country_Region', 'Province_State', 'Admin2', 'Data_Date']).reset_index().drop(['index'], axis = 1)
granular_covid_data
```

Out[28]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_Rate	Case_Fatality_Ratio	Data_Date
0	NaN	nan	NaN	Azerbaijan	2020-02-28T15:03:26	NaN	NaN	1.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-02-28
1	NaN	nan	NaN	Afghanistan	2020-02-24T23:33:02	NaN	NaN	1.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-02-24
2	NaN	nan	NaN	Afghanistan	2020-02-24T23:33:02	NaN	NaN	1.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-02-25
3	NaN	nan	NaN	Afghanistan	2020-02-24T23:33:02	NaN	NaN	1.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-02-26
4	NaN	nan	NaN	Afghanistan	2020-02-24T23:33:02	NaN	NaN	1.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-02-27
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1154387	NaN	nan	NaN	occupied Palestinian territory	2020-03-11T20:53:02	31.9522	35.2332	0.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-12
1154388	NaN	nan	NaN	occupied Palestinian territory	2020-03-11T20:53:02	31.9522	35.2332	0.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-14
1154389	NaN	nan	NaN	occupied Palestinian territory	2020-03-11T20:53:02	31.9522	35.2332	0.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-15
1154390	NaN	nan	NaN	occupied Palestinian territory	2020-03-11T20:53:02	31.9522	35.2332	0.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-16
1154391	NaN	nan	NaN	occupied Palestinian territory	2020-03-11T20:53:02	31.9522	35.2332	0.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-17

1154392 rows × 15 columns

In [29]:

```
# We will be using XGBoost later in the notebook to observe the utility of certain features for predicting future cases.
# However, XGBoost doesn't like a datetime dtype, so we'll convert to an integer. This will measure time as days since March 22, 2020.
# March 22 is selected because it is the first date in the dataset where States/Counties are clearly split and identified with their FIPS code.
granular_covid_data['Start'] = '2020-03-22'
granular_covid_data['Date_Block'] = (pd.to_datetime(granular_covid_data['Data_Date']) - pd.to_datetime(granular_covid_data['Start']))//np.timedelta64(1,'D')

# new cases per day comparing each row (date) to the prior row (prior day)

granular_covid_data['New_Cases'] = granular_covid_data['Confirmed'].diff(periods = 1).fillna(0).clip(lower = 0)
```

In [30]:

```
# let's take a look at MD

md_granular = granular_covid_data[(granular_covid_data['Country_Region'] == 'US') & (granular_covid_data['Province_State'] == 'Maryland')]
md_granular
```

Out[30]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_Rate	Case_Fatality_Ratio	Data_Date	Start	Date_Block	New_Cases
539013	24001.0	Allegany	Maryland	US	3/22/20 23:45	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-22	2020-03-22	0.0	0.0
539014	24001.0	Allegany	Maryland	US	2020-03-23 23:19:34	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-23	2020-03-22	1.0	0.0
539015	24001.0	Allegany	Maryland	US	2020-03-24 23:37:31	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-24	2020-03-22	2.0	0.0
539016	24001.0	Allegany	Maryland	US	2020-03-25 23:33:19	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-25	2020-03-22	3.0	0.0
539017	24001.0	Allegany	Maryland	US	2020-03-26 23:48:35	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-26	2020-03-22	4.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
546651	NaN	nan	Maryland	US	2020-03-17T16:13:14	39.063900	-76.802100	60.0	0.0	3.0	NaN	NaN	NaN	NaN	2020-03-17	2020-03-22	-5.0	19.0
546652	NaN	nan	Maryland	US	2020-03-18T19:14:34	39.063900	-76.802100	85.0	0.0	0.0	NaN	NaN	NaN	NaN	2020-03-18	2020-03-22	-4.0	25.0
546653	NaN	nan	Maryland	US	2020-03-19T21:43:03	39.063900	-76.802100	107.0	1.0	0.0	NaN	NaN	NaN	NaN	2020-03-19	2020-03-22	-3.0	22.0
546654	NaN	nan	Maryland	US	2020-03-20T16:43:10	39.063900	-76.802100	149.0	1.0	0.0	NaN	NaN	NaN	NaN	2020-03-20	2020-03-22	-2.0	42.0
546655	NaN	nan	Maryland	US	2020-03-21T23:13:18	39.063900	-76.802100	193.0	2.0	0.0	NaN	NaN	NaN	NaN	2020-03-21	2020-03-22	-1.0	44.0

7643 rows × 18 columns

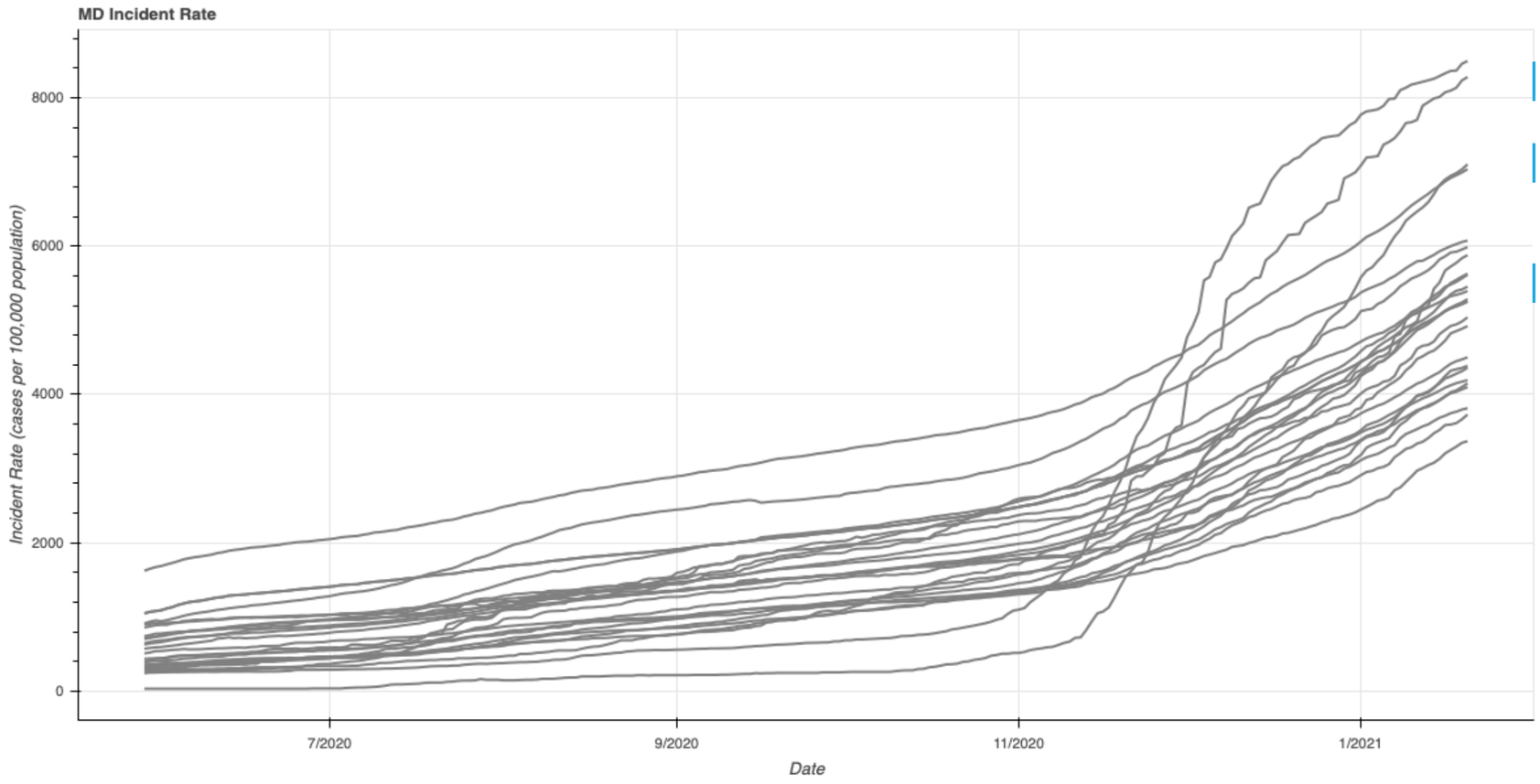
In [31]:

```
# There are many rows without a FIPS code; this occured in daily files where State/County data wasn't clearly separated.
# This drops the rows without a FIPS code.
md_granular = md_granular.dropna(subset=['Date_Block','FIPS']).reset_index().drop(['index'], axis=1)
```

In [32]:

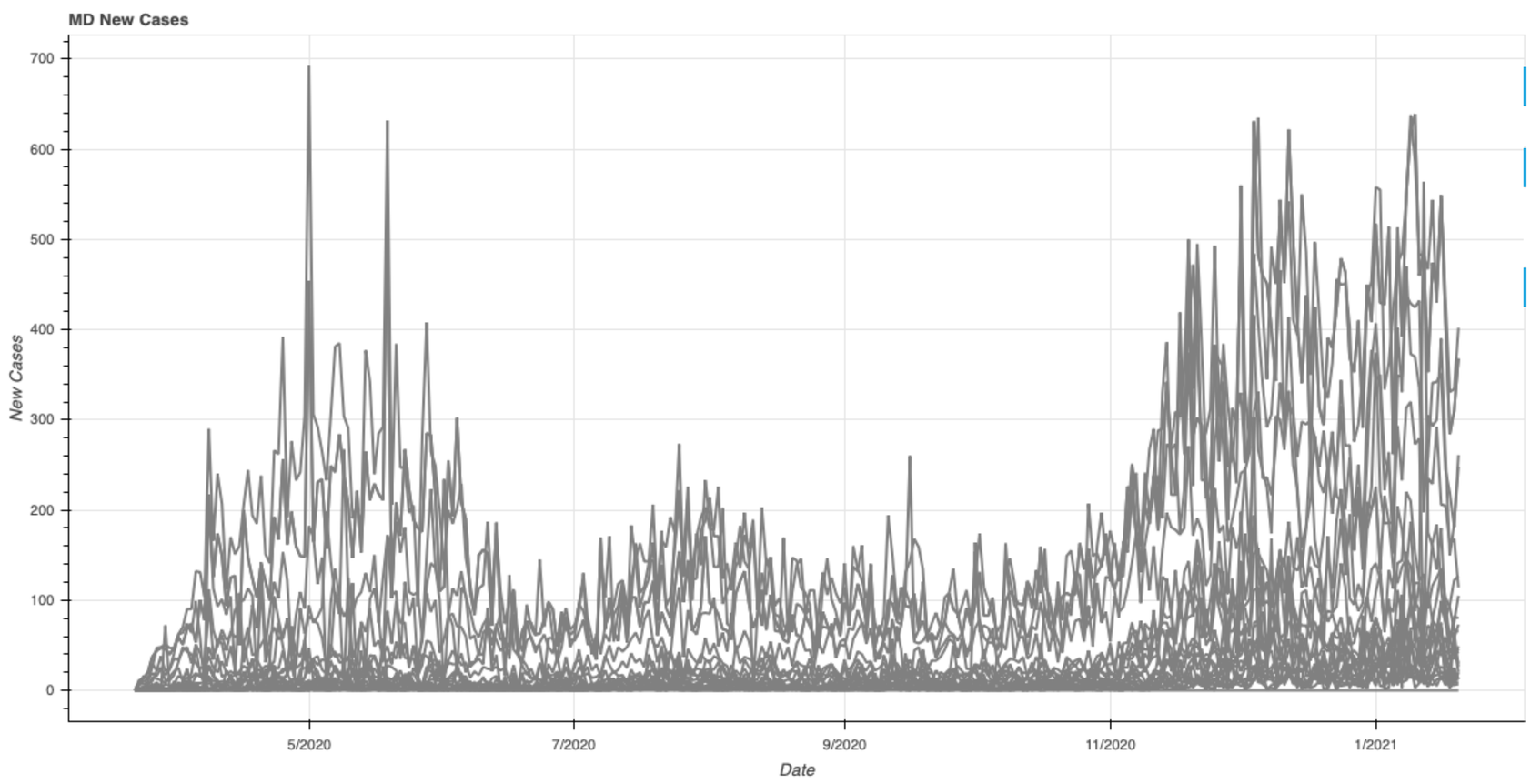
```
# now we will create a DataFrame with the incident rate for each county in Maryland

md_incident_df = pd.DataFrame(md_granular['Data_Date'].unique(), columns = ['Data_Date'])
counties = md_granular['Admin2'].unique()

for county in counties:
    tmp = md_granular[md_granular["Admin2"]==county][['Data_Date','Incidence_Rate']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "Incidence_Rate": county})
    md_incident_df = pd.merge(md_incident_df,tmp, on="Data_Date", how = 'left')
```

In [33]:
```
source = ColumnDataSource(md_incident_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="MD Incident Rate", x_axis_type="datetime")
renderer = p.line('Data_Date','Montgomery', source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("Incident Rate", "$y{0,0}"), ("County", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for county in counties:

    renderer = p.line('Data_Date', county, source = source, line_color = 'gray', line_width=2, name = county)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Incident Rate (cases per 100,000 population)'

show(p)
```



In [34]:
```
# Let's look at new cases for each county in Maryland

md_new_cases_df = pd.DataFrame(md_granular.sort_values(['Data_Date'])['Data_Date'].unique(), columns = ['Data_Date']) # needed to sort by date first because for some reason, 3-31-20 and 4-1-20 were placed after
counties = md_granular['Admin2'].unique()

for county in counties:
    tmp = md_granular[md_granular["Admin2"]==county][['Data_Date','New_Cases']].reset_index().drop(['index'], axis=1)
    tmp = tmp.rename(columns={"Data_Date": "Data_Date", "New_Cases": county})
    md_new_cases_df = pd.merge(md_new_cases_df,tmp, on="Data_Date", how = 'left')
```

In [35]:
```
source = ColumnDataSource(md_new_cases_df)

p = figure(plot_width=1200, plot_height=600, tools="tap, wheel_zoom, save, pan, reset", title="MD New Cases", x_axis_type="datetime")
renderer = p.line('Data_Date','Montgomery', source = source, line_color = 'gray', line_width=2)

hover = HoverTool(tooltips=[('date', '@Data_Date{%F}'), ("New Cases", "$y{0,0}"), ("County", "$name")],
          formatters={'@Data_Date': 'datetime'})
hover.point_policy='snap_to_data'
hover.line_policy = "nearest"


for county in counties:

    renderer = p.line('Data_Date', county, source = source, line_color = 'gray', line_width=2, name = county)

    selected_line = Line(line_color='red', line_width=4)
    nonselected_line = Line(line_color='gray')

    renderer.selection_glyph = selected_line
    renderer.nonselection_glyph = nonselected_line

p.add_tools(hover)
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'New Cases'

show(p)
```



In [36]:
```
'''
Now let's generate some features that may be useful for prediction and plot them against one another.
Features such as:
* Population
* Lagged cases (e.g. new cases 7-days prior)
* New cases 14-days forward
* Change in new cases
* Rolling average new cases
* Change in rolling averages
* Slope of Incidence_Rate

