

- i) Is this an example of supervised or unsupervised learning?

This is supervised learning because it is similar to regression. We are predicting one value by mapping multiple features.

- ii) Write the function x_{t+1} explicitly in terms of the model parameters $\theta = [\theta_0, \theta_1]$ and x_t assuming f represents a linear model.

$$x_{t+1} = \theta_0 + \theta_1 * x_t$$

iii) Suppose we obtain an optimal $\hat{\theta} = [\hat{\theta}_0, \hat{\theta}_1]$. Derive x_{t+1} in terms of x_0 , $\hat{\theta}_0$ and $\hat{\theta}_1$.

$$x_{t+1} = \hat{\theta}_0 + \hat{\theta}_1 * x_0$$

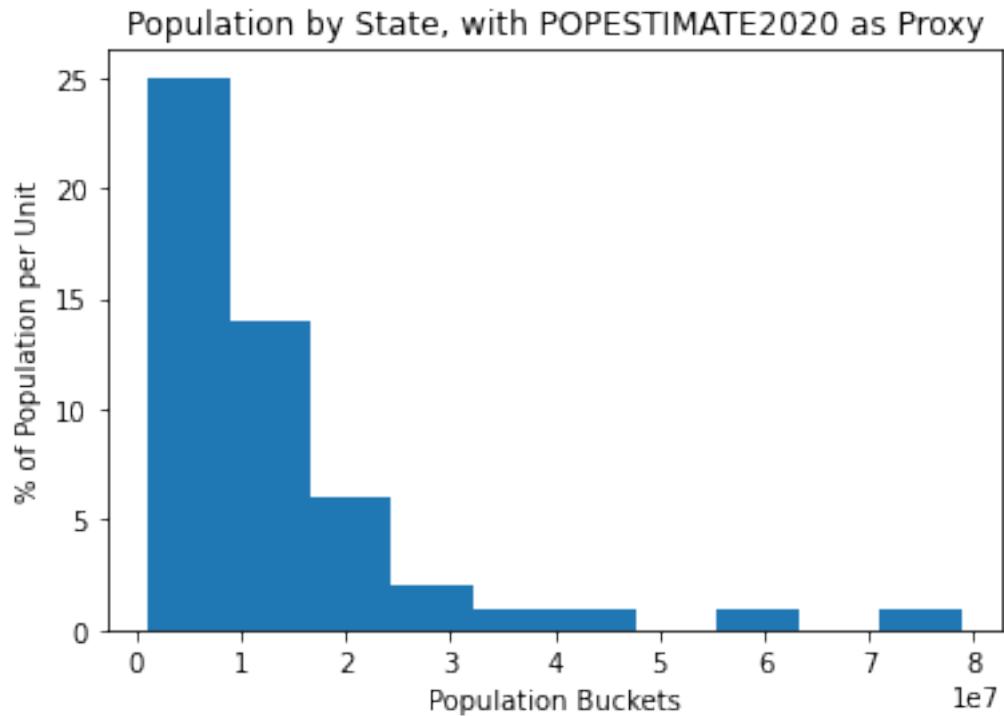
- iv) Suppose we add $k - 1$ more features such that our feature vector \tilde{x} contains k timesteps of past and current data. Describe how we could select an appropriate k .

We could select an appropriate k by utilizing cross-validation to find an appropriate balance between over- and under-fitting.

0.0.1 Question 1f

Generate the population by state using counties using the population estimate in 2020 (i.e. POPESTIMATE2020). Remark on any inconsistencies and propose a solution.

```
In [72]: plt.hist(counties[["POPESTIMATE2020", "STNAME"]].groupby("STNAME").sum())
plt.title("Population by State, with POPESTIMATE2020 as Proxy")
plt.ylabel("% of Population per Unit")
plt.xlabel("Population Buckets");
```

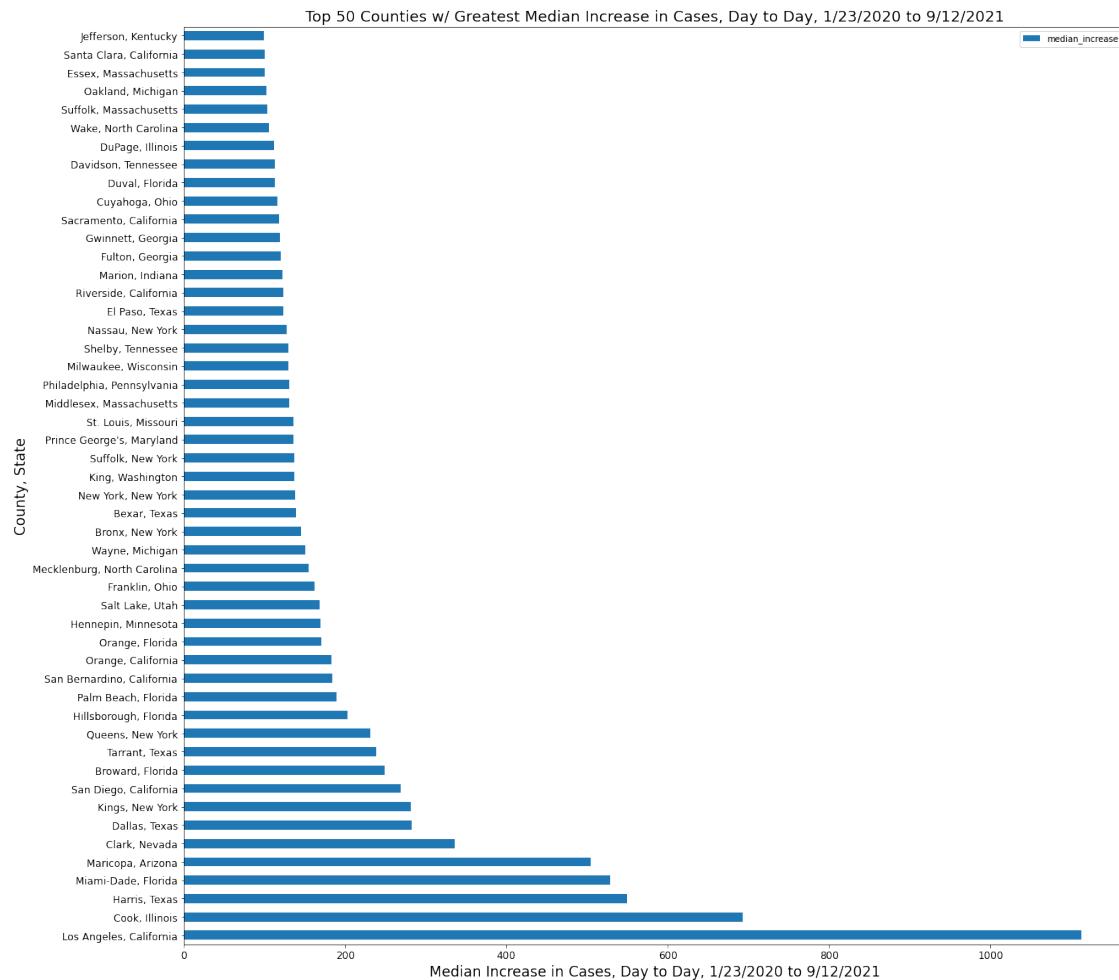


After generating a histogram plot of 2020 population estimate per state, it appears the data is skewed right. Thus, we can take the log of the numbers to make the distribution more normal.

