The Politicization of the Pandemic

Group 39

May 2021

1 Introduction

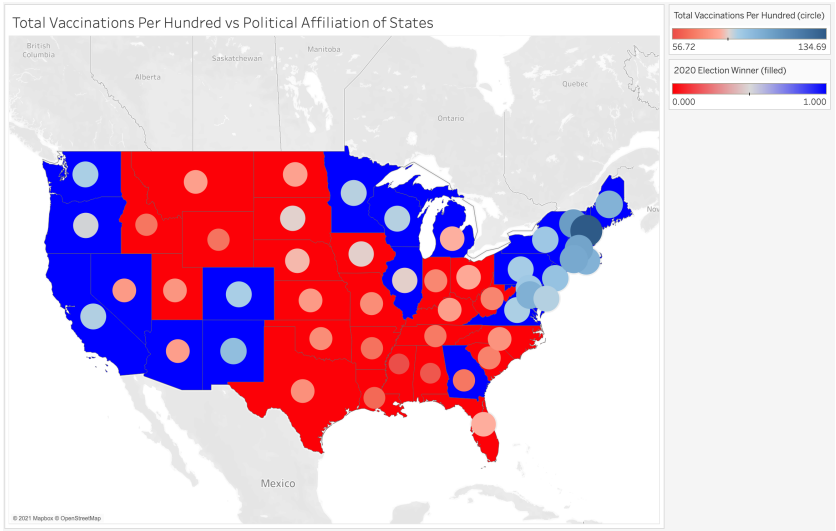
When the COVID-19 pandemic began in early 2020, it quickly became politicized; peoples' views of lockdown orders, mask mandates, social distancing, and more became highly tied to their political views. We saw states' reactions and policies differ greatly, and they were highly dependent on the political affiliations of the governor in charge. Even now, we're seeing reopening plans and policies contrasting in similar ways.

Now, over one year since the pandemic officially began in the United States, we have the data we need to explore questions regarding the effect of the politicization of the pandemic. Our overall goal with these datasets is to analyze and consider the political biases associated with COVID-19. We are exploring a wide variety of factors including tests, cases, deaths, vaccinations, lockdown rules, and how all of these tie in to political affiliation, especially regarding the results of the 2020 election. Our objective is to assess the extent of politicization of COVID-19 and how this bias, if any, affected the course of the pandemic in the United States.

2 Questions

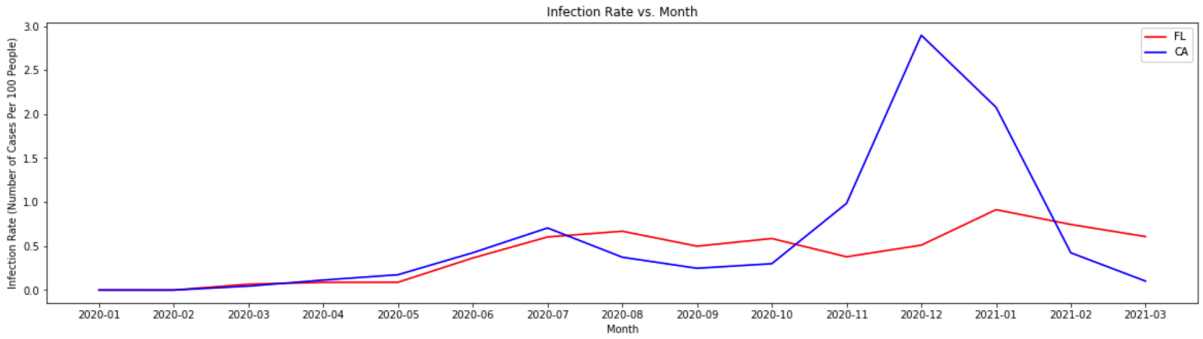
In milestone 1, we saw that:

1. Vaccine allocation (doses received from the government) and percent democratic votes in the 2020 election were not correlated per state. We will statistically show the lack of correlation later using linear regression.
2. Vaccination rate (vaccinations administered per 100 people) and percent democratic votes in the 2020 election were positively correlated per state. To further illustrate the correlation between vaccination rates and election results, we mapped the states colored by election result and then added dots where vaccination rates were colored by dark red (lowest) to dark blue (highest):



and from this visualization we clearly see a correlation where blue states tend to have bluer dots (higher vaccination rates) and red states tend to have redder dots (lower vaccination rates).