The focus will be on Maryland to keep the complexity down.
'''
```

Out[36]:
```
"\nNow let's generate some features that may be useful for prediction and plot them against one another.\nFeatures such as:\n* Population\n* Lagged cases (e.g. new cases 7-days prior)\n* New cases 14-days forwa
rd\n* Change in new cases\n* Rolling average new cases\n* Change in rolling averages\n* Slope of Incidence_Rate\n\nThe focus will be on Maryland to keep the complexity down.\n"
```

In [37]:
```
# Population may be a useful feature, as more heavily populated regions may see more cases.
# We can back into the population for each county.
md_imputed_population = md_granular[md_granular['Data_Date'] == '2020-05-29']
md_imputed_population['Population'] = md_imputed_population['Confirmed']//(md_imputed_population['Incidence_Rate']/100000)

# add population to the md dataset via a left join.
cols = ['Admin2', 'Province_State', 'Population']
tmp = md_imputed_population[cols]
md_granular = pd.merge(md_granular, tmp, on = ['Admin2', 'Province_State'], how = 'left')
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

In [38]:
```
# let's add seven and fourteen day lagged cases as historical cases could be used to predict new ones.

md_granular['Lag_7'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(7).fillna(0)
md_granular['Lag_14'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(14).fillna(0)
```

In [39]:
```
# now let's add new cases fourteen days ahead. this may become the target variable we want to predict.

md_granular['Cases_14_Days_Forward'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(-14).fillna(0)
```

In [40]:
```
# first, second and third order changes in new cases
md_granular['Daily_Change_In_New_Cases'] = md_granular['New_Cases'].diff(periods = 1).fillna(0)
md_granular['Daily_Acceleration_In_New_Cases'] = md_granular['Daily_Change_In_New_Cases'].diff(periods = 1).fillna(0)
md_granular['Daily_Aberrancy_In_New_Cases'] = md_granular['Daily_Acceleration_In_New_Cases'].diff(periods = 1).fillna(0)
```

In [41]:
```
# rolling average new cases

md_granular['Seven_Day_Rolling_Average_New_Cases']=md_granular.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(7).mean())
md_granular['Fourteen_Day_Rolling_Average_New_Cases']=md_granular.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(14).mean())
```

In [42]:
```
# change in incidence rate
md_granular['Incidence_Rate_Slope'] = md_granular['Incidence_Rate'].diff(periods = 1).fillna(0)
```

In [43]:
```
# lets plot several of these variables against one another, but we need to drop multiple columns to make this plot meaningful and feasible (E.g. no sense in plotting Data_Block vs FIPS)

md_granular_plot = md_granular.drop(['FIPS', 'Admin2', 'Population', 'Province_State', 'Country_Region', 'Last_Update', 'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active', 'Combined_Key', 'Case_Fatali

sns.pairplot(md_granular_plot, size = 2.5)
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/seaborn/axisgrid.py:1969: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[43]:
```
<seaborn.axisgrid.PairGrid at 0x7f9661e50a20>
```

XGBoost predictive models to review predictive utility of various features

Baseline model

In [44]:
```
'''
This will be a very basic model, using new cases to predict cases 14-days ahead.

The data relied upon is based on JHU's update on February 4, containing Feb 3 COVID data. We want to predict new cases for Feb 3 two weeks prior.
In other words, we want to make a prediction for February 3 given what we know as of January 20.

Training Set: starting Date_Block will depend on our features. If we use 14 day lagged cases (e.g. given where we are today, what were the case counts 14 days ago),
              the starting Date_Block will be 14.
Validation Set: the validaton set will be making a prediction on January 5 for January 19 (January 19 being reported on January 20).
                If we include days after that point, there will be a data leakage, as the model will precisely know the future.
Test Set: Test set will make a prediction for February 3 (reported on Feb 4) based on data available on January 20.

'''
```

Out[44]:
```
"\nThis will be a very basic model, using new cases to predict cases 14-days ahead.\n\nThe data relied upon is based on JHU's update on February 4, containing Feb 3 COVID data. We want to predict new cases for
Feb 3 two weeks prior. \nIn other words, we want to make a prediction for February 3 given what we know as of January 20.\n\nTraining Set: starting Date_Block will depend on our features. If we use 14 day lagge
d cases (e.g. given where we are today, what were the case counts 14 days ago), \n              the starting Date_Block will be 14. \nValidation Set: the validaton set will be making a prediction on January 5 f
or January 19 (January 19 being reported on January 20). \n                If we include days after that point, there will be a data leakage, as the model will precisely know the future.\nTest Set: Test set wil
l make a prediction for February 3 (reported on Feb 4) based on data available on January 20.\n\n"
```

In [45]:
```
# first let's re-load the raw data and rebuild the Maryland dataset.
# the folder containing the daily data had file-dates 1-21-21 to 2-4-21 removed to ensure that none of this data leaks into the training/validation set.
granular_covid_data = pd.read_csv("COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/10-03-2020.csv")
granular_covid_data.head()

path = "COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/*.csv"

for file in glob.glob(path):
    tmp = pd.read_csv(file)

    try:
        tmp['Province_State']
    except KeyError:
        tmp = tmp.rename(columns={'Province/State': 'Province_State', 'Country/Region': 'Country_Region', 'Last Update': 'Last_Update'})
        tmp['Admin2'] = np.NaN
        tmp['FIPS'] = np.NaN
        tmp = tmp.astype({'Admin2': 'str', 'FIPS': 'float64'})

    try:
        tmp['Lat']
    except KeyError:
        tmp = tmp.rename(columns={'Latitude': 'Lat', 'Longitude': 'Long_'})

    try:
        tmp['Incidence_Rate']
    except KeyError:
        tmp = tmp.rename(columns={'Incident_Rate': 'Incidence_Rate'})

    try:
        tmp['Case-Fatality_Ratio']
    except KeyError:
        tmp = tmp.rename(columns={'Case_Fatality_Ratio': 'Case-Fatality_Ratio'})

    file_date = file[-14:-4] #this extracts the date from the filename
    tmp['Data_Date'] = file_date # this adds the filename date as a new column

    tmp['Data_Date'] = pd.to_datetime(tmp['Data_Date'])

    granular_covid_data = pd.concat([granular_covid_data,tmp])

granular_covid_data = granular_covid_data.sort_values(by = ['Country_Region', 'Province_State', 'Admin2', 'Data_Date']).reset_index().drop(['index'], axis = 1)

granular_covid_data['Start'] = '2020-03-22'
granular_covid_data['Date_Block'] = (pd.to_datetime(granular_covid_data['Data_Date']) - pd.to_datetime(granular_covid_data['Start']))//np.timedelta64(1,'D')

granular_covid_data['New_Cases'] = granular_covid_data['Confirmed'].diff(periods = 1).fillna(0).clip(lower = 0)

md_granular = granular_covid_data[(granular_covid_data['Country_Region'] == 'US') & (granular_covid_data['Province_State'] == 'Maryland')]

md_granular = md_granular.dropna(subset=['Date_Block','FIPS']).reset_index().drop(['index'], axis=1)

md_granular.head()
```

Out[45]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_Rate	Case-Fatality_Ratio	Data_Date	Start	Date_Block	New_Cases
0	24001.0	Allegany	Maryland	US	3/22/20 23:45	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-22	2020-03-22	0.0	0.0
1	24001.0	Allegany	Maryland	US	2020-03-23 23:19:34	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-23	2020-03-22	1.0	0.0
2	24001.0	Allegany	Maryland	US	2020-03-24 23:37:31	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-24	2020-03-22	2.0	0.0
3	24001.0	Allegany	Maryland	US	2020-03-25 23:33:19	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-25	2020-03-22	3.0	0.0
4	24001.0	Allegany	Maryland	US	2020-03-26 23:48:35	39.623576	-78.692805	0.0	0.0	0.0	0.0	Allegany, Maryland, US	NaN	NaN	2020-03-26	2020-03-22	4.0	0.0

In [46]:
```
# generate our target variable
md_granular['Cases_14_Days_Forward'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(-14).fillna(0)
```

In [47]:
```
md_granular[['FIPS', 'Date_Block', 'Data_Date', 'New_Cases', 'Cases_14_Days_Forward']].tail(20)
```

Out[47]:

	FIPS	Date_Block	Data_Date	New_Cases	Cases_14_Days_Forward
7581	24047.0	285.0	2021-01-01	64.0	44.0
7582	24047.0	286.0	2021-01-02	27.0	39.0
7583	24047.0	287.0	2021-01-03	20.0	41.0
7584	24047.0	288.0	2021-01-04	47.0	25.0
7585	24047.0	289.0	2021-01-05	15.0	9.0
7586	24047.0	290.0	2021-01-06	31.0	20.0
7587	24047.0	291.0	2021-01-07	40.0	0.0
7588	24047.0	292.0	2021-01-08	61.0	0.0
7589	24047.0	293.0	2021-01-09	53.0	0.0
7590	24047.0	294.0	2021-01-10	50.0	0.0
7591	24047.0	295.0	2021-01-11	29.0	0.0
7592	24047.0	296.0	2021-01-12	19.0	0.0
7593	24047.0	297.0	2021-01-13	28.0	0.0
7594	24047.0	298.0	2021-01-14	31.0	0.0
7595	24047.0	299.0	2021-01-15	44.0	0.0
7596	24047.0	300.0	2021-01-16	39.0	0.0
7597	24047.0	301.0	2021-01-17	41.0	0.0
7598	24047.0	302.0	2021-01-18	25.0	0.0
7599	24047.0	303.0	2021-01-19	9.0	0.0
7600	24047.0	304.0	2021-01-20	20.0	0.0

In [48]:
```
md_granular.columns
```

Out[48]:
```
Index(['FIPS', 'Admin2', 'Province_State', 'Country_Region', 'Last_Update',
       'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio', 'Data_Date',
       'Start', 'Date_Block', 'New_Cases', 'Cases_14_Days_Forward'],
      dtype='object')
```

In [49]:
```
# retain only those columns necessary for the baseline model.
baseline_data = md_granular.drop(['Admin2', 'Province_State', 'Country_Region', 'Last_Update', 'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active', 'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Rati
```

In [50]:
```
baseline_data.columns
```

Out[50]:
```
Index(['FIPS', 'Date_Block', 'New_Cases', 'Cases_14_Days_Forward'], dtype='object')
```

In [51]:
```
baseline_data = baseline_data.sort_values(by = ['FIPS', 'Date_Block'])
```

In [52]:
```
# FIPS 90024 is associated with "Unassigned". Let's keep it for now as the predicted "New_Cases" for this FIPS should be close to zero.
```

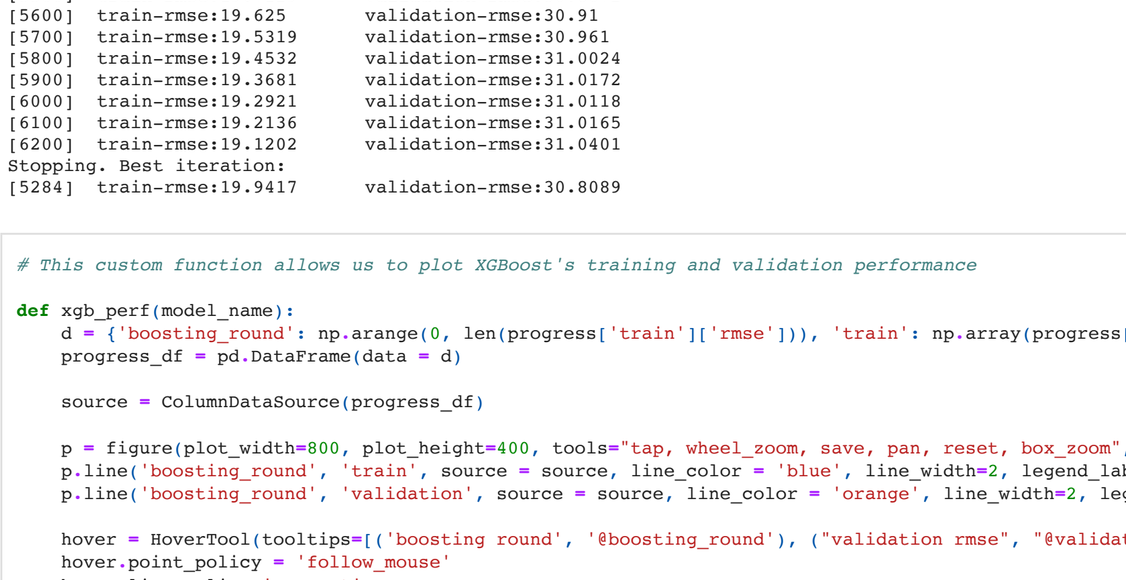
In [53]:
```
'''
Our validation set is Date_Block 290, as this makes a prediction for Date_Block 304 (January 19 cases reported on January 20).
'''