0.0.2 Question 2a

Create a visualization that shows the median increase in number of cases for the 50 counties with the greatest median increase over January 23, 2020 to September 12, 2021. Make sure to include the name of state in which each county is located since county names are not necessarily unique.

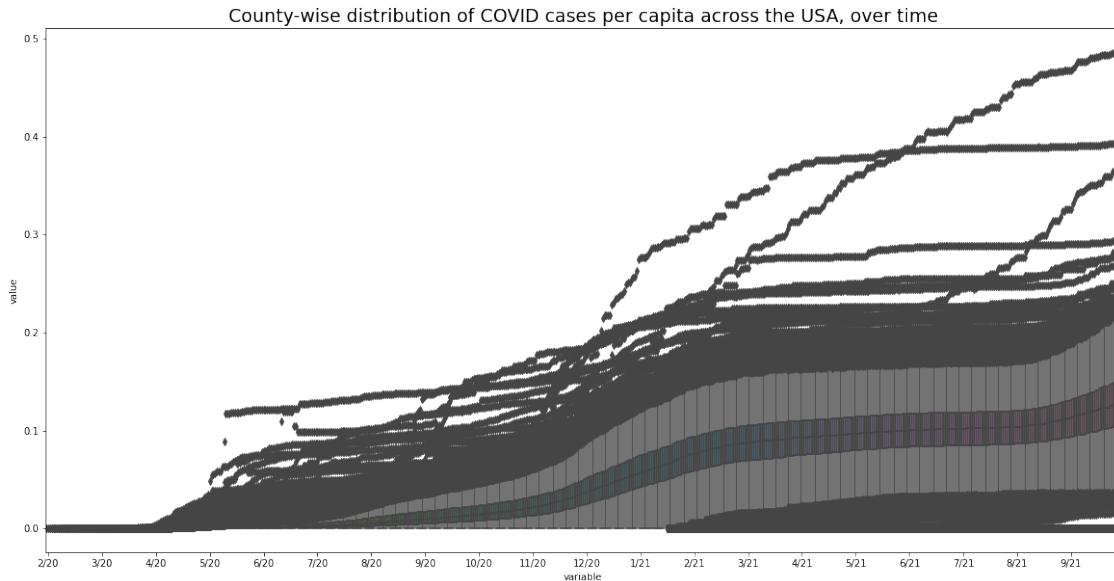
```
In [73]: increases = cases.loc[:, "1/23/20":"9/12/21"]
increases["median_increase"] = [np.median(i) for i in np.diff(increases, axis = 1)]
increases["County & State"] = cases["Admin2"] + ", " + cases["Province_State"]
top_50 = increases.sort_values("median_increase", ascending = False).head(50)[["County & State",
    "median_increase"]]
top_50.plot.barh(x = "County & State", y = "median_increase", rot = 0, fontsize = 12, figsize = (12, 10))
plt.xlabel("Median Increase in Cases, Day to Day, 1/23/2020 to 9/12/2021", fontsize = 17)
plt.ylabel("County, State", fontsize = 17)
plt.title("Top 50 Counties w/ Greatest Median Increase in Cases, Day to Day, 1/23/2020 to 9/12/2021")
```



0.0.3 Question 2b

Create a visualization that shows the overall county-wise distribution of the number of COVID-19 cases per capita across the United States as a function of time. For all questions that reference population, use the population estimates from 2020.

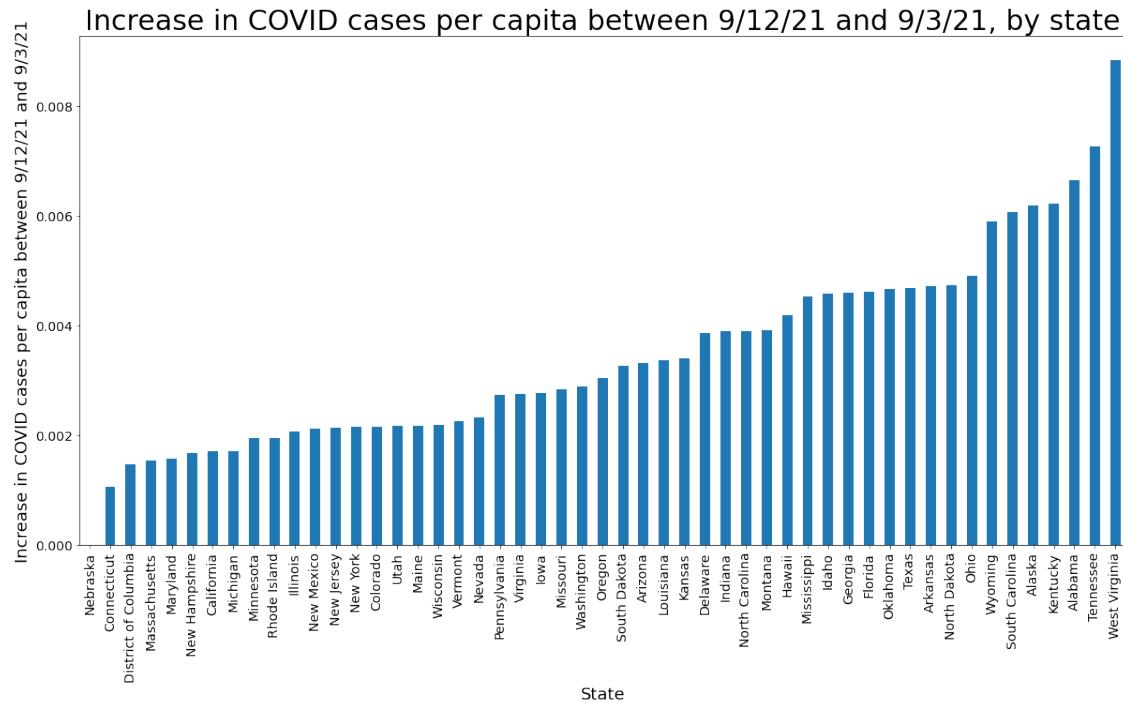
```
In [74]: timeline_per_capita = county_data.loc[:, "1/23/20":"9/12/21"]
for i in list(timeline_per_capita):
    timeline_per_capita[i] = timeline_per_capita[i]/county_data["POPESTIMATE2020"]
plt.figure(figsize=(20,10))
sns.boxplot(x = "variable", y = "value", data = pd.melt(timeline_per_capita))
plt.xticks(ticks = range(1, 600, 30), labels = ["2/20","3/20","4/20","5/20","6/20","7/20","8/20",
       "6/21","7/21","8/21","9/21"])
plt.title("County-wise distribution of COVID cases per capita across the USA, over time", font...
```