3. California saw a more extreme third wave than Florida, despite California having more strict COVID policies and mandates. The severity of California's third wave in comparison to Florida's is clear:



4. COVID death rate (deaths per 100,000 people) and percent democratic votes in the 2020 election were not correlated per state. We will statistically show the lack of correlation later using linear regression.

The first two pieces of information were not surprising – we hypothesized that the federal government would be "fair" in allocating vaccines and that Republican voters would be less likely to choose to receive it. This lines up with our observation that Republican voters tend to be more opposed to social distancing, wearing masks, etc. than Democratic voters.

Based on the first two pieces of information, which helped add validity to our observations, the next two were surprising – we expected more democratic states to see lower case and death rates due to the stricter policies and mandates as well as the stronger adherence to them from the general public. At the same time, we recognized that there are evidently other factors which play a role here, such as demographics, population density, when the COVID spikes occurred, etc.

So, we decided to use data analytical techniques such as linear regression, forecasting, and assumption testing to explore the following questions:

1. Based on states' COVID statistics, can we predict what percentage of their population voted for the democratic candidate in the 2020 Presidential Election?

   → We will explore factors such as COVID case incident rates, death rates, testing rates, vaccination rates, etc. as well as other factors which may play a role such as percentage of population over 60 and population density.

   → We will test the assumptions of our model to determine its validity.

2. Based on the trends of vaccinations rates in different states, can we forecast what they will look like in the coming weeks?

   → About 90% of the vaccines administered are RNA-based from Moderna and Pfizer, which require 2-doses about 3-weeks apart. This "seasonality" of vaccinations allows us to use Holt-Winters and experiment with different parameters to find the best fit.

   → With states and cities introducing new incentives to get the vaccine, we want to explore different possible paths that vaccine administration may take.

3 Can Covid Statistics Predict State Political Results?

In our efforts to understand how the politicization of the pandemic affected the policies, decisions, and outcomes we raised a question:

Given COVID statistics such as

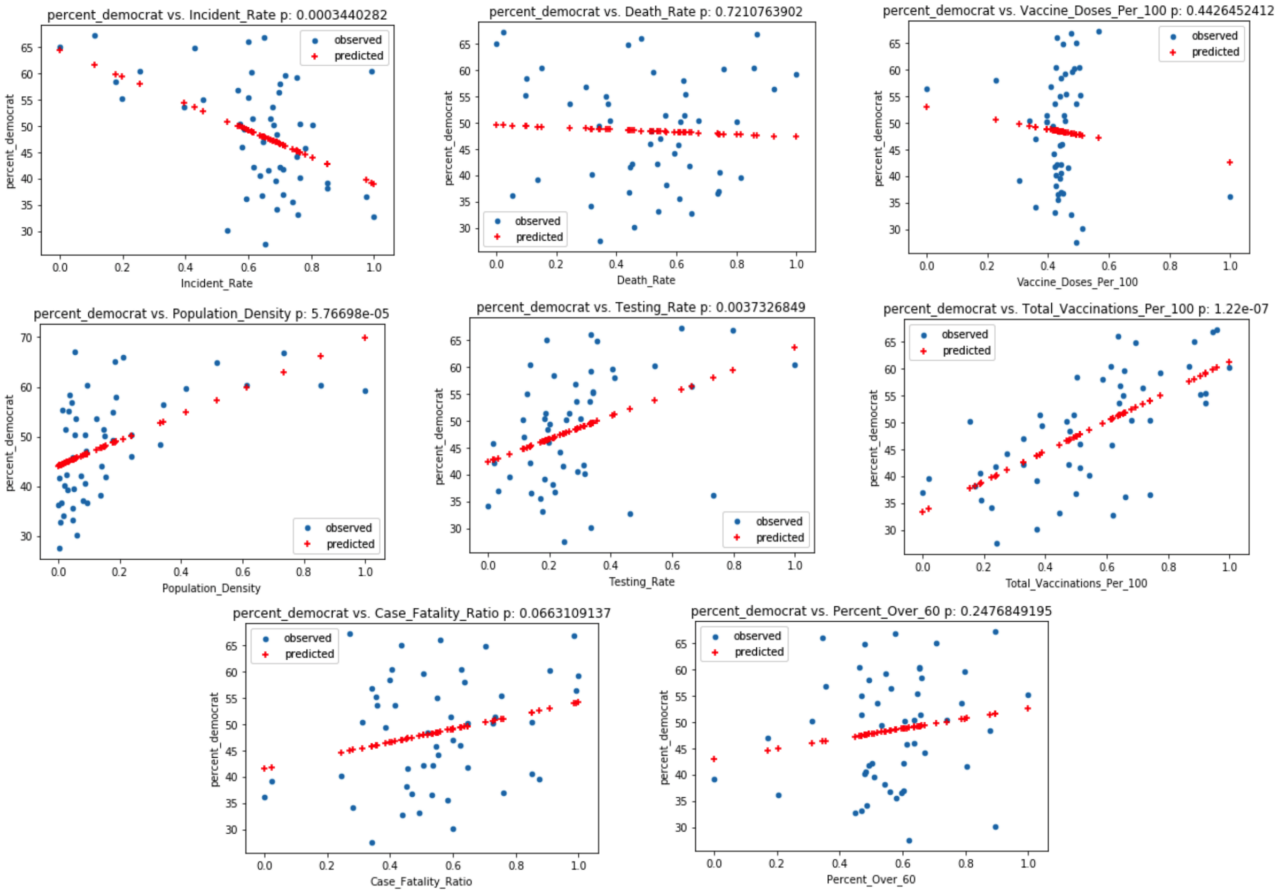
1. Incident Rate – number of COVID cases per 100,000 people
2. Death Rate – number of COVID deaths per 100,000 people
3. Case-Fatality Ratio – number of COVID deaths divided by number of cases
3. Testing Rate – number of COVID tests administered per 100,000 people
4. Vaccination Rate – total vaccinations per 100 people
5. Vaccine Allocation – total vaccine doses allocated by the Federal government per 100 people