X_train = baseline_data[(baseline_data['Date_Block'] > 14) & (baseline_data['Date_Block'] <= 289)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = baseline_data[(baseline_data['Date_Block'] > 14) & (baseline_data['Date_Block'] <= 289)]['Cases_14_Days_Forward']

X_validation = baseline_data[(baseline_data['Date_Block'] > 289) & (baseline_data['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = baseline_data[(baseline_data['Date_Block'] > 289) & (baseline_data['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [54]:
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

baseline_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 1000000,  watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=1000, evals_result = progress)
```

```
[0]     train-rmse:95.3432      validation-rmse:139.907
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 1000 rounds.
[100]   train-rmse:87.4028      validation-rmse:124.341
[200]   train-rmse:80.2874      validation-rmse:111.428
[300]   train-rmse:73.9114      validation-rmse:100.423
[400]   train-rmse:68.227       validation-rmse:88.6994
[500]   train-rmse:63.1518      validation-rmse:80.9487
[600]   train-rmse:58.6326      validation-rmse:75.5135
[700]   train-rmse:54.5902      validation-rmse:70.5985
[800]   train-rmse:50.9915      validation-rmse:66.0283
[900]   train-rmse:47.7889      validation-rmse:61.7394
[1000]  train-rmse:44.9302      validation-rmse:58.049
[1100]  train-rmse:42.3882      validation-rmse:53.8082
[1200]  train-rmse:40.1435      validation-rmse:50.2308
[1300]  train-rmse:38.1565      validation-rmse:47.152
[1400]  train-rmse:36.4146      validation-rmse:45.073
[1500]  train-rmse:34.8851      validation-rmse:43.6463
[1600]  train-rmse:33.5379      validation-rmse:42.2479
[1700]  train-rmse:32.337       validation-rmse:40.7409
[1800]  train-rmse:31.2682      validation-rmse:39.2373
[1900]  train-rmse:30.3152      validation-rmse:38.2233
[2000]  train-rmse:29.4824      validation-rmse:37.4061
[2100]  train-rmse:28.7422      validation-rmse:36.7466
[2200]  train-rmse:28.053       validation-rmse:36.2345
[2300]  train-rmse:27.3963      validation-rmse:35.6601
[2400]  train-rmse:26.8035      validation-rmse:35.1721
[2500]  train-rmse:26.2611      validation-rmse:34.8061
[2600]  train-rmse:25.7557      validation-rmse:34.5797
[2700]  train-rmse:25.3102      validation-rmse:34.046
[2800]  train-rmse:24.8969      validation-rmse:33.6906
[2900]  train-rmse:24.5042      validation-rmse:33.172
[3000]  train-rmse:24.1432      validation-rmse:32.7806
[3100]  train-rmse:23.8445      validation-rmse:32.5356
[3200]  train-rmse:23.5371      validation-rmse:32.2458
[3300]  train-rmse:23.2558      validation-rmse:32.5345
[3400]  train-rmse:23.0131      validation-rmse:32.5625
[3500]  train-rmse:22.776       validation-rmse:32.4905
[3600]  train-rmse:22.5476      validation-rmse:32.3081
[3700]  train-rmse:22.3407      validation-rmse:32.1149
[3800]  train-rmse:22.1322      validation-rmse:31.9485
[3900]  train-rmse:21.9338      validation-rmse:31.7563
[4000]  train-rmse:21.7346      validation-rmse:31.5586
[4100]  train-rmse:21.5485      validation-rmse:31.3727
[4200]  train-rmse:21.3875      validation-rmse:31.3116
[4300]  train-rmse:21.2283      validation-rmse:31.2658
[4400]  train-rmse:21.0626      validation-rmse:31.1312
[4500]  train-rmse:20.876       validation-rmse:30.9833
[4600]  train-rmse:20.7112      validation-rmse:30.8937
[4700]  train-rmse:20.5371      validation-rmse:30.8784
[4800]  train-rmse:20.4024      validation-rmse:30.8838
[4900]  train-rmse:20.2997      validation-rmse:30.9595
[5000]  train-rmse:20.2136      validation-rmse:31.0011
[5100]  train-rmse:20.1192      validation-rmse:30.954
[5200]  train-rmse:20.0168      validation-rmse:30.8674
[5300]  train-rmse:19.9262      validation-rmse:30.8133
[5400]  train-rmse:19.8274      validation-rmse:30.8287
[5500]  train-rmse:19.7129      validation-rmse:30.8672
[5600]  train-rmse:19.625       validation-rmse:30.91
[5700]  train-rmse:19.5319      validation-rmse:30.961
[5800]  train-rmse:19.4532      validation-rmse:31.0024
[5900]  train-rmse:19.3681      validation-rmse:31.0172
[6000]  train-rmse:19.2921      validation-rmse:31.0118
[6100]  train-rmse:19.2136      validation-rmse:31.0165
[6200]  train-rmse:19.1202      validation-rmse:31.0401
Stopping. Best iteration:
[5284]  train-rmse:19.9417      validation-rmse:30.8089
```

In [55]:
```
# This custom function allows us to plot XGBoost's training and validation performance

def xgb_perf(model_name):
    d = {'boosting_round': np.arange(0, len(progress['train']['rmse'])), 'train': np.array(progress['train']['rmse']), 'validation': np.array(progress['validation']['rmse'])}
    progress_df = pd.DataFrame(data = d)

    source = ColumnDataSource(progress_df)

    p = figure(plot_width=800, plot_height=400, tools="tap, wheel_zoom, save, pan, reset, box_zoom", title="XGBoost Performance - " + model_name)
    p.line('boosting_round', 'train', source = source, line_color = 'blue', line_width=2, legend_label = 'train')
    p.line('boosting_round', 'validation', source = source, line_color = 'orange', line_width=2, legend_label = 'validation')

    hover = HoverTool(tooltips=[('boosting round', '@boosting_round'), ("validation rmse", "@validation{0,0.00}"), ("train rmse", "@train{0,0.00}")], mode='mouse')
    hover.point_policy = 'follow_mouse'
    hover.line_policy='nearest'

    cross = CrosshairTool(dimensions = 'height')

    p.add_tools(hover)
    p.add_tools(cross)

    p.xaxis.axis_label = 'Boosting Round'
    p.yaxis.axis_label = 'RMSE'
    p.legend.location = 'top_right'

    show(p)
```

In [56]:
```
xgb_perf("Baseline")
```



In [57]:
```
X_test = baseline_data[(baseline_data['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = baseline_data[(baseline_data['Date_Block'] == 304)]['Cases_14_Days_Forward']
```

In [58]:
```
prediction = baseline_model.predict(xgb.DMatrix(X_test), ntree_limit=baseline_model.best_ntree_limit)
```

In [59]:
```
# import the test set

test_set = pd.read_csv("COVID-19-master/csse_covid_19_data/csse_covid_19_daily_reports/10-03-2020.csv")
cols = ['FIPS', 'Admin2','Province_State','Country_Region']
test_set = test_set[cols]

# Test_Set data stored outside the training data. This folder contains 2-3-21 and 2-4-21 dated files.
path = "COVID-19-master/csse_covid_19_data/test_set/*.csv"

for file in glob.glob(path):
    tmp = pd.read_csv(file)

    try:
        tmp['Province_State']
    except KeyError:
        tmp = tmp.rename(columns={'Province/State': 'Province_State', 'Country/Region': 'Country_Region', 'Last Update': 'Last_Update'})
        tmp['Admin2'] = np.NaN
        tmp['FIPS'] = np.NaN
        tmp = tmp.astype({'Admin2': 'str', 'FIPS': 'float64'})

    try:
        tmp['Lat']
    except KeyError:
        tmp = tmp.rename(columns={'Latitude': 'Lat', 'Longitude': 'Long_'})

    try:
        tmp['Incidence_Rate']
    except KeyError:
        tmp = tmp.rename(columns={'Incident_Rate': 'Incidence_Rate'})

    try:
        tmp['Case-Fatality_Ratio']
    except KeyError:
        tmp = tmp.rename(columns={'Case_Fatality_Ratio': 'Case-Fatality_Ratio'})

    file_date = file[-14:-4] #this extracts the date from the filename
    tmp['Data_Date'] = file_date # this adds the filename date as a new column

    tmp['Data_Date'] = pd.to_datetime(tmp['Data_Date'])
    test_set = pd.concat([test_set,tmp])

test_set = test_set.sort_values(by = ['Country_Region', 'Province_State', 'Admin2', 'Data_Date']).reset_index().drop(['index'], axis = 1)

md_test = test_set[(test_set['Country_Region'] == 'US') & (test_set['Province_State'] == 'Maryland')].sort_values(by = ['Country_Region', 'Province_State', 'Admin2', 'Data_Date'])

md_test['New_Cases'] = md_test['Confirmed'].diff(periods = 1).fillna(0).clip(lower = 0) # this will calculate new cases by comparing each row (date) to the prior row (prior day)
md_test = md_test.dropna(subset=['Data_Date','FIPS']).reset_index().drop(['index'], axis=1)
md_test['Start'] = '2020-03-22'
md_test['Date_Block'] = (pd.to_datetime(md_test['Data_Date']) - pd.to_datetime(md_test['Start']))//np.timedelta64(1,'D')
```

In [60]:
```
md_test.head()
```

Out[60]:

	FIPS	Admin2	Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	Combined_Key	Incidence_Rate	Case-Fatality_Ratio	Data_Date	New_Cases	Start	Date_Block
0	24001.0	Allegany	Maryland	US	2021-02-03 05:22:52	39.623576	-78.692805	6212.0	186.0	0.0	6026.0	Allegany, Maryland, US	8821.858668	2.994205	2021-02-02	0.0	2020-03-22	317
1	24001.0	Allegany	Maryland	US	2021-02-04 05:22:26	39.623576	-78.692805	6218.0	189.0	0.0	6029.0	Allegany, Maryland, US	8830.379459	3.039563	2021-02-03	6.0	2020-03-22	318
2	24003.0	Anne Arundel	Maryland	US	2021-02-03 05:22:52	39.006702	-76.603293	33111.0	474.0	0.0	32637.0	Anne Arundel, Maryland, US	5716.342618	1.431548	2021-02-02	0.0	2020-03-22	317
3	24003.0	Anne Arundel	Maryland	US	2021-02-04 05:22:26	39.006702	-76.603293	33194.0	481.0	0.0	32713.0	Anne Arundel, Maryland, US	5730.671887	1.449057	2021-02-03	83.0	2020-03-22	318
4	24005.0	Baltimore	Maryland	US	2021-02-03 05:22:52	39.457847	-76.629120	47608.0	1153.0	0.0	46455.0	Baltimore, Maryland, US	5754.136602	2.421862	2021-02-02	0.0	2020-03-22	317

In [61]:
```
md_test = md_test.sort_values(by = ['FIPS', 'Date_Block'])
md_test = md_test[md_test['Date_Block'] == 318]
```

In [62]:
```
cols = ['Admin2', 'FIPS', 'New_Cases']
results = md_test[cols]
results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[62]:

	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	28.611877
3	Anne Arundel	24003.0	83.0	237.756607
5	Baltimore	24005.0	127.0	222.895401
9	Calvert	24009.0	8.0	32.490196
11	Caroline	24011.0	2.0	26.225483
13	Carroll	24013.0	16.0	42.928600
15	Cecil	24015.0	8.0	49.343437
17	Charles	24017.0	39.0	51.150768
19	Dorchester	24019.0	7.0	17.873108
21	Frederick	24021.0	37.0	124.111206
23	Garrett	24023.0	2.0	13.364948
25	Harford	24025.0	31.0	81.991776
27	Howard	24027.0	48.0	82.390221
29	Kent	24029.0	3.0	11.843496
31	Montgomery	24031.0	147.0	333.911804
33	Prince George's	24033.0	195.0	349.376801
35	Queen Anne's	24035.0	8.0	17.290003
39	St. Mary's	24037.0	18.0	56.124935
37	Somerset	24039.0	7.0	15.036382
41	Talbot	24041.0	9.0	23.734104
45	Washington	24043.0	27.0	95.518532
47	Wicomico	24045.0	32.0	42.668835
49	Worcester	24047.0	8.0	25.599157
7	Baltimore City	24510.0	74.0	94.475800
43	Unassigned	90024.0	0.0	-0.457787