0.0.4 Question 2c

Create a visualization that shows states' increases in the number of COVID-19 cases per capita from September 3rd to September 12th, sorted from least to greatest increases.

```
In [75]: states = county_data.loc[:, "1/23/20":"9/12/21"]
states["State"] = county_data["STNAME"]
states["Population"] = county_data["POPESTIMATE2020"]
states = states.groupby("State").sum()
for i in list(states):
    states[i] = states[i]/states["Population"]
states_diffs = data = states["9/12/21"] - states["9/3/21"]
states_diffs = data = states_diffs.sort_values(ascending = True)
states_diffs.plot.bar(rot = 90, fontsize = 14, figsize = (20,10))
plt.xlabel("State", fontsize = 18)
plt.ylabel("Increase in COVID cases per capita between 9/12/21 and 9/3/21", fontsize = 18)
plt.title("Increase in COVID cases per capita between 9/12/21 and 9/3/21, by state", fontsize = 18)
```



0.0.5 Question 2d

Create a plot using a technique of your choice that visualizes the relationship between:

- a) the number of COVID-19 cases as a function of time for the Mobile, Tarrant, Cook, and San Fransisco counties located in Alabama, Texas, Illinois, and California.
- b) the frequency of never, rarely or sometimes wearing a mask in the respective counties

Hint: You may find a description of the data [here](#).

```
In [76]: four = county_data[(county_data["CTYNAME"] == "Mobile County") | (county_data["CTYNAME"] == "Tarrant County") | (county_data["CTYNAME"] == "Cook County") | (county_data["CTYNAME"] == "San Francisco County")]
four = four[four["STNAME"].isin(["Alabama", "Texas", "Illinois", "California"])]
final_four = four.loc[:, "1/22/20":"9/12/21"]
final_four["Population"] = four["POPESTIMATE2020"]
final_four["CTYNAME + STNAME"] = four["CTYNAME"] + ", " + four["STNAME"]
final_four["Never/Rarely/Sometimes Wearing a Mask"] = four["NEVER"] + four["RARELY"] + four["SOMETIMES"]
final_four = final_four.set_index("CTYNAME + STNAME")
final_four
for i in list(final_four)[-2:]:
    final_four[i] = final_four[i]/final_four["Population"]

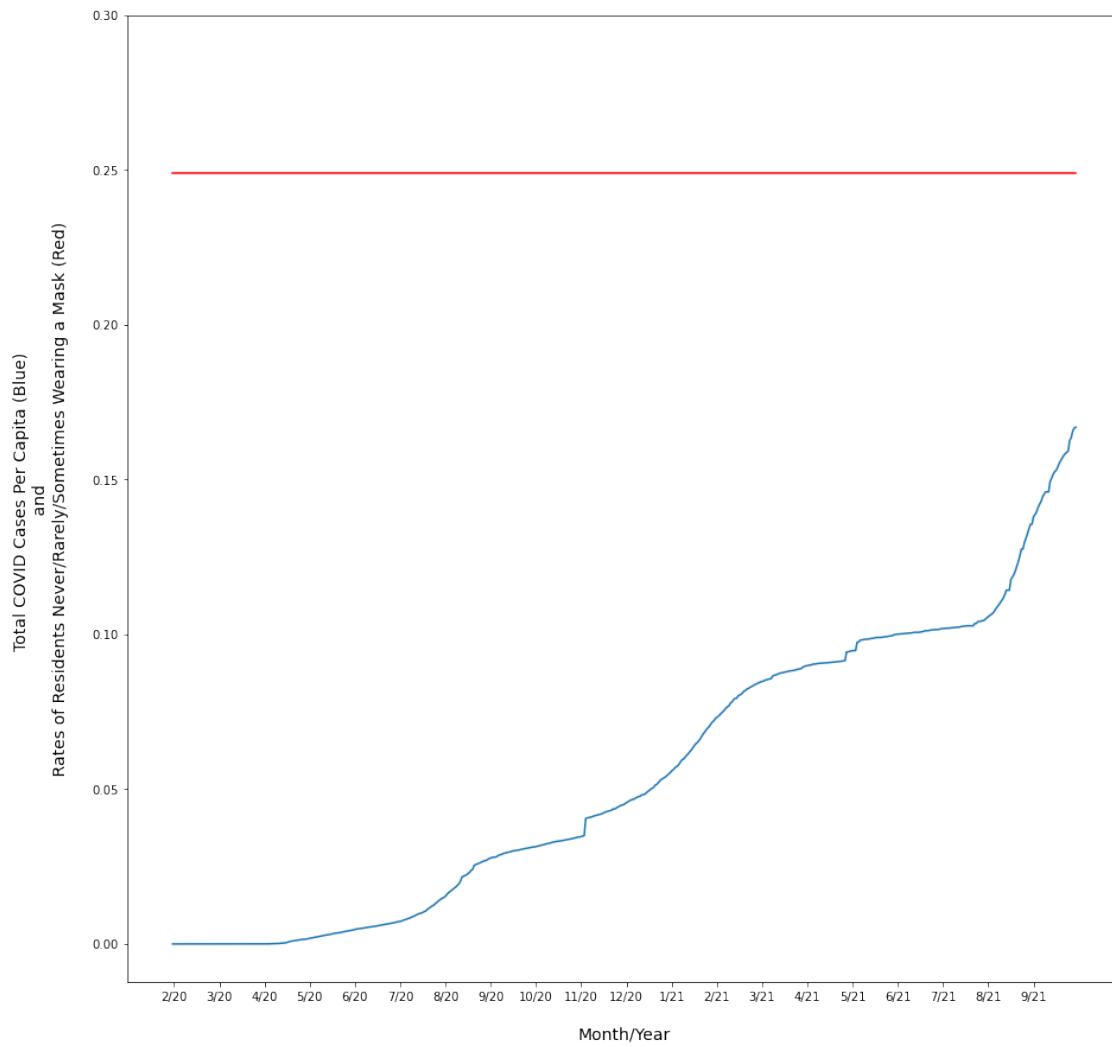
for i in final_four.index:

    fig, ax = plt.subplots(figsize=(15,15))
    x = pd.DataFrame(data = final_four.loc[i, "1/22/20":"9/12/21"])

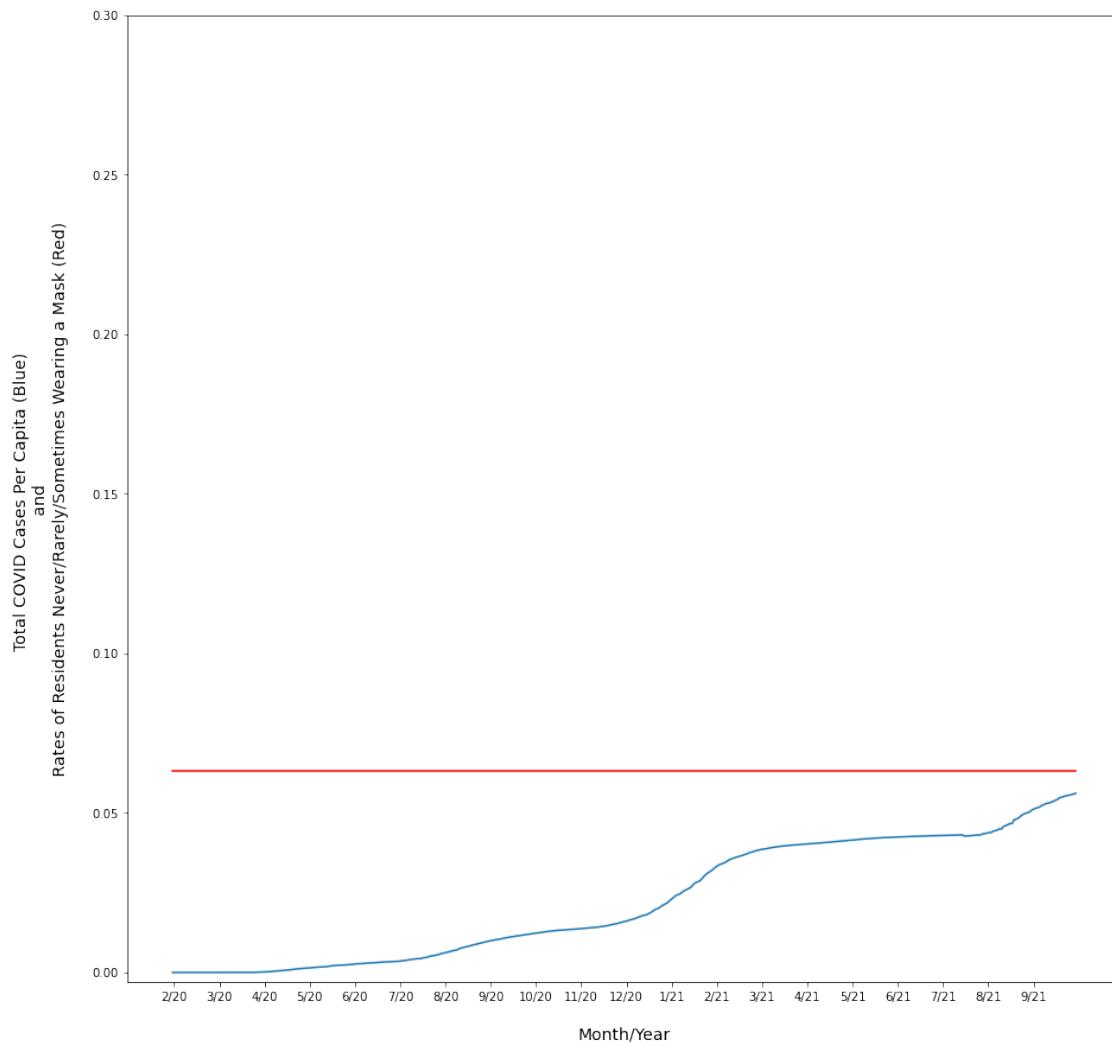
    ax.plot(x)
    ax.plot([final_four.loc[i, "Never/Rarely/Sometimes Wearing a Mask"] for n in range(x.shape[0])], color="blue")
    plt.xticks(ticks = range(1, 600, 30), labels = ["2/20", "3/20", "4/20", "5/20", "6/20", "7/20", "8/20", "9/20"])
    plt.yticks([0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3])
    plt.ylabel("Total COVID Cases Per Capita (Blue) \n and \n Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red)")

    plt.title(i + ": Total COVID Cases Per Capita (Blue) vs. \n Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red) \n from Jan 2020 to Sept 2021")
    plt.xlabel("\nMonth/Year\n", fontsize = 14)
```