and other demographic statistics which may also have an impact on COVID transmission and cases such as

6. Population Density – number of people per square mile
7. Percent Over 60 – percentage of population aged 60+

can we accurately predict the percentage of democratic votes for each state in the 2020 Presidential election?

In other words, for each of these variables, we are exploring whether there is a signficant correlation with political standing. First, we will look at them separately – which variables are correlated with political standing and can be used in a linear regression model to predict? In order to answer this question, for each variable, we first normalized the values and then ran linear regression with percentage democratic votes as the target and plotted the actual vs. predicted results as well as the p-values in the titles:



From these figures, we see the variables that are not significantly correlated (p-value > 0.05) with percentage democratic votes:

1. Death Rate (p-value ≈ 0.721)
2. Vaccine Doses Allocated Per 100 (p-value ≈ 0.443)
3. Percentage Over 60 Years Old (p-value ≈ 0.248)
4. Case Fatality Ratio (p-value ≈ 0.067)

while we see some interesting correlations:

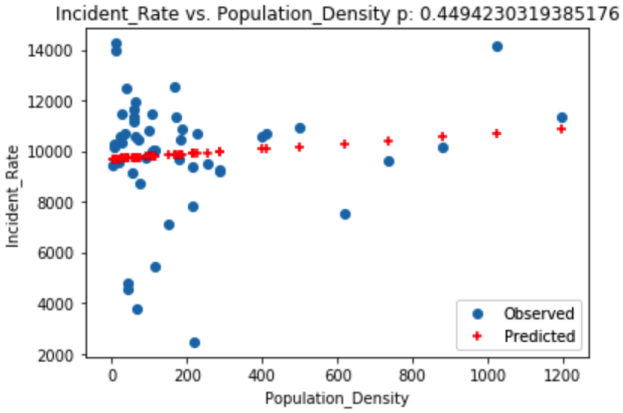
1. States with higher incident rates tend to have lower percentages of democratic votes
2. States with higher population densities tend to have higher percentages of democratic votes
3. States with higher testing rates tend to have higher percentages of democratic votes
4. States with higher vaccination rates tend to have higher percentages of democratic votes