In [63]:
```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[63]:
```
67.8969436395403
```

In [64]:
```
'''
The results aren't particularly good, which is expected. This model seems to overestimate future cases.
'''
```

Out[64]:
```
"\nThe results aren't particularly good, which is expected. This model seems to overestimate future cases.\n"
```

Baseline with lagged cases

In [65]:
```
md_granular['Lag_1'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(1).fillna(0)
md_granular['Lag_2'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(2).fillna(0)
md_granular['Lag_3'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(3).fillna(0)
md_granular['Lag_4'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(4).fillna(0)
md_granular['Lag_5'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(5).fillna(0)
md_granular['Lag_6'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(6).fillna(0)
md_granular['Lag_7'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(7).fillna(0)
md_granular['Lag_8'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(8).fillna(0)
md_granular['Lag_9'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(9).fillna(0)
md_granular['Lag_10'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(10).fillna(0)
md_granular['Lag_11'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(11).fillna(0)
md_granular['Lag_12'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(12).fillna(0)
md_granular['Lag_13'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(13).fillna(0)
md_granular['Lag_14'] = md_granular.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])['New_Cases'].shift(14).fillna(0)
```

In [66]:
```
md_lagged = md_granular.drop(['Admin2', 'Province_State', 'Country_Region', 'Last_Update', 'Lat', 'Long_', 'Confirmed', 'Deaths', 'Recovered', 'Active', 'Combined_Key', 'Incidence_Rate', 'Case-Fatality_Ratio',
```

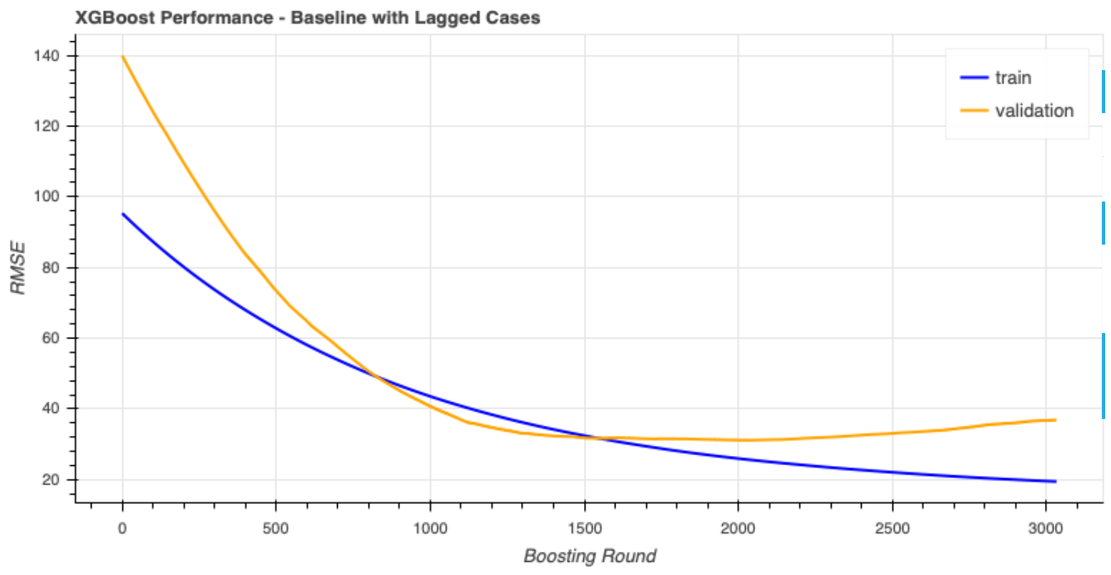
In [67]:
```
md_lagged = md_lagged.sort_values(by = ['FIPS', 'Date_Block'])
```

In [68]:
```
X_train = md_lagged[(md_lagged['Date_Block'] > 14) & (md_lagged['Date_Block'] <= 289)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = md_lagged[(md_lagged['Date_Block'] > 14) & (md_lagged['Date_Block'] <= 289)]['Cases_14_Days_Forward']

X_validation = md_lagged[(md_lagged['Date_Block'] > 289) & (md_lagged['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = md_lagged[(md_lagged['Date_Block'] > 289) & (md_lagged['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [69]:
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

lagged_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 1000000,  watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=1000, evals_result = progress)
```

```
[0]     train-rmse:95.3424      validation-rmse:139.905
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 1000 rounds.
[100]   train-rmse:87.3202      validation-rmse:124.065
[200]   train-rmse:80.1425      validation-rmse:109.57
[300]   train-rmse:73.7201      validation-rmse:95.9834
[400]   train-rmse:67.9724      validation-rmse:83.7951
[500]   train-rmse:62.7828      validation-rmse:73.4225
[600]   train-rmse:58.0829      validation-rmse:64.7891
[700]   train-rmse:53.8661      validation-rmse:57.5509
[800]   train-rmse:50.0427      validation-rmse:50.6391
[900]   train-rmse:46.6092      validation-rmse:45.2081
[1000]  train-rmse:43.5469      validation-rmse:40.7531
[1100]  train-rmse:40.8265      validation-rmse:36.9376
[1200]  train-rmse:38.3681      validation-rmse:34.7301
[1300]  train-rmse:36.1662      validation-rmse:33.1156
[1400]  train-rmse:34.1918      validation-rmse:32.3723
[1500]  train-rmse:32.4445      validation-rmse:31.7971
[1600]  train-rmse:30.8654      validation-rmse:31.8006
[1700]  train-rmse:29.4443      validation-rmse:31.559
[1800]  train-rmse:28.1406      validation-rmse:31.5438
[1900]  train-rmse:26.9911      validation-rmse:31.3663
[2000]  train-rmse:25.9618      validation-rmse:31.1843
[2100]  train-rmse:25.0385      validation-rmse:31.2737
[2200]  train-rmse:24.2055      validation-rmse:31.6057
[2300]  train-rmse:23.4233      validation-rmse:32.0422
[2400]  train-rmse:22.7238      validation-rmse:32.5883
[2500]  train-rmse:22.0749      validation-rmse:33.0817
[2600]  train-rmse:21.4741      validation-rmse:33.6189
[2700]  train-rmse:20.9247      validation-rmse:34.3457
[2800]  train-rmse:20.4299      validation-rmse:35.421
[2900]  train-rmse:19.9966      validation-rmse:36.0574
[3000]  train-rmse:19.5785      validation-rmse:36.7113
Stopping. Best iteration:
[2033]  train-rmse:25.6463      validation-rmse:31.1368
```

In [70]:
```
xgb_perf("Baseline with Lagged Cases")
```



In [71]:
```
X_test = md_lagged[(md_lagged['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = md_lagged[(md_lagged['Date_Block'] == 304)]['Cases_14_Days_Forward']
```

In [72]:
```
prediction = lagged_model.predict(xgb.DMatrix(X_test), ntree_limit=lagged_model.best_ntree_limit)
```

In [73]:
```
results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[73]:

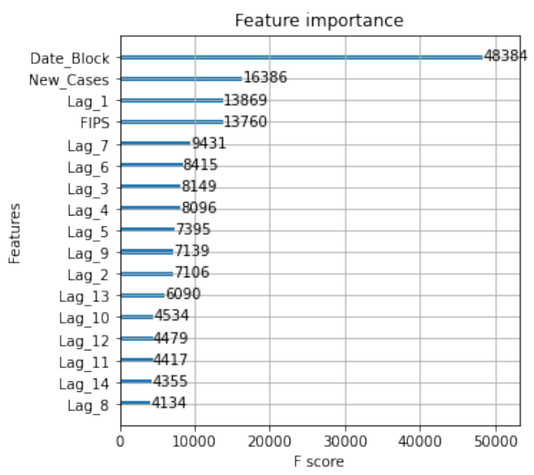
	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	38.367638
3	Anne Arundel	24003.0	83.0	237.198990
5	Baltimore	24005.0	127.0	246.622665
9	Calvert	24009.0	8.0	37.687435
11	Caroline	24011.0	2.0	19.882269
13	Carroll	24013.0	16.0	40.072723
15	Cecil	24015.0	8.0	47.607952
17	Charles	24017.0	39.0	55.690777
19	Dorchester	24019.0	7.0	20.531078
21	Frederick	24021.0	37.0	147.609680
23	Garrett	24023.0	2.0	17.330803
25	Harford	24025.0	31.0	77.217751
27	Howard	24027.0	48.0	70.164688
29	Kent	24029.0	3.0	12.456251
31	Montgomery	24031.0	147.0	347.114319
33	Prince George's	24033.0	195.0	347.146210
35	Queen Anne's	24035.0	8.0	20.078966
39	St. Mary's	24037.0	18.0	52.101139
37	Somerset	24039.0	7.0	16.785507
41	Talbot	24041.0	9.0	20.813562
45	Washington	24043.0	27.0	86.008492
47	Wicomico	24045.0	32.0	49.216488
49	Worcester	24047.0	8.0	23.483620
7	Baltimore City	24510.0	74.0	167.732742
43	Unassigned	90024.0	0.0	1.074616

In [74]:
```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[74]: 73.66828634351073

In [75]:
```
# Plot feature importance
plt.rcParams["figure.figsize"] = (5, 5)
plot_importance(lagged_model)
```

Out[75]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>



In [76]:
```
'''
Interestingly, this model is somewhat worse than the baseline.
What's also odd is there are some training iterations where training RMSE is slightly worse than validation.
Perhaps this suggests that the validation set is too small. Backpropagation is limited to just this one validation data point.
'''
```

Out[76]: "\nInterestingly, this model is somewhat worse than the baseline.\nWhat's also odd is there are some training iterations where training RMSE is slightly worse than validation. \nPerhaps this suggests that the validation set is too small. Backpropagation is limited to just this one validation data point.\n"

Baseline plus Neighbor model

In [77]:
```
'''
Here, let's modify the baseline model to include new cases of neighboring counties.
The idea is that people travel. If Prince George's county is seeing rapidly increasing case counts, Montgomery County may experience an uptick as well.
'''
```

Out[77]: "\nHere, let's modify the baseline model to include new cases of neighboring counties.\nThe idea is that people travel. If Prince George's county is seeing rapidly increasing case counts, Montgomery County may experience an uptick as well.\n"

In [78]:
```
baseline_plus_neighbors_data = baseline_data.copy()
```

In [79]:
```
county_adjacency_list = pd.read_csv("county_adjacency2010.csv") #this file lists each county's neighbors. Source: https://data.nber.org/data/county-adjacency.html

counties = md_granular['FIPS'].unique()

for county in counties:

    neighbors = county_adjacency_list[county_adjacency_list['fipscounty'] == county]['fipsneighbor'].unique()

    for neighbor in neighbors:
        if neighbor == county:
            continue
        else:
            col = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county)

            tmp = granular_covid_data[granular_covid_data['FIPS'] == neighbor].sort_values(['FIPS','Date_Block'])[['New_Cases','FIPS','Date_Block']]
            tmp['FIPS'] = county
            tmp = tmp.rename(columns={"New_Cases": col})

            baseline_plus_neighbors_data = pd.merge(baseline_plus_neighbors_data, tmp, on = ["FIPS", 'Date_Block'], how = 'left')
```

In [80]:
```
baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] == 100) & (baseline_plus_neighbors_data['FIPS'] == 24031.0)][['FIPS', 'Date_Block', 'New_Cases', 'Neighbor_24033_to_County_24031.0']]
```

Out[80]:

	FIPS	Date_Block	New_Cases	Neighbor_24033_to_County_24031.0
4368	24031.0	100.0	62.0	67.0

In [81]:
```
baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] == 100) & (baseline_plus_neighbors_data['FIPS'] == 24033.0)][['FIPS', 'Date_Block', 'New_Cases', 'Neighbor_24031_to_County_24033.0']]
```

Out[81]:

	FIPS	Date_Block	New_Cases	Neighbor_24031_to_County_24033.0
4674	24033.0	100.0	67.0	62.0

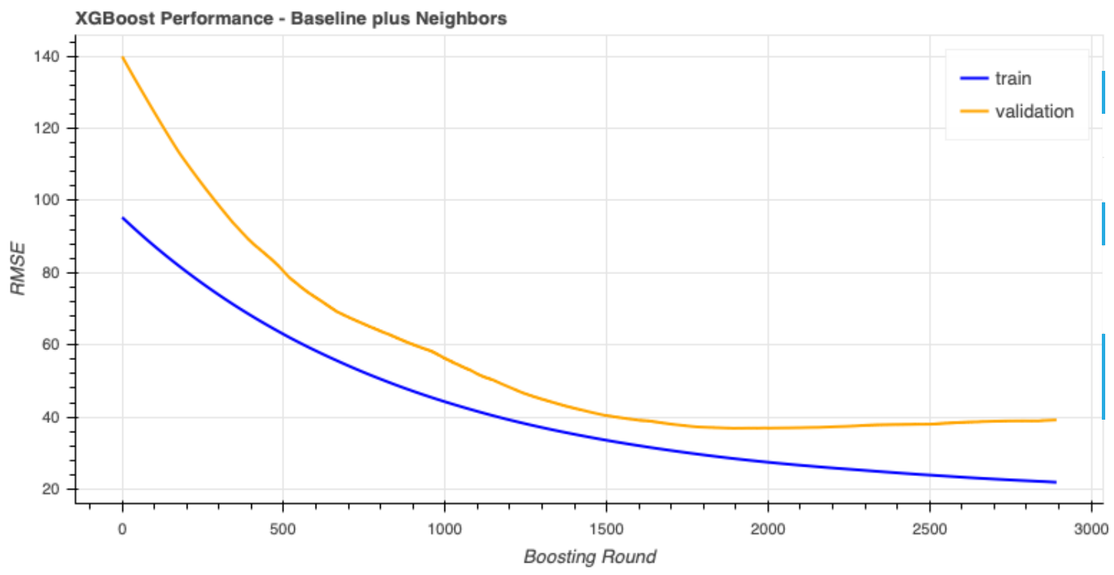
In [82]:
```
baseline_plus_neighbors_data = baseline_plus_neighbors_data.sort_values(by = ['FIPS', 'Date_Block'])
```

In [83]:
```
X_train = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] > 14) & (baseline_plus_neighbors_data['Date_Block'] <= 289)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] > 14) & (baseline_plus_neighbors_data['Date_Block'] <= 289)]['Cases_14_Days_Forward']