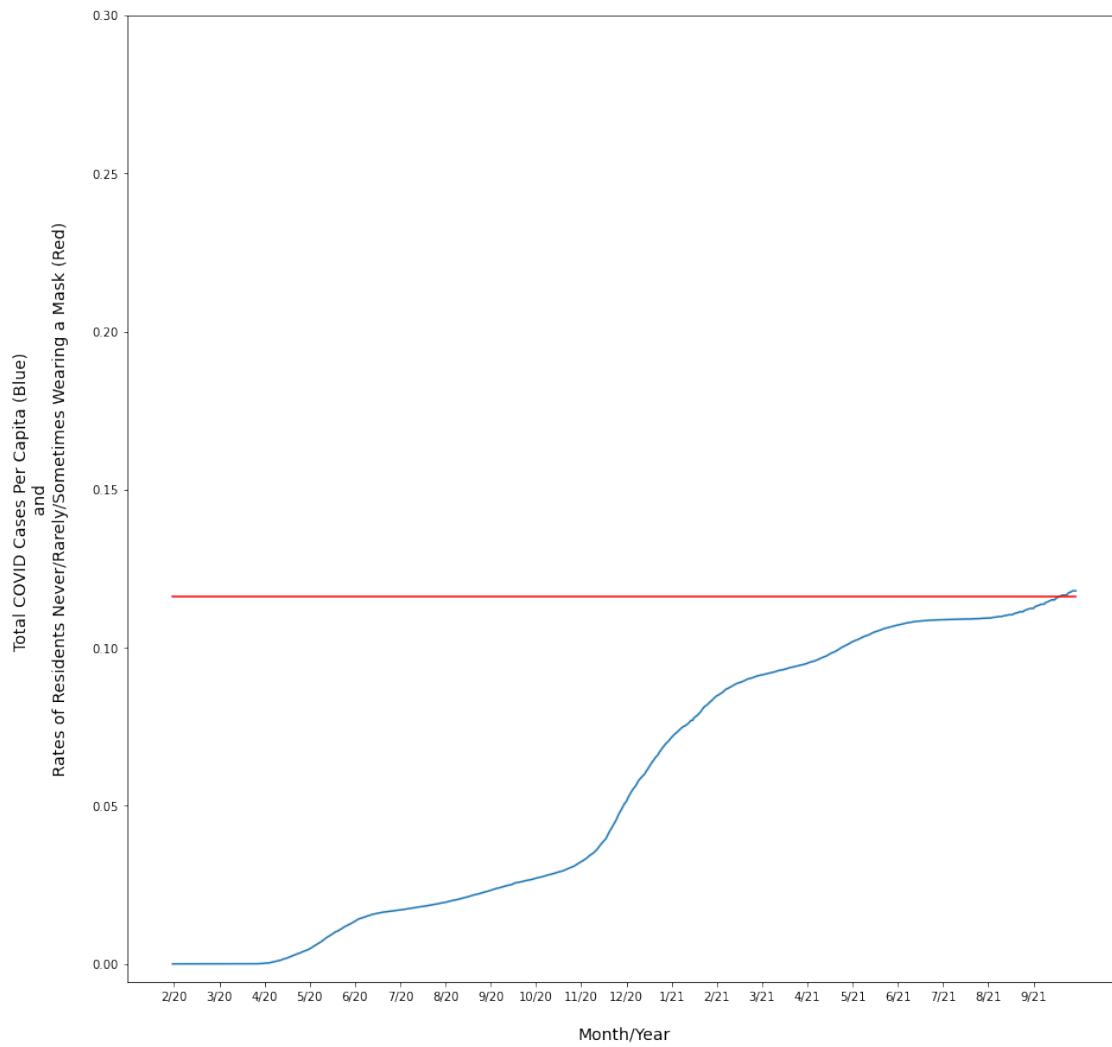
Mobile County, Alabama: Total COVID Cases Per Capita (Blue) vs.
Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red), 1/22/20 to 9/12/21