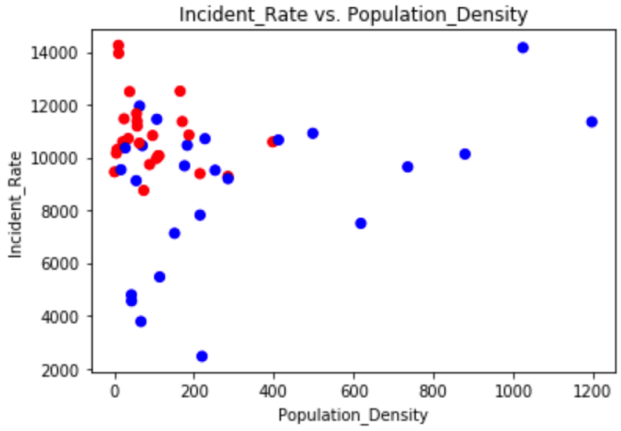
Now, if we move on with four significant correlations found above: incident rate, population density, testing rate, and vaccination rate and perform model selection using Min AIC to predict percent democratic votes using linear regression, we get the following model:

	coef	std err	t	P>|t|	[0.025	0.975]
const	50.8159	4.269	11.904	0.000	42.223	59.409
Incident_Rate	-21.9876	4.812	-4.569	0.000	-31.674	-12.301
Total_Vaccinations_Per_100	14.7257	4.202	3.504	0.001	6.267	23.185
Population_Density	21.0522	4.600	4.577	0.000	11.793	30.311

An important takeaway from running linear regression to predict percentage democratic votes per state is: incident rate and percentage of democratic votes are *negatively* correlated but population and percentage of democratic votes are *positively* correlated.

Logically, we'd expect to see areas with higher population densities have higher incident rates due to the greater risk and ease of transmission, but when we plot COVID incident rates vs. population densities we see a lack of correlation (left chart below):





In fact, when we color states by the Presidential election results (red for Republican, blue for Democrat) we see that the states with the six highest population densities are all democratic (right chart above). Some democratic states with multiples of population densities of other republican states have roughly the same incident rates. And, if we look at states with population densities below 400, we clearly see that republican states tend to have higher incident rates than democratic states on average.

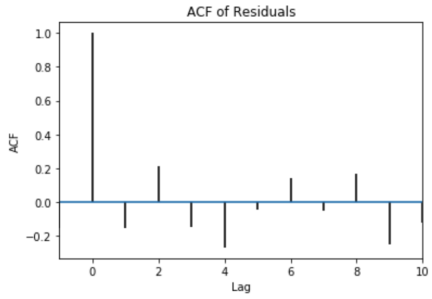
So, why are we seeing this? All we know from these models is the correlation present in the data – this says nothing about cause or the reason for the correlation, but seeing the correlations raises the possibility of stricter lockdowns, policies, more adherence to COVID-safe behaviors like mask-wearing and social distancing, more widespread testing, and greater vaccine acceptance – generally associated with more democratic states – may be playing a role here.

In addition, the fact that we could use the data statistics, such as incident rate and vaccination rate, to predict percentage democratic votes suggests a correlation between a states political affiliation and COVID outcome – i.e., that the politicization of the pandemic might have had an affect on the policies, mandates, and decisions made by states, and therefore also the outcome of the pandemic.

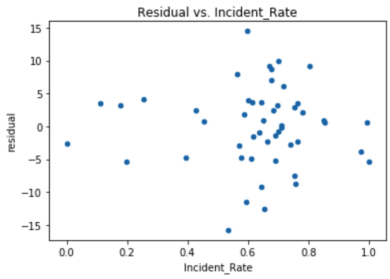
4 Testing Assumptions

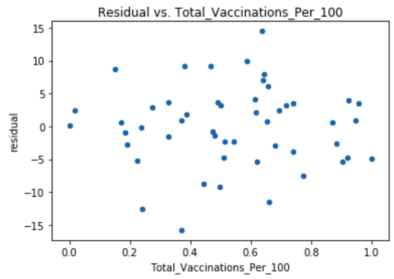
In order for the statistics above to be accurate, some key assumptions must hold:

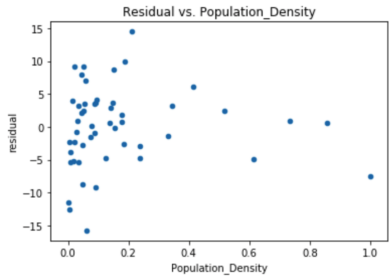
1. Mutual Independence of Residuals: using the auto-correlation function to plot, we see some ACF values approach the test bounds (± 2) but do not exceed it, so we can conclude that the residuals are mutually independent:



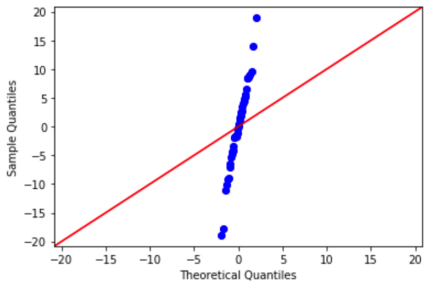
2. Linearity in the Predictors: by plotting the residuals vs. each covariate used in. the model, we see that the the residuals and covariates are independent, which signals linearity in the predictors:



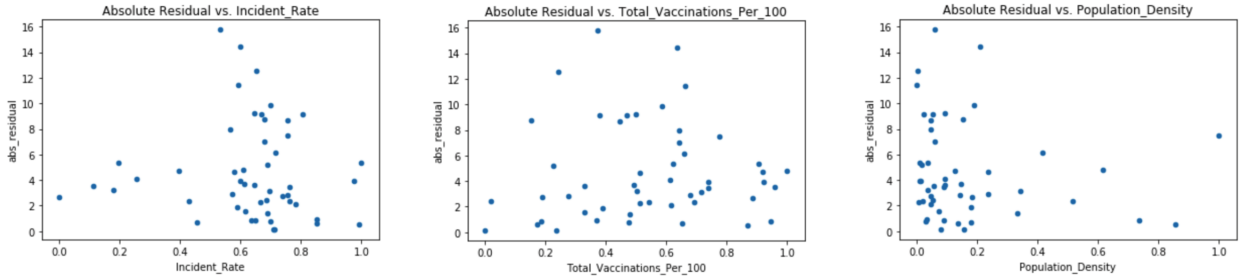




3. Normal Distribution of the Residuals: the Q-Q plot formals a straight line (although not 45 degrees) which signals normal distribution:



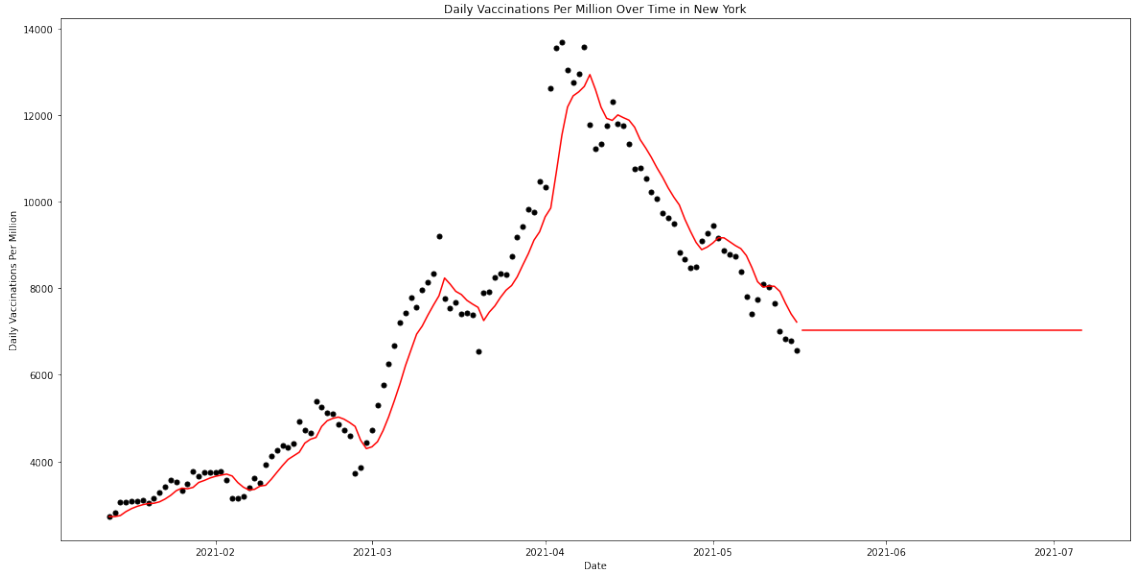
4. Constant Variance: the absolute values of the residuals and covariates are independent, which suggests a constant variance:



Thus, we see that our assumptions for the linear regression model appear to hold up here.