X_validation = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] > 289) & (baseline_plus_neighbors_data['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] > 289) & (baseline_plus_neighbors_data['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [84]:
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

baseline_plus_neighbors_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 1000000,  watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=1000, evals_result = progress)
```

```
[0]     train-rmse:95.3428      validation-rmse:139.909
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 1000 rounds.
[100]   train-rmse:87.3662      validation-rmse:124.382
[200]   train-rmse:80.2311      validation-rmse:110.298
[300]   train-rmse:73.8224      validation-rmse:98.5538
[400]   train-rmse:68.0855      validation-rmse:88.5035
[500]   train-rmse:62.9419      validation-rmse:80.3609
[600]   train-rmse:58.3545      validation-rmse:73.1102
[700]   train-rmse:54.2429      validation-rmse:67.7035
[800]   train-rmse:50.5372      validation-rmse:63.8126
[900]   train-rmse:47.2088      validation-rmse:60.1331
[1000]  train-rmse:44.2428      validation-rmse:56.2563
[1100]  train-rmse:41.5921      validation-rmse:51.9482
[1200]  train-rmse:39.2138      validation-rmse:48.2478
[1300]  train-rmse:37.1171      validation-rmse:44.9726
[1400]  train-rmse:35.2568      validation-rmse:42.47
[1500]  train-rmse:33.5812      validation-rmse:40.4291
[1600]  train-rmse:32.0907      validation-rmse:39.1452
[1700]  train-rmse:30.7433      validation-rmse:38.0612
[1800]  train-rmse:29.5381      validation-rmse:37.2176
[1900]  train-rmse:28.4498      validation-rmse:36.9206
[2000]  train-rmse:27.4972      validation-rmse:36.9637
[2100]  train-rmse:26.6665      validation-rmse:37.067
[2200]  train-rmse:25.9143      validation-rmse:37.314
[2300]  train-rmse:25.2346      validation-rmse:37.7121
[2400]  train-rmse:24.5596      validation-rmse:37.9422
[2500]  train-rmse:23.9486      validation-rmse:38.0441
[2600]  train-rmse:23.3653      validation-rmse:38.531
[2700]  train-rmse:22.8389      validation-rmse:38.863
[2800]  train-rmse:22.3773      validation-rmse:38.9496
Stopping. Best iteration:
[1894]  train-rmse:28.5118      validation-rmse:36.9086
```

In [85]:
```
xgb_perf("Baseline plus Neighbors")
```



In [86]:
```
X_test = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = baseline_plus_neighbors_data[(baseline_plus_neighbors_data['Date_Block'] == 304)]['Cases_14_Days_Forward']
```

In [87]:
```
prediction = baseline_plus_neighbors_model.predict(xgb.DMatrix(X_test), ntree_limit=baseline_plus_neighbors_model.best_ntree_limit)
```

In [88]:
```
results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[88]:

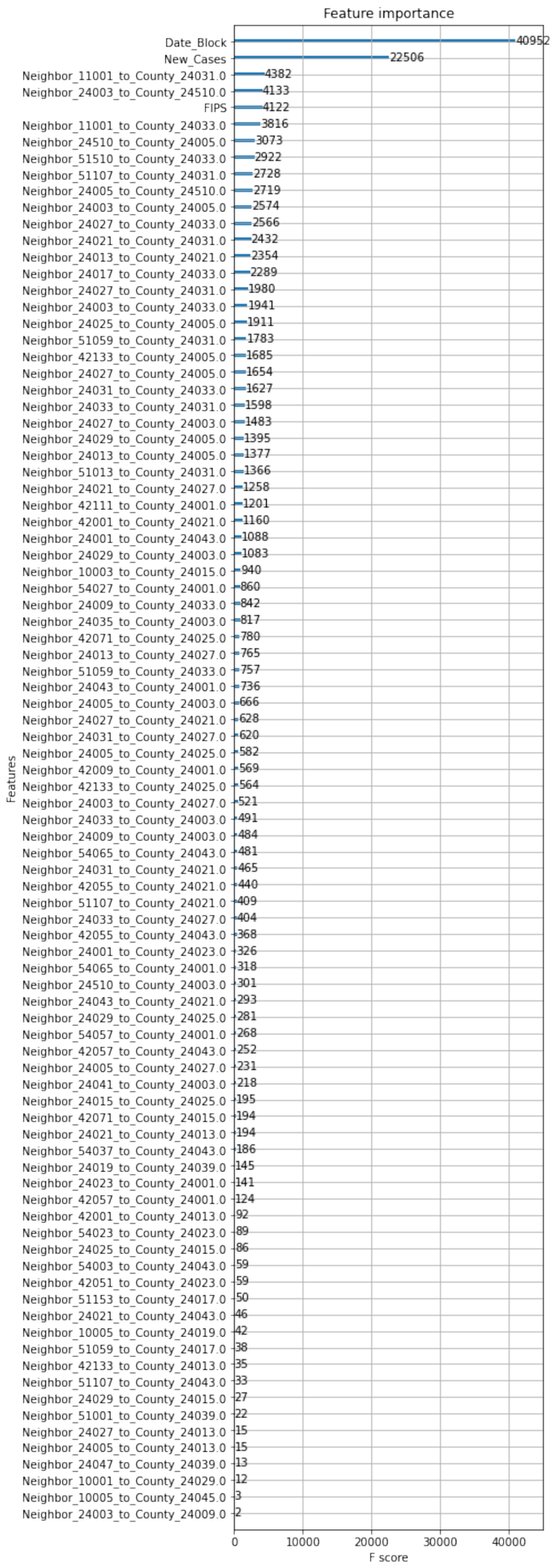
	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	33.303524
3	Anne Arundel	24003.0	83.0	225.192596
5	Baltimore	24005.0	127.0	232.640198
9	Calvert	24009.0	8.0	22.886620
11	Caroline	24011.0	2.0	22.204823
13	Carroll	24013.0	16.0	41.752094
15	Cecil	24015.0	8.0	48.395821
17	Charles	24017.0	39.0	45.815922
19	Dorchester	24019.0	7.0	15.591151
21	Frederick	24021.0	37.0	111.445366
23	Garrett	24023.0	2.0	15.070510
25	Harford	24025.0	31.0	74.275406
27	Howard	24027.0	48.0	72.604431
29	Kent	24029.0	3.0	12.595866
31	Montgomery	24031.0	147.0	273.439850
33	Prince George's	24033.0	195.0	314.937195
35	Queen Anne's	24035.0	8.0	17.467436
39	St. Mary's	24037.0	18.0	65.493271
37	Somerset	24039.0	7.0	15.591151
41	Talbot	24041.0	9.0	22.204823
45	Washington	24043.0	27.0	91.650963
47	Wicomico	24045.0	32.0	43.583927
49	Worcester	24047.0	8.0	22.852480
7	Baltimore City	24510.0	74.0	105.817451
43	Unassigned	90024.0	0.0	3.208226

In [89]:
```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[89]:
```
57.26251836179671
```

In [90]:
```
plt.rcParams["figure.figsize"] = (5, 25)
plot_importance(baseline_plus_neighbors_model)
```

Out[90]: <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>



In [91]:
```
'''
Interestingly, using neighbor data seems to work much better than the lagged model.
However, I still think that lagged features would be useful, perhaps when used with other features.
'''
```

Out[91]: '\nInterestingly, using neighbor data seems to work much better than the lagged model.\nHowever, I still think that lagged features would be useful, perhaps when used with other features.\n'

Lagged model with lagged neighbors

In [92]:
```
lagged_with_neighbors = md_lagged.copy()
```

In [93]:
```
for county in counties:

    neighbors = county_adjacency_list[county_adjacency_list['fipscounty'] == county]['fipsneighbor'].unique()

    for neighbor in neighbors:
        if neighbor == county:
            continue
        else:
            col = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county)

            tmp = granular_covid_data[granular_covid_data['FIPS'] == neighbor].sort_values(['FIPS','Date_Block'])[['New_Cases','FIPS','Date_Block']]
            tmp['FIPS'] = county
            tmp = tmp.rename(columns={"New_Cases": col})

            lagged_with_neighbors = pd.merge(lagged_with_neighbors, tmp, on = ["FIPS", 'Date_Block'], how = 'left')

            for i in range(1,15):
                lag_col = col + '_Lag_' + str(i)

                lagged_with_neighbors[lag_col] = lagged_with_neighbors.sort_values(['FIPS','Date_Block']).groupby(['FIPS'])[col].shift(i).fillna(0)
```

In [94]:
```
lagged_with_neighbors.columns
```

Out[94]:
```
Index(['FIPS', 'Date_Block', 'New_Cases', 'Cases_14_Days_Forward', 'Lag_1',
       'Lag_2', 'Lag_3', 'Lag_4', 'Lag_5', 'Lag_6',
       ...
       'Neighbor_51001_to_County_24047.0_Lag_5',
       'Neighbor_51001_to_County_24047.0_Lag_6',
       'Neighbor_51001_to_County_24047.0_Lag_7',
       'Neighbor_51001_to_County_24047.0_Lag_8',
       'Neighbor_51001_to_County_24047.0_Lag_9',
       'Neighbor_51001_to_County_24047.0_Lag_10',
       'Neighbor_51001_to_County_24047.0_Lag_11',
       'Neighbor_51001_to_County_24047.0_Lag_12',
       'Neighbor_51001_to_County_24047.0_Lag_13',
       'Neighbor_51001_to_County_24047.0_Lag_14'],
      dtype='object', length=2193)
```

In [95]:
```
lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] == 100) & (lagged_with_neighbors['FIPS'] == 24031.0)][['FIPS', 'Date_Block', 'New_Cases','Lag_6', 'Neighbor_24033_to_County_24031.0', 'Neighbor_24033_t
```

Out[95]:

	FIPS	Date_Block	New_Cases	Lag_6	Neighbor_24033_to_County_24031.0	Neighbor_24033_to_County_24031.0_Lag_6
4368	24031.0	100.0	62.0	79.0	67.0	71.0

In [96]:
```
lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] == 100) & (lagged_with_neighbors['FIPS'] == 24033.0)][['FIPS', 'Date_Block', 'New_Cases','Lag_6', 'Neighbor_24031_to_County_24033.0', 'Neighbor_24031_t
```

Out[96]:

	FIPS	Date_Block	New_Cases	Lag_6	Neighbor_24031_to_County_24033.0	Neighbor_24031_to_County_24033.0_Lag_6
4674	24033.0	100.0	67.0	71.0	62.0	79.0

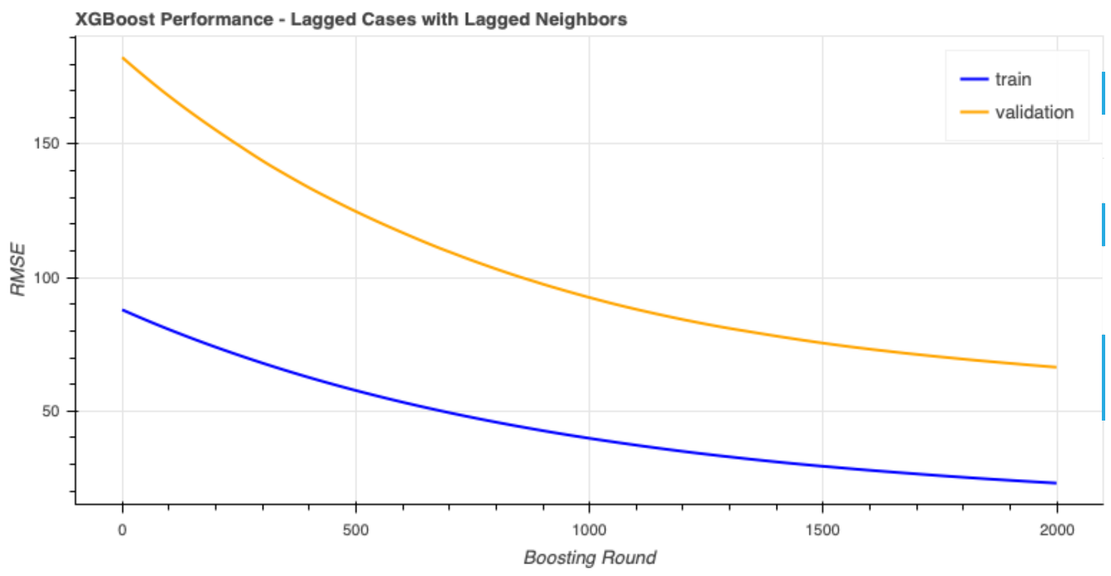
In [97]:
```
lagged_with_neighbors = lagged_with_neighbors.sort_values(by = ['FIPS', 'Date_Block'])
```

In [98]:
```
X_train = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] > 14) & (lagged_with_neighbors['Date_Block'] <= 275)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] > 14) & (lagged_with_neighbors['Date_Block'] <= 275)]['Cases_14_Days_Forward']

X_validation = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] > 275) & (lagged_with_neighbors['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] > 275) & (lagged_with_neighbors['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [99]:
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42,
        'tree_method': 'gpu_hist' #added to accelerate computations. Remove if you don't have an NVIDIA GPU
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

lagged_with_neighbors_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 2000,  watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=1000, evals_result = progress)
```