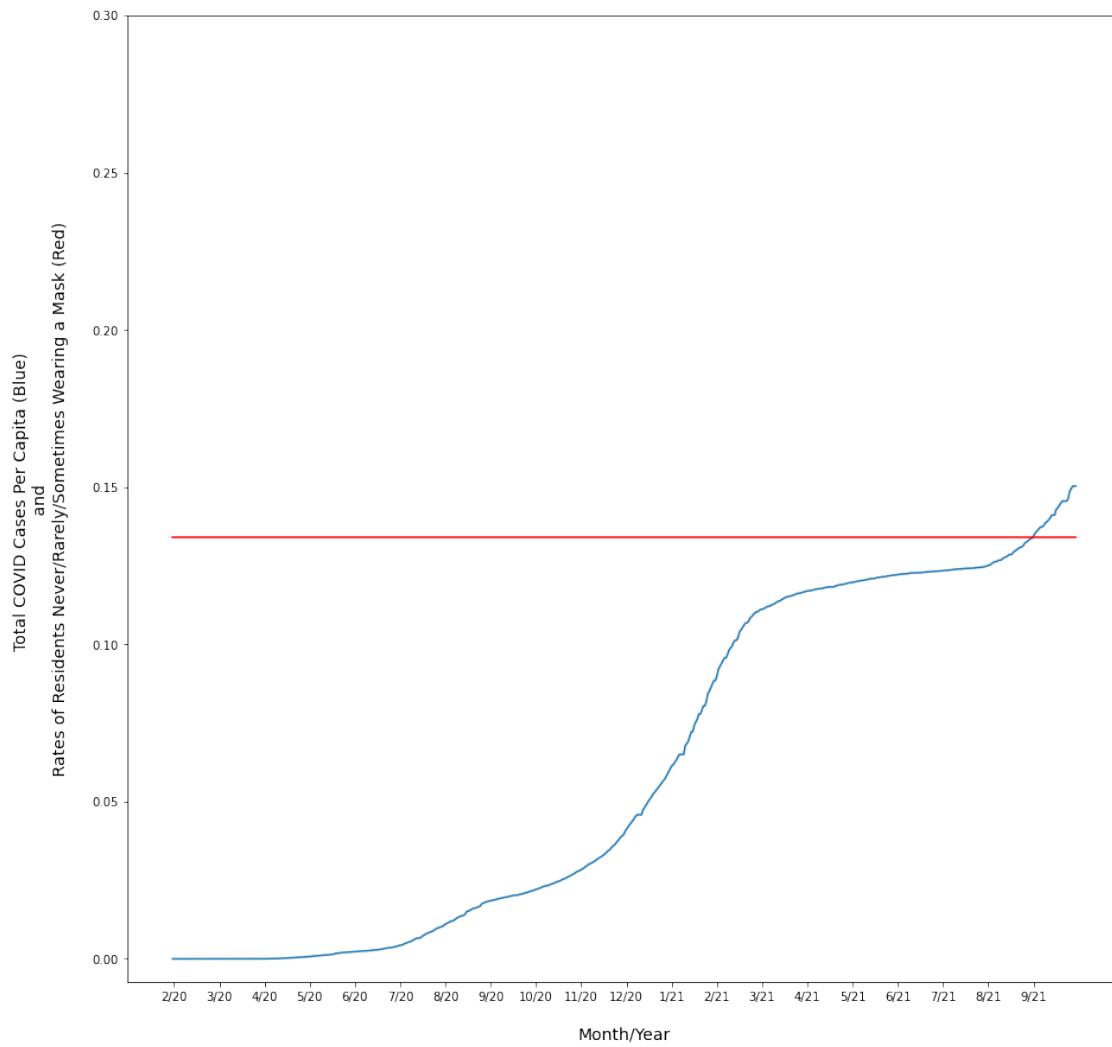
San Francisco County, California: Total COVID Cases Per Capita (Blue) vs.
Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red), 1/22/20 to 9/12/21



Cook County, Illinois: Total COVID Cases Per Capita (Blue) vs.
Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red), 1/22/20 to 9/12/21



Tarrant County, Texas: Total COVID Cases Per Capita (Blue) vs.
Rates of Residents Never/Rarely/Sometimes Wearing a Mask (Red), 1/22/20 to 9/12/21



0.0.6 Question 2e

Visualize and describe the spatial relationship between the location and the following quantities over the contiguous United States:

- a) the number of most recent COVID-19 cases per capita on September 12, 2021
- b) the frequency of never, rarely or sometimes wearing a mask

Hint: Use `plotly` to generate a heatmap on a geographical plot of the United States!

```
In [77]: grouped = county_data.groupby("STNAME").sum()
#grouped = grouped[(grouped.index != "Hawaii") & (grouped.index != "Alaska")]
grouped["Cases per Capita"] = grouped["9/12/21"] / grouped["POPESTIMATE2020"]
grouped = grouped[["Cases per Capita"]]

mask_usage = county_data.groupby("STNAME").mean()
mask_usage["Average Never/Rarely/Sometimes Wearing a Mask"] = mask_usage["NEVER"] + mask_usage["SOMETIMES"]
mask_usage = mask_usage[["Average Never/Rarely/Sometimes Wearing a Mask"]]
mask_usage

joined = grouped.merge(mask_usage, how = "inner", left_on = "STNAME", right_on = "STNAME")
joined = joined.reset_index()

joined = joined.replace({
    "Alabama": "AL",
    "Alaska": "AK",
    "Arizona": "AZ",
    "Arkansas": "AR",
    "California": "CA",
    "Colorado": "CO",
    "Connecticut": "CT",
    "Delaware": "DE",
    "Florida": "FL",
    "Georgia": "GA",
    "Hawaii": "HI",
    "Idaho": "ID",
    "Illinois": "IL",
    "Indiana": "IN",
    "Iowa": "IA",
    "Kansas": "KS",
    "Kentucky": "KY",
    "Louisiana": "LA",
    "Maine": "ME",
    "Maryland": "MD",
    "Massachusetts": "MA",
    "Michigan": "MI",
    "Minnesota": "MN",
```

```

    "Mississippi": "MS",
    "Missouri": "MO",
    "Montana": "MT",
    "Nebraska": "NE",
    "Nevada": "NV",
    "New Hampshire": "NH",
    "New Jersey": "NJ",
    "New Mexico": "NM",
    "New York": "NY",
    "North Carolina": "NC",
    "North Dakota": "ND",
    "Ohio": "OH",
    "Oklahoma": "OK",
    "Oregon": "OR",
    "Pennsylvania": "PA",
    "Rhode Island": "RI",
    "South Carolina": "SC",
    "South Dakota": "SD",
    "Tennessee": "TN",
    "Texas": "TX",
    "Utah": "UT",
    "Vermont": "VT",
    "Virginia": "VA",
    "Washington": "WA",
    "West Virginia": "WV",
    "Wisconsin": "WI",
    "Wyoming": "WY",
    "District of Columbia": "DC"})