5 Can we forecast vaccination rates?

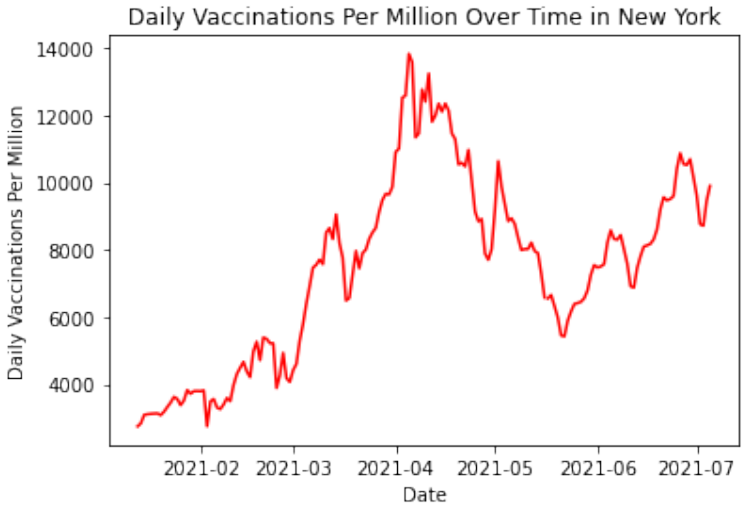
Now that we've looked back at how COVID statistics can retroactively predict election results, our objective is to see how COVID statistics can predict the course of the pandemic in the future. Namely, we wish to predict the rate of vaccinations by state in the coming weeks using current vaccination data. We began with simple exponential smoothing, which we used to fit the vaccination data for the state of New York. Then, we used this model to forecast the vaccinations in New York over the next 50 days. This time period was chosen as a reasonable period to forecast vaccinations into the future. It isn't too short such that we aren't able to properly evaluate the forecast nor is it too long such that the extrapolation becomes inaccurate due to lack of data.



Thus, we see that although simple exponential smoothing fits the data quite well, the forecast is flat at the most recent vaccination rate, so this model may not be the best predictor of vaccinations. It is indeed possible that vaccination rates could stay stagnant over the coming weeks as people continue to get them at a lesser frequency than when they first became widely available. It has been some time since all adults in New York became eligible, so this could lead one to conclude that the majority of people that want the vaccine have either gotten it or have begun the process. However, we thought exploring other forecasting methods would be worthwhile.

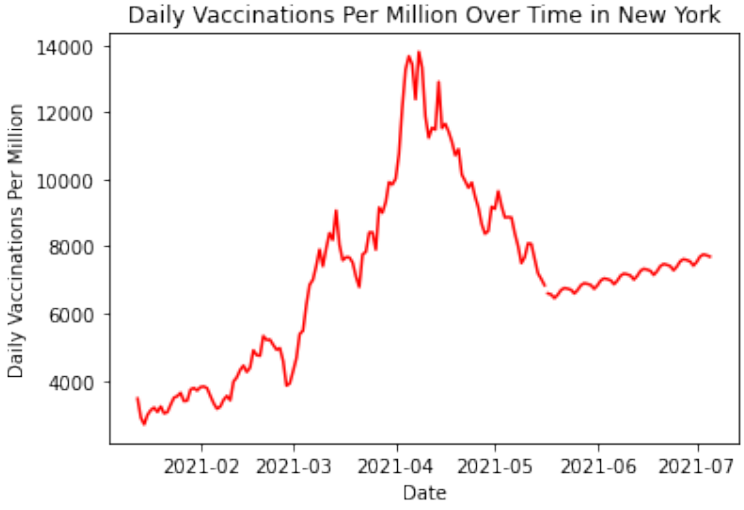
Since Pfizer and Moderna constitute the majority of vaccinations in the US, we considered incorporating a "vaccination cycle" into our model to account for the time in between the two doses. We chose Pfizer's three weeks

over Moderna's four weeks to represent this since Pfizer is used a bit more overall, but the choice is mostly arbitrary since one week is not likely to make much of a difference. Based on this three week "vaccination cycle," it seemed natural to try the Holt-Winters method, which allows for defining a period/season. Thus, we used the Holt-Winters method where our trend and seasonality are both multiplicative and our seasonal periods value is 21 days. For New York state, our results of a forecast for the next 50 days are the following.