```
[0]     train-rmse:87.9589      validation-rmse:182.504
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 1000 rounds.
[100]   train-rmse:80.5811      validation-rmse:168.03
[200]   train-rmse:73.9671      validation-rmse:155.323
[300]   train-rmse:67.9907      validation-rmse:143.793
[400]   train-rmse:62.5848      validation-rmse:133.677
[500]   train-rmse:57.7135      validation-rmse:124.776
[600]   train-rmse:53.33        validation-rmse:116.818
[700]   train-rmse:49.389       validation-rmse:109.651
[800]   train-rmse:45.8459      validation-rmse:103.238
[900]   train-rmse:42.654       validation-rmse:97.5544
[1000]  train-rmse:39.7921      validation-rmse:92.5433
[1100]  train-rmse:37.23        validation-rmse:88.1179
[1200]  train-rmse:34.9008      validation-rmse:84.2782
[1300]  train-rmse:32.8418      validation-rmse:80.9635
[1400]  train-rmse:30.9692      validation-rmse:78.0838
[1500]  train-rmse:29.3166      validation-rmse:75.4541
[1600]  train-rmse:27.8179      validation-rmse:73.1944
[1700]  train-rmse:26.4617      validation-rmse:71.2262
[1800]  train-rmse:25.1938      validation-rmse:69.467
[1900]  train-rmse:24.0545      validation-rmse:67.887
[1999]  train-rmse:23.0124      validation-rmse:66.4301
```

In [100…
```
xgb_perf("Lagged Cases with Lagged Neighbors")
```



In [101…
```
X_test = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = lagged_with_neighbors[(lagged_with_neighbors['Date_Block'] == 304)]['Cases_14_Days_Forward']
```

In [102…
```
prediction = lagged_with_neighbors_model.predict(xgb.DMatrix(X_test), ntree_limit=lagged_with_neighbors_model.best_ntree_limit)
```

In [103…
```
results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[103…

	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	36.127354
3	Anne Arundel	24003.0	83.0	223.324905
5	Baltimore	24005.0	127.0	226.656815
9	Calvert	24009.0	8.0	35.189888
11	Caroline	24011.0	2.0	17.267895
13	Carroll	24013.0	16.0	37.166183
15	Cecil	24015.0	8.0	44.255589
17	Charles	24017.0	39.0	48.674133
19	Dorchester	24019.0	7.0	18.611799
21	Frederick	24021.0	37.0	151.668762
23	Garrett	24023.0	2.0	15.272314
25	Harford	24025.0	31.0	75.180305
27	Howard	24027.0	48.0	70.657242
29	Kent	24029.0	3.0	10.891318
31	Montgomery	24031.0	147.0	357.483154
33	Prince George's	24033.0	195.0	344.066437
35	Queen Anne's	24035.0	8.0	19.010708
39	St. Mary's	24037.0	18.0	55.196453
37	Somerset	24039.0	7.0	27.773626
41	Talbot	24041.0	9.0	17.099518
45	Washington	24043.0	27.0	92.893326
47	Wicomico	24045.0	32.0	50.212658
49	Worcester	24047.0	8.0	20.942490
7	Baltimore City	24510.0	74.0	142.293777
43	Unassigned	90024.0	0.0	2.138870

In [104…
```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[104… 71.38117072011069

In [105…
```
'''
Interestingly, including lagged variables -- at least in this current configuration of the model -- seems to make the predictions worse.
This model is also overfitting badly.
'''
```

Out[105… '\nInterestingly, including lagged variables -- at least in this current configuration of the model -- seems to make the predictions worse.\nThis model is also overfitting badly.\n'

Baseline with Neighbors and Rates of Change

In [106…]
```
baseline_plus_neighbors_roc_data = baseline_data.copy()
```

In [107…]
```
# add first, second and third derivatives for each county

baseline_plus_neighbors_roc_data['Daily_Change_In_New_Cases'] = baseline_plus_neighbors_roc_data['New_Cases'].diff(periods = 1).fillna(0)
baseline_plus_neighbors_roc_data['Daily_Acceleration_In_New_Cases'] = baseline_plus_neighbors_roc_data['Daily_Change_In_New_Cases'].diff(periods = 1).fillna(0)
baseline_plus_neighbors_roc_data['Daily_Aberrancy_In_New_Cases'] = baseline_plus_neighbors_roc_data['Daily_Acceleration_In_New_Cases'].diff(periods = 1).fillna(0)
```

In [108…]
```
for county in counties:

    neighbors = county_adjacency_list[county_adjacency_list['fipscounty'] == county]['fipsneighbor'].unique()

    for neighbor in neighbors:
        if neighbor == county:
            continue
        else:
            col = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county)

            tmp = granular_covid_data[granular_covid_data['FIPS'] == neighbor].sort_values(['FIPS','Date_Block'])[['New_Cases','FIPS','Date_Block']]
            tmp['FIPS'] = county
            tmp = tmp.rename(columns={"New_Cases": col})

            baseline_plus_neighbors_roc_data = pd.merge(baseline_plus_neighbors_roc_data, tmp, on = ["FIPS", 'Date_Block'], how = 'left')

            change = col + '_Daily_Change_In_New_Cases'
            acceleration = col + '_Daily_Acceleration_In_New_Cases'
            aberrancy = col + '_Daily_Aberrancy_In_New_Cases'

            baseline_plus_neighbors_roc_data[change] = baseline_plus_neighbors_roc_data[col].diff(periods = 1).fillna(0)
            baseline_plus_neighbors_roc_data[acceleration] = baseline_plus_neighbors_roc_data[change].diff(periods = 1).fillna(0)
            baseline_plus_neighbors_roc_data[aberrancy] = baseline_plus_neighbors_roc_data[acceleration].diff(periods = 1).fillna(0)
```

In [109…]
```
baseline_plus_neighbors_roc_data = baseline_plus_neighbors_roc_data.sort_values(by = ['FIPS', 'Date_Block'])
```

In [110…]
```
baseline_plus_neighbors_roc_data.columns
```

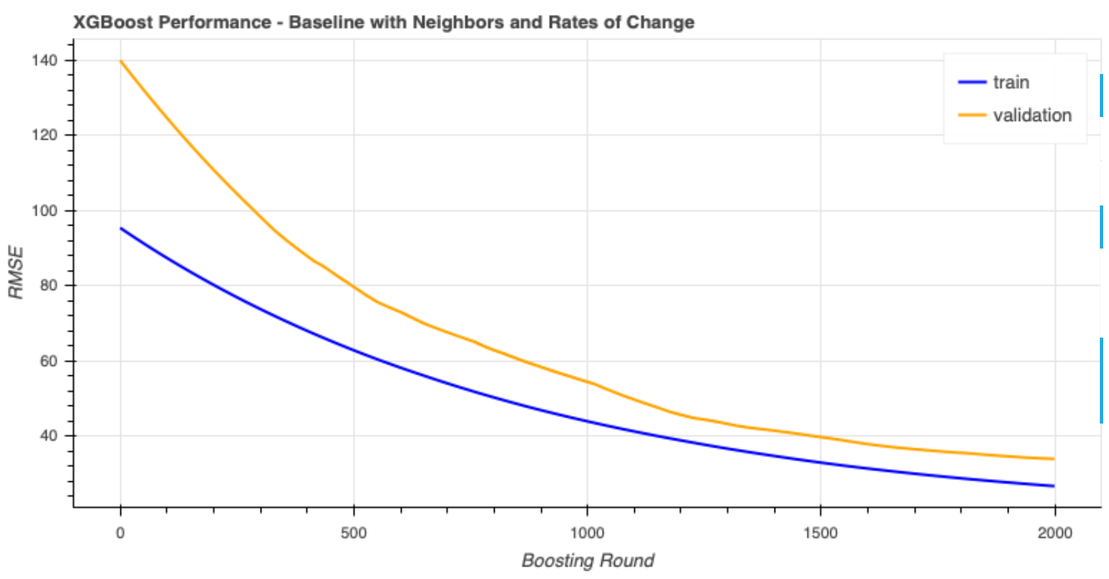
Out[110…]
```
Index(['FIPS', 'Date_Block', 'New_Cases', 'Cases_14_Days_Forward',
       'Daily_Change_In_New_Cases', 'Daily_Acceleration_In_New_Cases',
       'Daily_Aberrancy_In_New_Cases', 'Neighbor_24023_to_County_24001.0',
       'Neighbor_24023_to_County_24001.0_Daily_Change_In_New_Cases',
       'Neighbor_24023_to_County_24001.0_Daily_Acceleration_In_New_Cases',
       ...
       'Neighbor_24039_to_County_24047.0_Daily_Acceleration_In_New_Cases',
       'Neighbor_24039_to_County_24047.0_Daily_Aberrancy_In_New_Cases',
       'Neighbor_24045_to_County_24047.0',
       'Neighbor_24045_to_County_24047.0_Daily_Change_In_New_Cases',
       'Neighbor_24045_to_County_24047.0_Daily_Acceleration_In_New_Cases',
       'Neighbor_24045_to_County_24047.0_Daily_Aberrancy_In_New_Cases',
       'Neighbor_51001_to_County_24047.0',
       'Neighbor_51001_to_County_24047.0_Daily_Change_In_New_Cases',
       'Neighbor_51001_to_County_24047.0_Daily_Acceleration_In_New_Cases',
       'Neighbor_51001_to_County_24047.0_Daily_Aberrancy_In_New_Cases'],
      dtype='object', length=587)
```

In [111…]
```
X_train = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_roc_data['Date_Block'] > 14) & (baseline_plus_neighbors_roc_data['Date_Block'] <= 289)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_roc_data['Date_Block'] > 14) & (baseline_plus_neighbors_roc_data['Date_Block'] <= 289)]['Cases_14_Days_Forward']

X_validation = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_data['Date_Block'] > 289) & (baseline_plus_neighbors_roc_data['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_data['Date_Block'] > 289) & (baseline_plus_neighbors_roc_data['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [112…]
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

baseline_plus_neighbors_roc_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 2000, watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=500, evals_result = progress)
```

```
[0]     train-rmse:95.3425      validation-rmse:139.91
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 500 rounds.
[100]   train-rmse:87.3345      validation-rmse:124.602
[200]   train-rmse:80.1657      validation-rmse:110.757
[300]   train-rmse:73.7292      validation-rmse:98.2719
[400]   train-rmse:67.9531      validation-rmse:87.8642
[500]   train-rmse:62.7642      validation-rmse:79.6482
[600]   train-rmse:58.1398      validation-rmse:72.9396
[700]   train-rmse:53.9777      validation-rmse:67.5541
[800]   train-rmse:50.2181      validation-rmse:62.8084
[900]   train-rmse:46.8539      validation-rmse:58.3533
[1000]  train-rmse:43.8675      validation-rmse:54.3517
[1100]  train-rmse:41.1869      validation-rmse:49.6489
[1200]  train-rmse:38.7892      validation-rmse:45.6004
[1300]  train-rmse:36.6022      validation-rmse:43.1719
[1400]  train-rmse:34.6513      validation-rmse:41.3784
[1500]  train-rmse:32.8807      validation-rmse:39.6545
[1600]  train-rmse:31.3165      validation-rmse:37.8158
[1700]  train-rmse:29.9417      validation-rmse:36.4475
[1800]  train-rmse:28.7074      validation-rmse:35.4864
[1900]  train-rmse:27.6171      validation-rmse:34.55
[1999]  train-rmse:26.6431      validation-rmse:33.8711
```

In [113…]
```
xgb_perf("Baseline with Neighbors and Rates of Change")
```



In [114…]
```
X_test = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_roc_data['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = baseline_plus_neighbors_roc_data[(baseline_plus_neighbors_roc_data['Date_Block'] == 304)]['Cases_14_Days_Forward']
```

In [115…]
```
prediction = baseline_plus_neighbors_roc_model.predict(xgb.DMatrix(X_test), ntree_limit=baseline_plus_neighbors_roc_model.best_ntree_limit)
```

In [116…]
```
results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[116…

	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	33.231346
3	Anne Arundel	24003.0	83.0	235.413101
5	Baltimore	24005.0	127.0	253.429260
9	Calvert	24009.0	8.0	30.352615
11	Caroline	24011.0	2.0	20.042467
13	Carroll	24013.0	16.0	40.241989
15	Cecil	24015.0	8.0	40.639820
17	Charles	24017.0	39.0	46.451126
19	Dorchester	24019.0	7.0	17.637766
21	Frederick	24021.0	37.0	121.380898
23	Garrett	24023.0	2.0	13.841155
25	Harford	24025.0	31.0	70.119797
27	Howard	24027.0	48.0	68.674934
29	Kent	24029.0	3.0	12.036987
31	Montgomery	24031.0	147.0	310.118256
33	Prince George's	24033.0	195.0	311.367767
35	Queen Anne's	24035.0	8.0	16.440384
39	St. Mary's	24037.0	18.0	63.050835
37	Somerset	24039.0	7.0	18.444042
41	Talbot	24041.0	9.0	20.959269
45	Washington	24043.0	27.0	100.351837
47	Wicomico	24045.0	32.0	47.607449
49	Worcester	24047.0	8.0	20.452845
7	Baltimore City	24510.0	74.0	118.500381
43	Unassigned	90024.0	0.0	3.353017

In [117…

```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[117… 63.936494484010865

Baseline plus rolling averages

In [118…