```

```

import plotly.graph_objects as go

fig = go.Figure(data=go.Choropleth(
    locations=joined["STNAME"],
    z = joined["Cases per Capita"].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Reds',
    colorbar_title = "Cases per Capita"
))

fig.update_layout(
    title_text = 'COVID Cases Per Capita by State, 9/12/21',
    geo_scope='usa'
)
fig.show()

fig2 = go.Figure(data=go.Choropleth(
    locations=joined["STNAME"],

```

```

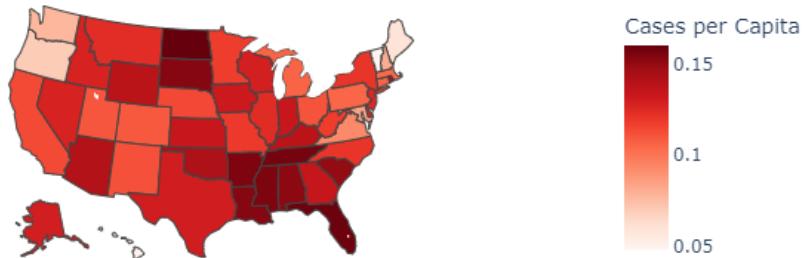
z = joined[["Average Never/Rarely/Sometimes Wearing a Mask"]].astype(float),
locationmode = 'USA-states',
colorscale = 'Blues',
colorbar_title = "Average Never/Rarely/Sometimes Wearing a Mask"
))

fig2.update_layout(
    title_text = 'Average Never/Rarely/Sometimes Wearing a Mask, by State',
    geo_scope='usa',
)

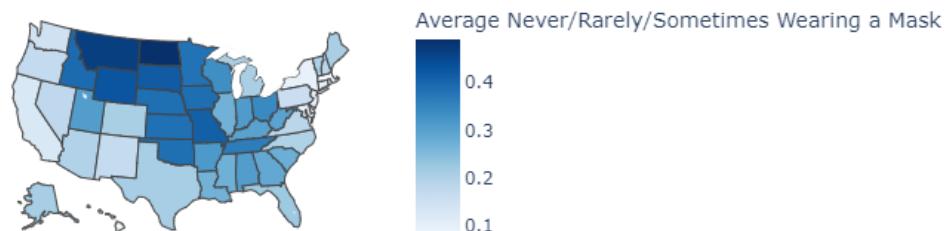
fig2.show()

```

COVID Cases Per Capita by State, 9/12/21



Average Never/Rarely/Sometimes Wearing a Mask, by State



0.0.7 Question 2f

Describe and interpret the relationships in the plots above. What conclusions can be drawn about the COVID-19 dataset from these plots?

Based on the group of plots as a whole, there are a few prevalent trends that can be extracted. The first is that mask usage is negatively associated with the spread of COVID. There are a few pieces of evidence for this. First, based on 2d, states with high rates of residents who never, rarely, or sometimes wear a mask generally had higher growth rates of cases. Additionally, based on the geoplots in 2e, states with less mask usage tended to also have a higher amount of cases per capita on 9/12/21; the converse appears to be true as well.

Additionally, from a broader perspective, some counties have much more cases than others. The county with the largest amount of median cases (LA County) had almost 1.7x the same figure for the county with the second largest figure (Cook County, Illinois), providing insight into the shape and distribution of the dataset.

Finally, in accordance with patterns reported in new outlets, there are certain periods where cases per capita across the USA were stagnant, and other periods where they spiked significantly. For instance, between November 2020 and February 2021, there was a steady increase in cases per capita, as illustrated by 2b. On the other hand, based on the same plot, there was a period from March 2021 to August 2021 in which cases per capita didn't rise significantly.

0.0.8 Question 3a

Using SVD, decompose the standardized matrix X that contains the standardized current proportion of fully vaccinated and partially vaccinated, cases per capita, and mask usage for every state as of the most recent day in the data.

Hint: The matrix X should have rows that describe location.

```
In [78]: recent = vaccinations.sort_values("Date", ascending = False)[["Date"]].iloc[900]

state_mask = pd.DataFrame(data = county_data).groupby("STNAME").mean()
state_mask = state_mask[["NEVER", "RARELY", "SOMETIMES", "FREQUENTLY", "ALWAYS"]]

mask_and_vax = vaccinations.merge(state_mask, how = "inner", left_on = "Province_State", right_on = "STNAME")
mask_and_vax = mask_and_vax[mask_and_vax["Date"] == recent]
mask_and_vax = mask_and_vax.groupby("Province_State").mean()

state_pops = county_data.groupby("STNAME").sum()[["POPESTIMATE2020"]].reset_index()

mask_and_vax = state_pops.merge(mask_and_vax, how = "inner", left_on = "STNAME", right_on = "STNAME")
state_cases = county_data.groupby("STNAME").sum()[["9/12/21"]].reset_index()
mask_and_vax_ag = state_cases.merge(mask_and_vax, how = "inner", left_on = "STNAME", right_on = "STNAME")

mask_and_vax_ag["Proportion Fully Vaccinated"] = mask_and_vax_ag["People_Fully_Vaccinated"]
mask_and_vax_ag["Proportion Fully Vaccinated"] = mask_and_vax_ag["Proportion Fully Vaccinated"]
mask_and_vax_ag["Cases per Capita"] = mask_and_vax_ag["9/12/21"]
mask_and_vax_ag["Cases per Capita"] = mask_and_vax_ag["Cases per Capita"]/mask_and_vax_ag["POPESTIMATE2020"]
mask_and_vax_ag["Proportion Partially Vaccinated"] = mask_and_vax_ag["People_Partially_Vaccinated"]
mask_and_vax_ag["Proportion Partially Vaccinated"] = mask_and_vax_ag["Proportion Partially Vaccinated"]

X = mask_and_vax_ag[["STNAME", "Proportion Fully Vaccinated", "Proportion Partially Vaccinated"]]

X_mean = np.mean(X, axis = 0)
X_std = np.std(X, axis = 0)
X = (X - X_mean)/X_std

u, s, vt = np.linalg.svd(X, full_matrices = False)
```