Perhaps surprisingly, this forecast predicts that daily vaccinations will bounce back in late May and continue increasing well into the summer. Some potential explanations for this change include the fact that ages 12-15 were recently approved for the Pfizer vaccine, many states are using various incentives to increase vaccinations such as a lottery in Ohio, and those that were hesitant are beginning to come around after seeing cases and deaths fall as well as mask/distancing restrictions being reduced and/or removed for vaccinated people by the CDC.

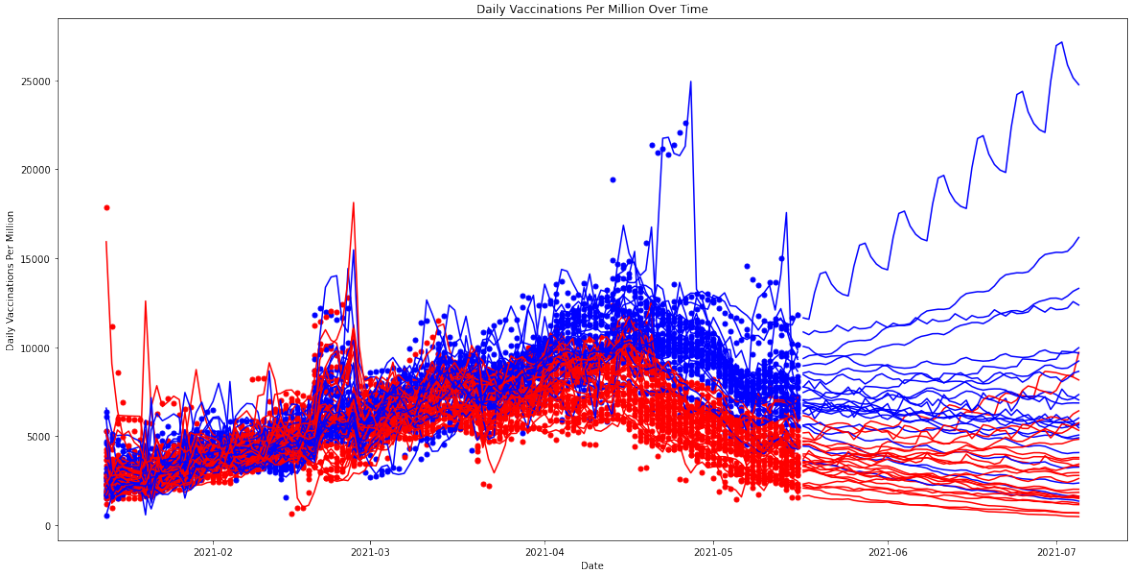
Since we chose our trend, seasonality, and period somewhat arbitrarily in the previous model, we decided to build a Holt-Winters model with the least mean squared error by testing every combination of trend (additive vs multiplicative), seasonality (additive vs multiplicative), and period (2-31 days). We found that a model with an additive trend, multiplicative seasonality, and a period of 6 (approximately representing a week) had the minimum error. Thus, we forecasted the vaccination rate in New York over the next 50 days using this model.



Here, we see vaccinations turning around and increasing as we go into the summer, but not as intensely as the previous model. The forecast also has a wavy shape, which reflects our 6 day period and indicates that some days

of the week (such as the weekend) are more popular than others to get vaccinated as people may have more free time on these days. While our previous two models represented two extremes (vaccinations stay the same vs vaccinations increase significantly), this one is in between. This may be because the reasons we speculated for the first model and the reasons for the second model are *all* in effect. Thus, there are forces that may cause vaccinations to stay stagnant or decrease (US is past vaccine peak so we're now dealing with more hesitant folks) and forces that may cause them to increase (more eligibility, easier availability, incentives, seeing the pandemic slowly ending). This model seems to represent the compilation of these forces quite well.

Finally, we created a model that we believe to be the most realistic by using elements from the previous three models. Contrary to the previous model, we decided it would make sense to have a multiplicative trend rather than additive because in the real world, vaccination is a compounding effect where as more and more people make their decisions regarding vaccination, others are increasingly influenced by these voices and make their own decisions. We agree with the previous model in keeping a multiplicative seasonality and a period of about a week (this time we used exactly 7 days instead of 6). We used this model to predict vaccination rates for every state for the next 50 days, colored by political affiliation and compiled into one graph.