```
baseline_plus_averages_data = baseline_data.copy()
```

In [119…

```
baseline_plus_averages_data['Three_Day_Rolling_Average_New_Cases']=baseline_plus_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(3).mean())
baseline_plus_averages_data['Five_Day_Rolling_Average_New_Cases']=baseline_plus_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(5).mean())
baseline_plus_averages_data['Seven_Day_Rolling_Average_New_Cases']=baseline_plus_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(7).mean())
baseline_plus_averages_data['Ten_Day_Rolling_Average_New_Cases']=baseline_plus_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(10).mean())
baseline_plus_averages_data['Fourteen_Day_Rolling_Average_New_Cases']=baseline_plus_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(14).mean())
```

In [120…

```
baseline_plus_averages_data = baseline_plus_averages_data.sort_values(by = ['FIPS', 'Date_Block'])
```

In [121…

```
X_train = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] > 14) & (baseline_plus_averages_data['Date_Block'] <= 289)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_train = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] > 14) & (baseline_plus_averages_data['Date_Block'] <= 289)]['Cases_14_Days_Forward']

X_validation = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] > 289) & (baseline_plus_averages_data['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_validation = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] > 289) & (baseline_plus_averages_data['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

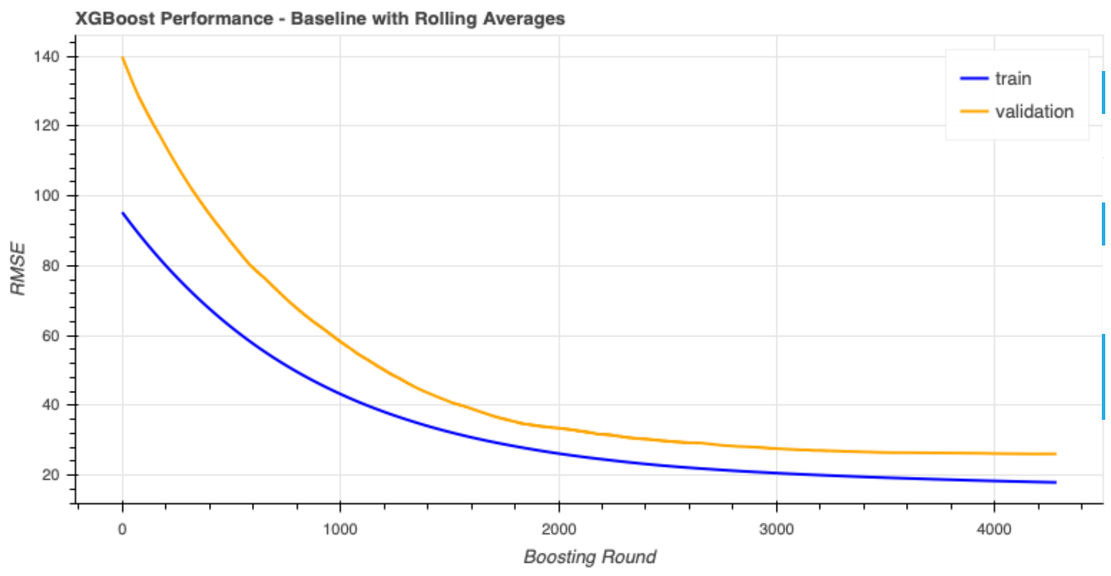
In [122…

```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

baseline_plus_averages_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 1000000, watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=100, evals_result = progress)
```

```
[0]     train-rmse:95.3424      validation-rmse:139.911
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 100 rounds.
[100]   train-rmse:87.3039      validation-rmse:125.556
[200]   train-rmse:80.0649      validation-rmse:114.207
[300]   train-rmse:73.5578      validation-rmse:103.798
[400]   train-rmse:67.7009      validation-rmse:94.8196
[500]   train-rmse:62.4273      validation-rmse:86.6957
[600]   train-rmse:57.6913      validation-rmse:79.4124
[700]   train-rmse:53.4589      validation-rmse:73.577
[800]   train-rmse:49.6804      validation-rmse:67.8449
[900]   train-rmse:46.3075      validation-rmse:62.9299
[1000]  train-rmse:43.2882      validation-rmse:58.2886
[1100]  train-rmse:40.5792      validation-rmse:54.0081
[1200]  train-rmse:38.1739      validation-rmse:50.1904
[1300]  train-rmse:36.0363      validation-rmse:46.6984
[1400]  train-rmse:34.1091      validation-rmse:43.6318
[1500]  train-rmse:32.3972      validation-rmse:41.0944
[1600]  train-rmse:30.8747      validation-rmse:39.0852
[1700]  train-rmse:29.5203      validation-rmse:36.989
[1800]  train-rmse:28.3127      validation-rmse:35.3574
[1900]  train-rmse:27.2135      validation-rmse:34.2023
[2000]  train-rmse:26.2313      validation-rmse:33.5187
[2100]  train-rmse:25.366       validation-rmse:32.6412
[2200]  train-rmse:24.5945      validation-rmse:31.7207
[2300]  train-rmse:23.875       validation-rmse:30.9534
[2400]  train-rmse:23.2299      validation-rmse:30.3604
[2500]  train-rmse:22.6597      validation-rmse:29.7787
[2600]  train-rmse:22.1676      validation-rmse:29.3177
[2700]  train-rmse:21.7276      validation-rmse:28.9179
[2800]  train-rmse:21.317       validation-rmse:28.3716
[2900]  train-rmse:20.9556      validation-rmse:28.0584
[3000]  train-rmse:20.6254      validation-rmse:27.6399
[3100]  train-rmse:20.3283      validation-rmse:27.32
[3200]  train-rmse:20.0488      validation-rmse:27.1076
[3300]  train-rmse:19.7843      validation-rmse:26.8901
[3400]  train-rmse:19.5451      validation-rmse:26.7003
[3500]  train-rmse:19.3214      validation-rmse:26.5254
[3600]  train-rmse:19.1098      validation-rmse:26.4825
[3700]  train-rmse:18.9126      validation-rmse:26.4342
[3800]  train-rmse:18.725       validation-rmse:26.3803
[3900]  train-rmse:18.5467      validation-rmse:26.3345
[4000]  train-rmse:18.3779      validation-rmse:26.231
[4100]  train-rmse:18.2193      validation-rmse:26.1682
[4200]  train-rmse:18.0718      validation-rmse:26.1258
Stopping. Best iteration:
[4183]  train-rmse:18.0952      validation-rmse:26.1214
```

In [123…

```
xgb_perf("Baseline with Rolling Averages")
```



In [124…

```
X_test = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = baseline_plus_averages_data[(baseline_plus_averages_data['Date_Block'] == 304)]['Cases_14_Days_Forward']

prediction = baseline_plus_averages_model.predict(xgb.DMatrix(X_test), ntree_limit=baseline_plus_averages_model.best_ntree_limit)

results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[124…

	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	51.891506
3	Anne Arundel	24003.0	83.0	170.505157
5	Baltimore	24005.0	127.0	194.828979
9	Calvert	24009.0	8.0	49.747776
11	Caroline	24011.0	2.0	24.510384
13	Carroll	24013.0	16.0	46.168755
15	Cecil	24015.0	8.0	52.050186
17	Charles	24017.0	39.0	59.217537
19	Dorchester	24019.0	7.0	20.083805
21	Frederick	24021.0	37.0	219.510834
23	Garrett	24023.0	2.0	15.828080
25	Harford	24025.0	31.0	87.098961
27	Howard	24027.0	48.0	130.920135
29	Kent	24029.0	3.0	13.908047
31	Montgomery	24031.0	147.0	365.357147
33	Prince George's	24033.0	195.0	340.310059
35	Queen Anne's	24035.0	8.0	25.264954
39	St. Mary's	24037.0	18.0	58.032623
37	Somerset	24039.0	7.0	17.283356
41	Talbot	24041.0	9.0	21.433895
45	Washington	24043.0	27.0	92.025177
47	Wicomico	24045.0	32.0	54.817654
49	Worcester	24047.0	8.0	28.133091
7	Baltimore City	24510.0	74.0	146.254059
43	Unassigned	90024.0	0.0	1.057636

In [125…

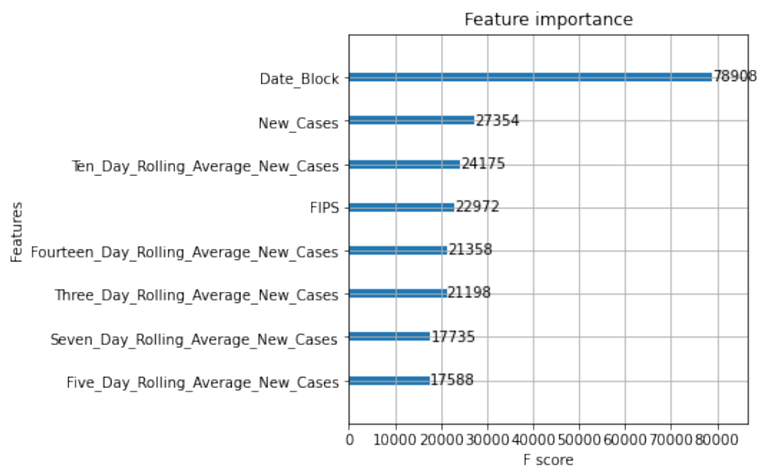
```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[125… `76.1627800085347`

In [126…

```
plt.rcParams["figure.figsize"] = (5, 5)
plot_importance(baseline_plus_averages_model)
```

Out[126… `<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>`



Baseline plus neighbors and rolling averages

In [127…

```
baseline_plus_neighbors_averages_data = baseline_data.copy()
```

In [128…

```
baseline_plus_neighbors_averages_data['Three_Day_Rolling_Average_New_Cases']=baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(3
baseline_plus_neighbors_averages_data['Five_Day_Rolling_Average_New_Cases']=baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(5)
baseline_plus_neighbors_averages_data['Seven_Day_Rolling_Average_New_Cases']=baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(7
baseline_plus_neighbors_averages_data['Ten_Day_Rolling_Average_New_Cases']=baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rolling(10)
baseline_plus_neighbors_averages_data['Fourteen_Day_Rolling_Average_New_Cases']=baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block']).groupby('FIPS')['New_Cases'].transform(lambda x: x.rollin
```

In [129…]
```
for county in counties:

    neighbors = county_adjacency_list[county_adjacency_list['fipscounty'] == county]['fipsneighbor'].unique()

    for neighbor in neighbors:
        if neighbor == county:
            continue
        else:
            col = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county)

            col_three = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county) +'_Three_Day_Rolling_Average_New_Cases'
            col_five = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county) +'_Five_Day_Rolling_Average_New_Cases'
            col_seven = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county) +'_Seven_Day_Rolling_Average_New_Cases'
            col_ten = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county) +'_Ten_Day_Rolling_Average_New_Cases'
            col_fourteen = 'Neighbor_' + str(neighbor) + '_to_County_' + str(county) +'_Fourteen_Day_Rolling_Average_New_Cases'

            tmp = baseline_plus_neighbors_averages_data.sort_values(['FIPS','Date_Block'])[['FIPS','Date_Block','New_Cases', 'Three_Day_Rolling_Average_New_Cases', 'Five_Day_Rolling_Average_New_Cases', 'Seven_D

            tmp = tmp[tmp['FIPS'] == neighbor]
            tmp['FIPS'] = county

            tmp = tmp.rename(columns={"New_Cases": col,
                                      "Three_Day_Rolling_Average_New_Cases": col_three,
                                      "Five_Day_Rolling_Average_New_Cases": col_five,
                                      "Seven_Day_Rolling_Average_New_Cases": col_seven,
                                      "Ten_Day_Rolling_Average_New_Cases": col_ten,
                                      "Fourteen_Day_Rolling_Average_New_Cases": col_fourteen})

            baseline_plus_neighbors_averages_data = pd.merge(baseline_plus_neighbors_averages_data, tmp, on = ["FIPS", 'Date_Block'], how = 'left')
```

In [130…]
```
baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] == 300) & (baseline_plus_neighbors_averages_data['FIPS'] == 24031.0)][['FIPS', 'Date_Block', 'Fourteen_Day_Rolling_Aver
```

Out[130…]

	FIPS	Date_Block	Fourteen_Day_Rolling_Average_New_Cases	Neighbor_24033_to_County_24031.0_Fourteen_Day_Rolling_Average_New_Cases
4567	24031.0	300.0	477.5	485.5

In [131…]
```
baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] == 300) & (baseline_plus_neighbors_averages_data['FIPS'] == 24033.0)][['FIPS', 'Date_Block', 'Fourteen_Day_Rolling_Aver
```

Out[131…]

	FIPS	Date_Block	Fourteen_Day_Rolling_Average_New_Cases	Neighbor_24031_to_County_24033.0_Fourteen_Day_Rolling_Average_New_Cases
4872	24033.0	300.0	485.5	477.5