0.0.9 Question 3b

Visualize the top 2 principal components and label the data points with the corresponding location. Color each data points based on the number of cases per capita in the location to which it corresponds.

```
In [79]: import plotly.express as px

top2 = pd.DataFrame(data = u * s).loc[:, :1]
top2.rename(columns = {0:"pc1", 1:"pc2"}, inplace = True)
top2 = top2.reset_index().join(mask_and_vax)[["pc1", "pc2", "STNAME", "Cases per Capita"]]
top2

fig = px.scatter(top2, x="pc1", y="pc2", text="STNAME", color = "Cases per Capita");

fig.update_traces(textposition='top center');

fig.show();
```

```
-----  
KeyError                                     Traceback (most recent call last)  
/tmp/ipykernel_44/635451244.py in <module>  
      3 top2 = pd.DataFrame(data = u * s).loc[:, :1]  
      4 top2.rename(columns = {0:"pc1", 1:"pc2"}, inplace = True)  
----> 5 top2 = top2.reset_index().join(mask_and_vax)[["pc1", "pc2", "STNAME", "Cases per Capita"]]  
      6 top2  
      7  
  
/opt/conda/lib/python3.9/site-packages/pandas/core/frame.py in __getitem__(self, key)  
 3462             if is_iterator(key):  
 3463                 key = list(key)  
-> 3464             indexer = self.loc._get_listlike_indexer(key, axis=1)[1]  
 3465  
 3466         # take() does not accept boolean indexers  
  
/opt/conda/lib/python3.9/site-packages/pandas/core/indexing.py in _get_listlike_indexer(self, key, axis)  
 1312             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)  
 1313  
-> 1314             self._validate_read_indexer(keyarr, indexer, axis)  
 1315  
 1316             if needs_i8_conversion(ax.dtype) or isinstance(  
  
/opt/conda/lib/python3.9/site-packages/pandas/core/indexing.py in _validate_read_indexer(self, key, axis)  
 1375  
 1376             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())  
-> 1377             raise KeyError(f"{not_found} not in index")  
 1378  
 1379
```

KeyError: "['Cases per Capita'] not in index"

0.0.10 Question 3c

What could the first and second principal component roughly represent? Interpret the visualization above and report any findings from the first two principal components given by PCA.

The first PC could represent mask usage. It appears that data points with low PC1 values correspond to states with low mask usage, and vice versa. For instance, according to analysis in previous questions, Hawaii is the state that masks the most, and North Dakota the least.

The second PC could represent cases per capita within each subgroup of PC1 - i.e. within each subgroup of low and high mask usage. Within each of these two groups, a high PC2 would indicate a relatively low level of cases, and a low PC2 would indicate a relatively higher one. For instance, Maine has high PC2 and low cases per capita while Alabama and Florida have low PC2 but very high cases per capita.

It's important to note that the plot suggests the insight provided by PC2 is influenced by PC1. For instance, Alabama and Pennsylvania have comparable PC2's, but since they have significantly different PC1s, they have different amounts of cases per capita.

0.0.11 Question 3d

Construct a scree plot displaying the proportion of variance captured by each principal component.

```
In [ ]: #pcs = pd.DataFrame(data = u * s)
#pcs.rename(columns = {0:"pc1", 1:"pc2", 2: "pc3", 3: "pc4", 4: "pc5", 5: "pc6", 6: "pc7", 7: "pc8", 8: "pc9"});
#pcs

plt.plot(range(1, 9), s**2 / sum(s**2));
plt.xlabel('PC #');
plt.ylabel('Fraction of Variance Explained');
plt.title('Fraction of Variance Explained by each Principal Component');
```


0.0.12 Question 3f

Investigate the principal component neighbors for two states of your choice, and remark on any observations you have about their COVID-19 statistics given in the dataset, geographical or political ties.

Compare these neighbors with the rankings from the visualization from Question 2c that depicted recent greatest increases in cases across states.

State 1: Alabama Alabama's top-5 principal component neighbors are Arkansas, Mississippi, Georgia, South Carolina, and Louisiana. This is expected because they are all southern states, and all of them voted Red in 2020 (except Georgia, whose election was still very tight).

State 2: Delaware Delaware's top-5 principal component neighbors are California, New Jersey, Rhode Island, New York, and Massachusetts. Unlike Alabama and its top 5 neighbors, Delaware isn't that close to 3 of its top 5 neighbors (Rhode Island, Massachusetts, and especially California). However, similar to Alabama, these 6 states share similar voting patterns. In fact, all 6 of these states generally vote in favor of Democratic candidates.

0.1 Question 4: Open EDA

Perform EDA of your choice on the data to generate 2 visualizations. You may follow the line of investigation in the guided sections by visualizing the relationship between safety protocols and the spread of COVID-19 spatially or temporally, or you may explore other areas of your choice.

For each visualization, make sure to address the following:

1. Address a relationship in the data concerning the spread, effect, or prevention of COVID-19 through a visualization, with at *most* one univariate data visualization. If needed, use an unsupervised learning technique such as PCA to reveal patterns within the data.
2. Comment on the visualization in terms of the distribution, trends, and patterns it shows.
3. Comment on what the visualization indicates about the data with regards to the features you may choose for a supervised learning task.

```
In [ ]: # EDA 1: Distribution within Individual States: \nCases per 100,000 Residents of Counties
```

```
q4a = county_data[["STNAME", "CTYNAME", "POPESTIMATE2020", "3/3/21", "3/10/21"]]

nh = q4a[(q4a["STNAME"] == "New Hampshire")]
de = q4a[(q4a["STNAME"] == "Delaware")]
sc = q4a[(q4a["STNAME"] == "South Carolina")]

nh["County, State"] = nh["CTYNAME"] + ", " + nh["STNAME"]
de["County, State"] = de["CTYNAME"] + ", " + de["STNAME"]
sc["County, State"] = sc["CTYNAME"] + ", " + sc["STNAME"]

nh["New Cases per 100,000 Residents: 3/3/21 to 3/10/21"] = 100000*((nh["3/10/21"] - nh["3/3/21"])
de["New Cases per 100,000 Residents: 3/3/21 to 3/10/21"] = 100000*((de["3/10/21"] - de["3/3/21"])
sc["New Cases per 100,000 Residents: 3/3/21 to 3/10/21"] = 100000*((sc["3/10/21"] - sc["3/3/21"])

q4a_final = nh.append(de).append(sc)

sns.violinplot(x = q4a_final["STNAME"], y = q4a_final["New Cases per 100,000 Residents: 3/3/21"])
```

```
In [ ]: # EDA 2: California, North Dakota, Massachusetts, Florida: Frequent/always masking vs. Partially
q4b = county_data[["STNAME", "FREQUENTLY", "ALWAYS", "POPESTIMATE2020"]]
q4b["Frequently + Always"] = q4b["FREQUENTLY"] + county_data["ALWAYS"]
q4b["Frequent/Always Maskers"] = q4b["Frequently + Always"]*q4b["POPESTIMATE2020"]
q4b = q4b[["STNAME", "Frequent/Always Maskers", "POPESTIMATE2020"]]
q4b = q4b.groupby("STNAME").sum()
q4b["Proportion of Frequent/Always Masking"] = q4b["Frequent/Always Maskers"] / q4b["POPESTIMATE2020"]
```

```

q4b = q4b[["Proportion of Frequent/Always Masking", "POPESTIMATE2020"]].reset_index()

q4b = q4b[(q4b["STNAME"] == "California") | (q4b["STNAME"] == "North Dakota") | (q4b["STNAME"] == "Massachusetts") | (q4b["STNAME"] == "Florida")]
q4b_vax = vaccinations[(vaccinations["Province_State"] == "California") | (vaccinations["Province_State"] == "North Dakota") | (vaccinations["Province_State"] == "Massachusetts") | (vaccinations["Province_State"] == "Florida")]

q4b = q4b.merge(q4b_vax, how = "inner", left_on = "STNAME", right_on = "Province_State")

q4b["Proportion Partially/Fully Vaccinated"] = (q4b["People_Fully_Vaccinated"] + q4b["People_Partially_Vaccinated"]) / q4b["POPESTIMATE2020"]

ca = q4b[q4b["STNAME"] == "California"]
ca[["Proportion of Frequent/Always Masking", "Proportion Partially/Fully Vaccinated", "Date"]].plot()
plt.ylabel("Proportion Partially/Fully Vaccinated \n and \n Proportion of Frequent/Always Masking")
plt.xlabel("Date\n", fontsize = 14)
plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
plt.title("California: Time Series of Frequent/Always Masking vs. Tendency for Residents to get Vaccinated")

nd = q4b[q4b["STNAME"] == "North Dakota"]
nd[["Proportion of Frequent/Always Masking", "Proportion Partially/Fully Vaccinated", "Date"]].plot()
plt.ylabel("Proportion Partially/Fully Vaccinated \n and \n Proportion of Frequent/Always Masking")
plt.xlabel("Date\n", fontsize = 14)
plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
plt.title("North Dakota: Time Series of Frequent/Always Masking vs. Tendency for Residents to get Vaccinated")

ma = q4b[q4b["STNAME"] == "Massachusetts"]
ma[["Proportion of Frequent/Always Masking", "Proportion Partially/Fully Vaccinated", "Date"]].plot()
plt.ylabel("Proportion Partially/Fully Vaccinated \n and \n Proportion of Frequent/Always Masking")
plt.xlabel("Date\n", fontsize = 14)
plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
plt.title("Massachusetts: Time Series of Frequent/Always Masking vs. Tendency for Residents to get Vaccinated")

fl = q4b[q4b["STNAME"] == "Florida"]
fl[["Proportion of Frequent/Always Masking", "Proportion Partially/Fully Vaccinated", "Date"]].plot()
plt.ylabel("Proportion Partially/Fully Vaccinated \n and \n Proportion of Frequent/Always Masking")
plt.xlabel("Date\n", fontsize = 14)
plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])
plt.title("Florida: Time Series of Frequent/Always Masking vs. Tendency for Residents to get Vaccinated")

In [ ]: # EDA 3:
vacc_data = vaccinations[['Date', 'People_Fully_Vaccinated', 'People_Partially_Vaccinated']].copy()
#vacc_data = vacc_data.merge(vacc_data, dfvac, how='left' , left_on='date', right_on='date')
#Replace NaN Values with Zeros
vacc_data['People_Fully_Vaccinated'] = vacc_data['People_Fully_Vaccinated'].fillna(0)
vacc_data['People_Partially_Vaccinated'] = vacc_data['People_Partially_Vaccinated'].fillna(0)
vacc_data

from plotly.subplots import make_subplots
import plotly.graph_objects as go
fig = make_subplots(rows = 2)
fig.add_scatter(x = vacc_data['Date'], y=vacc_data["People_Fully_Vaccinated"], name = "People Fully Vaccinated")
fig.add_scatter(x = vacc_data['Date'], y=vacc_data["People_Partially_Vaccinated"], name = "People Partially Vaccinated")

```

```

fig.add_scatter(x = vacc_data['Date'], y=vacc_data["People_Partially_Vaccinated"], name = "People_Partially_Vaccinated")
fig.update_layout(
    autosize=False,
    width=700,
    height=700,
    title_text="Vaccination Rates Over Time"
)

fig.update_xaxes(title_text="Date")
fig.update_yaxes(title_text="Number of Vaccinated Individuals", secondary_y=True)
fig.show()

fig1 = sns.lineplot(x=vacc_data['Date'],y=vacc_data["People_Fully_Vaccinated"],
                     label="Fully Vaccinated Rolling Average")
fig2 = sns.lineplot(x=vacc_data['Date'],y=vacc_data["People_Partially_Vaccinated"],
                     label="Partially Vaccinated Rolling Average")
fig1.set_ylabel("Number of Vaccinated Individuals", fontsize = 11)
fig1.set_xticks(range(1, 300, 60), minor = False)
plt.show()

```



```

from sklearn.metrics import mean_absolute_error
def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
def moving_average(series, n):
    return np.average(series[-n:])

## moving average code below pulled from Kaggle - data visualization for time series
def plotMovingAverage(series, window, plot_intervals=False, scale=1.96, plot_anomalies=False):
    rolling_mean = series.rolling(window=window).mean()

    plt.figure(figsize=(15,5))
    plt.title("Moving average\n window size = {}".format(window))
    plt.plot(rolling_mean, "g", label="Rolling mean trend")

    # Plot confidence intervals for smoothed values
    if plot_intervals:
        mae = mean_absolute_error(series[window:], rolling_mean[window:])
        deviation = np.std(series[window:] - rolling_mean[window:])
        lower_bound = rolling_mean - (mae + scale * deviation)
        upper_bound = rolling_mean + (mae + scale * deviation)
        plt.plot(upper_bound, "r--", label="Upper Bound / Lower Bound")
        plt.plot(lower_bound, "r--")

    # Having the intervals, find abnormal values
    if plot_anomalies:
        anomalies = pd.DataFrame(index=series.index, columns=series.columns)
        anomalies[series<lower_bound] = series[series<lower_bound]
        anomalies[series>upper_bound] = series[series>upper_bound]
        plt.plot(anomalies, "ro", markersize=10)

```

```
plt.plot(series[window:], label="Actual values")
plt.legend(loc="upper left")
plt.grid(True)
plotMovingAverage(vacc_data["People_Fully_Vaccinated"], 365*24, plot_intervals=True)
plt.ylabel("Normalized Fully Vaccinated People", fontsize = 13)
plt.xlabel("Time Window (hours)", fontsize = 13)
plotMovingAverage(vacc_data["People_Partially_Vaccinated"], 365*24, plot_intervals=True)
plt.ylabel("Normalized Partially Vaccinated People", fontsize = 13)
plt.xlabel("Time Window (hours)", fontsize = 13)
```

To double-check your work, the cell below will rerun all of the autograder tests.

```
In [ ]: grader.check_all()
```

0.2 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
In [ ]: # Save your notebook first, then run this cell to export your submission.
grader.export()
```