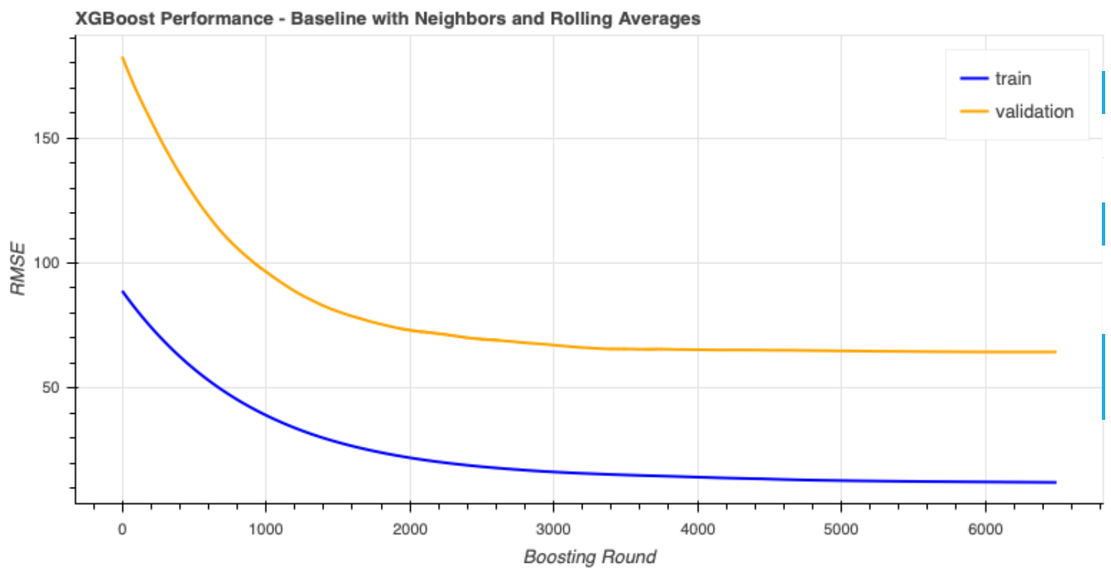
In [132…]
```
baseline_plus_neighbors_averages_data = baseline_plus_neighbors_averages_data.sort_values(by = ['FIPS', 'Date_Block'])
```

In [133…]
```
X_train = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] > 60) & (baseline_plus_neighbors_averages_data['Date_Block'] <= 275)].drop(['Cases_14_Days_Forward'], axis =
Y_train = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] > 60) & (baseline_plus_neighbors_averages_data['Date_Block'] <= 275)]['Cases_14_Days_Forward']

X_validation = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] > 275) & (baseline_plus_neighbors_averages_data['Date_Block'] <= 290)].drop(['Cases_14_Days_Forward'], a
Y_validation = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] > 275) & (baseline_plus_neighbors_averages_data['Date_Block'] <= 290)]['Cases_14_Days_Forward']
```

In [134…]
```
params = {
        'learning_rate': 0.001,
        'eval_metric': 'rmse',
        'seed': 42
    }

watchlist = [
    (xgb.DMatrix(X_train, Y_train), 'train'),
    (xgb.DMatrix(X_validation, Y_validation), 'validation')
]

progress = dict()

baseline_plus_neighbors_averages_model = xgb.train(params, xgb.DMatrix(X_train, Y_train), 1000000, watchlist, maximize=False, verbose_eval=100, early_stopping_rounds=250, evals_result = progress)
```

```
[0]     train-rmse:88.9049      validation-rmse:182.513
Multiple eval metrics have been passed: 'validation-rmse' will be used for early stopping.

Will train until validation-rmse hasn't improved in 250 rounds.
[100]   train-rmse:81.2956      validation-rmse:168.763
[200]   train-rmse:74.446       validation-rmse:156.971
[300]   train-rmse:68.2818      validation-rmse:145.832
[400]   train-rmse:62.7015      validation-rmse:135.78
[500]   train-rmse:57.6798      validation-rmse:126.903
[600]   train-rmse:53.168       validation-rmse:118.868
[700]   train-rmse:49.0982      validation-rmse:111.823
[800]   train-rmse:45.4401      validation-rmse:105.97
[900]   train-rmse:42.1512      validation-rmse:100.942
[1000]  train-rmse:39.1966      validation-rmse:96.5429
[1100]  train-rmse:36.5396      validation-rmse:92.4638
[1200]  train-rmse:34.1459      validation-rmse:88.7873
[1300]  train-rmse:32.0021      validation-rmse:85.6754
[1400]  train-rmse:30.0927      validation-rmse:82.9314
[1500]  train-rmse:28.3969      validation-rmse:80.6606
[1600]  train-rmse:26.8784      validation-rmse:78.7687
[1700]  train-rmse:25.5207      validation-rmse:77.0868
[1800]  train-rmse:24.3002      validation-rmse:75.583
[1900]  train-rmse:23.2 validation-rmse:74.2521
[2000]  train-rmse:22.2184      validation-rmse:73.2015
[2100]  train-rmse:21.3356      validation-rmse:72.4891
[2200]  train-rmse:20.5527      validation-rmse:71.8185
[2300]  train-rmse:19.8493      validation-rmse:71.0119
[2400]  train-rmse:19.2184      validation-rmse:70.1747
[2500]  train-rmse:18.645       validation-rmse:69.6209
[2600]  train-rmse:18.1387      validation-rmse:69.201
[2700]  train-rmse:17.6786      validation-rmse:68.7084
[2800]  train-rmse:17.2554      validation-rmse:68.175
[2900]  train-rmse:16.8656      validation-rmse:67.7438
[3000]  train-rmse:16.5411      validation-rmse:67.2307
[3100]  train-rmse:16.2433      validation-rmse:66.7148
[3200]  train-rmse:15.989       validation-rmse:66.2572
[3300]  train-rmse:15.7548      validation-rmse:65.913
[3400]  train-rmse:15.5338      validation-rmse:65.688
[3500]  train-rmse:15.3347      validation-rmse:65.6812
[3600]  train-rmse:15.1489      validation-rmse:65.5694
[3700]  train-rmse:14.9806      validation-rmse:65.5919
[3800]  train-rmse:14.8229      validation-rmse:65.5574
[3900]  train-rmse:14.6471      validation-rmse:65.4671
[4000]  train-rmse:14.467       validation-rmse:65.3766
[4100]  train-rmse:14.2911      validation-rmse:65.3162
[4200]  train-rmse:14.1191      validation-rmse:65.2702
[4300]  train-rmse:13.9741      validation-rmse:65.2869
[4400]  train-rmse:13.8403      validation-rmse:65.2464
[4500]  train-rmse:13.6981      validation-rmse:65.1922
[4600]  train-rmse:13.5326      validation-rmse:65.1616
[4700]  train-rmse:13.3875      validation-rmse:65.1254
[4800]  train-rmse:13.272       validation-rmse:65.0539
[4900]  train-rmse:13.1749      validation-rmse:64.9869
[5000]  train-rmse:13.0908      validation-rmse:64.932
[5100]  train-rmse:13.0084      validation-rmse:64.8772
[5200]  train-rmse:12.9393      validation-rmse:64.8027
[5300]  train-rmse:12.8675      validation-rmse:64.7549
[5400]  train-rmse:12.8066      validation-rmse:64.7183
[5500]  train-rmse:12.7493      validation-rmse:64.6715
[5600]  train-rmse:12.704       validation-rmse:64.6272
[5700]  train-rmse:12.6645      validation-rmse:64.5836
[5800]  train-rmse:12.6241      validation-rmse:64.5557
[5900]  train-rmse:12.5874      validation-rmse:64.5298
[6000]  train-rmse:12.551       validation-rmse:64.5081
[6100]  train-rmse:12.513       validation-rmse:64.4925
[6200]  train-rmse:12.4764      validation-rmse:64.4808
[6300]  train-rmse:12.44        validation-rmse:64.4818
[6400]  train-rmse:12.4014      validation-rmse:64.4826
Stopping. Best iteration:
[6247]  train-rmse:12.4593      validation-rmse:64.4801
```

In [135…]
```
xgb_perf("Baseline with Neighbors and Rolling Averages")
```



In [136…

```
X_test = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] == 304)].drop(['Cases_14_Days_Forward'], axis = 1)
Y_test = baseline_plus_neighbors_averages_data[(baseline_plus_neighbors_averages_data['Date_Block'] == 304)]['Cases_14_Days_Forward']

prediction = baseline_plus_neighbors_averages_model.predict(xgb.DMatrix(X_test), ntree_limit=baseline_plus_neighbors_averages_model.best_ntree_limit)

results['Predicted'] = prediction
results
```

```
/home/roman/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[136…

	Admin2	FIPS	New_Cases	Predicted
1	Allegany	24001.0	6.0	64.039345
3	Anne Arundel	24003.0	83.0	229.262192
5	Baltimore	24005.0	127.0	217.198700
9	Calvert	24009.0	8.0	37.654900
11	Caroline	24011.0	2.0	21.618217
13	Carroll	24013.0	16.0	52.744614
15	Cecil	24015.0	8.0	46.215649
17	Charles	24017.0	39.0	52.306355
19	Dorchester	24019.0	7.0	19.431511
21	Frederick	24021.0	37.0	129.992065
23	Garrett	24023.0	2.0	15.881237
25	Harford	24025.0	31.0	78.466858
27	Howard	24027.0	48.0	85.500595
29	Kent	24029.0	3.0	12.956955
31	Montgomery	24031.0	147.0	350.471130
33	Prince George's	24033.0	195.0	373.480682
35	Queen Anne's	24035.0	8.0	21.017349
39	St. Mary's	24037.0	18.0	51.381012
37	Somerset	24039.0	7.0	14.293799
41	Talbot	24041.0	9.0	20.212135
45	Washington	24043.0	27.0	108.327293
47	Wicomico	24045.0	32.0	48.715336
49	Worcester	24047.0	8.0	22.934902
7	Baltimore City	24510.0	74.0	186.342682
43	Unassigned	90024.0	0.0	1.752852

In [137…

```
mse = sklearn.metrics.mean_squared_error(results['New_Cases'], results['Predicted'])
rmse = mse**(0.5)
rmse
```

Out[137…

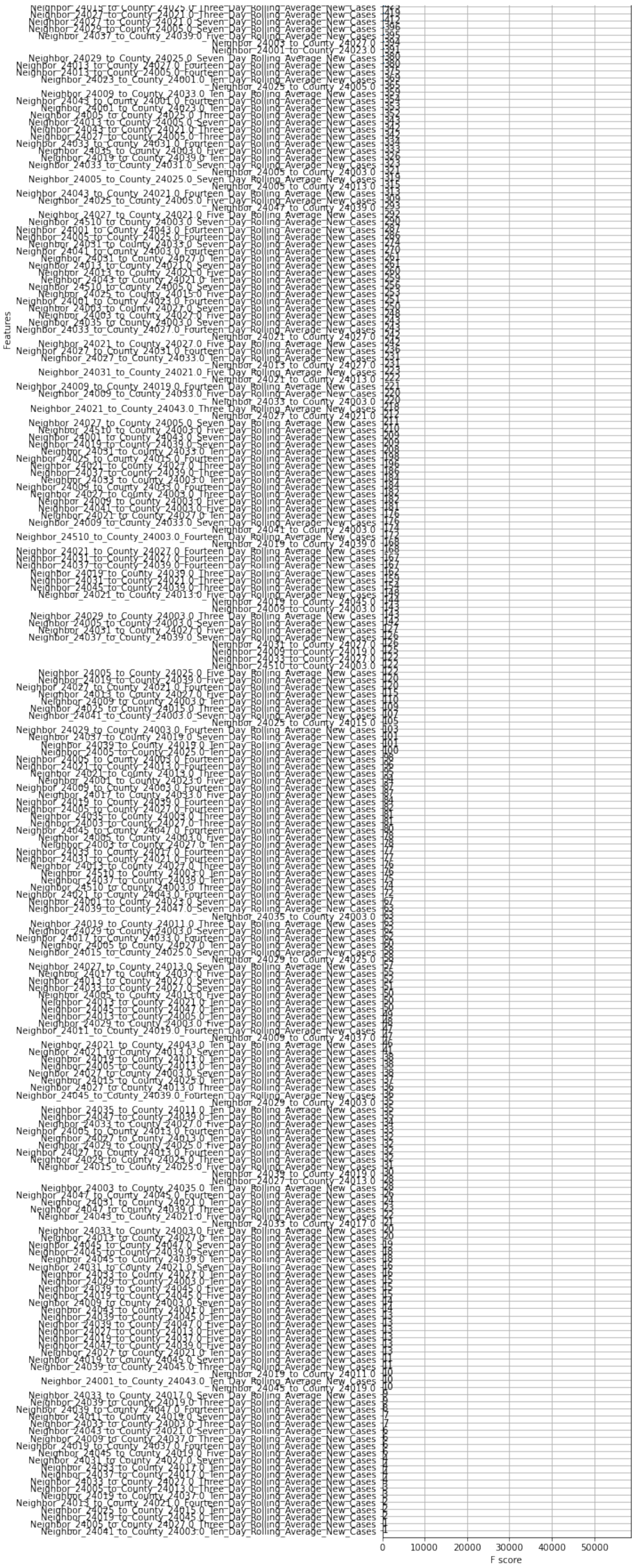
```
75.98981120842426
```

In [138…

```
plt.rcParams["figure.figsize"] = (5, 50)
plot_importance(baseline_plus_neighbors_averages_model)
```

Out[138…

```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>
```



Citations

Data source: "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University" url: https://github.com/CSSEGISandData/COVID-19.

Useful links:

- https://docs.bokeh.org/en/latest/docs/user_guide/styling.html#selected-and-unselected-glyphs
- https://docs.bokeh.org/en/latest/docs/user_guide/plotting.html
- https://docs.bokeh.org/en/latest/docs/user_guide/tools.html
- https://stackoverflow.com/questions/36561476/change-color-of-non-selected-bokeh-lines
- https://stackoverflow.com/questions/50835245/bokeh-multiline-plot
- https://stackoverflow.com/questions/31520951/plotting-multiple-lines-with-bokeh-and-pandas
- https://stackoverflow.com/questions/55524084/how-to-plot-bokeh-multiline-dataframe-from-csv
- https://github.com/bokeh/bokeh/issues/8658
- https://github.com/bokeh/bokeh/issues/2972
- http://holoviews.org/reference/streams/bokeh/heatmap_tap.html
- https://holoviews.org/gallery/demos/bokeh/measles_example.html
- https://data.nber.org/data/county-adjacency.html