This model forecasts that, in general, vaccination rates in blue states will either increase or remain around the same and that vaccination rates in red states will either decrease or remain stagnant. This is an interesting result because it aligns with what we saw through our explorations in the first milestone. Thus, we believe that this forecast is quite plausible and may even roughly match reality as we head into the summer. It does a great job capturing the forces that may cause vaccinations to increase or decrease as well as which forces will be more prominent in which states. As summer approaches, we look forward to seeing how accurate our model proves to be. Although unlikely, it is our hope that vaccinations will increase in all states as we all long for a return to normalcy through vaccination after more than a year of struggle and hardship.

Sources

CSSEGISandData. (2021, April) CSSE Covid 19 Daily Reports US 04-26-2021. Retrieved April 26,2021 from https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_daily_reports_us/04-26-2021.csv

CDC Case Surveillance Task Force. (2021, April). COVID-19 Case Surveillance Public Use Data with Geography. Retrieved April 25, 2021 from https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data-with-Ge/n8mc-b4w4

CDC Case Surveillance Task Force. (2021, April). COVID-19 Case Surveillance Public Use Data. Retrieved April 25, 2021 from https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data/vbim-akqf

Johns Hopkins University. (2021, April). Death Rates from Coronavirus (COVID-19) in the USA, by State. Retrieved April 25, 2021 from https://www.statista.com/statistics/1109011/coronavirus-covid19-death-rates-us-by-state/

HHS ASPA. (2021, April). COVID-19 Vaccine Distribution Allocations by Jurisdiction - Moderna. Retrieved April 25, 2021 from https://data.cdc.gov/Vaccinations/COVID-19-Vaccine-Distribution-Allocations-by-Juris/b7pe-5nws

HHS ASPA. (2021, April). COVID-19 Vaccine Distribution Allocations by Jurisdiction - Pfizer. Retrieved April 25, 2021 from https://data.cdc.gov/Vaccinations/COVID-19-Vaccine-Distribution-Allocations-by-Juris/saz5-9hgg

HHS ASPA. (2021, April). COVID-19 Vaccine Distribution Allocations by Jurisdiction - Janssen. Retrieved April 25, 2021 from https://data.cdc.gov/Vaccinations/COVID-19-Vaccine-Distribution-Allocations-by-Juris/w9zu-fywh

Mooney, P. (2021, April). USA COVID-19 Vaccinations, Version 65. Retrieved April 25, 2021 from https://www.kaggle.com/paultimothymooney/usa-covid19-vaccinations

Mooney, P. (2020, November). Percent Voting for Democratic Party by State, Version 5. Retrieved April 25, 2021 from https://www.kaggle.com/paultimothymooney/percent-voting-for-democratic-party-by-state

Vikas. (2018, December) .US State Populations – 2018, Version 1. Retrieved April 25, 2021 from https://www.kaggle.com/lucasvictor/us-state-populations-2018

Appendix Code

May 21, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
import seaborn
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
import statsmodels.api as sm
from sklearn import datasets
from datetime import datetime
```

[2]:
```
import warnings; warnings.simplefilter('ignore')
```

These next few cells are just putting all of the datasets into one in order to use for linear regression

[3]:
```
abbrevs = pd.read_csv('abbrevs.csv')
johns = pd.read_csv('johns_recent.csv')
percs = pd.read_csv('democratic_vs_republican_votes_by_usa_state_2020.csv')
pfizer = pd.
 ↪read_csv('COVID-19_Vaccine_Distribution_Allocations_by_Jurisdiction_-_Pfizer.
 ↪csv')
moderna = pd.
 ↪read_csv('COVID-19_Vaccine_Distribution_Allocations_by_Jurisdiction_-_Moderna.
 ↪csv')
janssen = pd.
 ↪read_csv('COVID-19_Vaccine_Distribution_Allocations_by_Jurisdiction_-_Janssen.
 ↪csv')
vac = pd.read_csv('us_state_vaccinations.csv')
states = pd.read_csv('abbrevs.csv')
demos=pd.read_csv('demographics.csv')
areas = pd.read_csv('state_sizes.csv')
```

[4]:
```
areas = areas.rename({'Square Miles (Land Area)': 'Area'}, axis=1)
```

[5]:
```
johns = johns.join(areas.set_index('State'), on='Province_State', how='inner')
johns = johns.reset_index()
```

```
johns = johns.drop('index', axis=1)
```

[6]:
```
demos = demos.groupby(['NAME', 'AGE']).sum().reset_index()
demos = demos[['NAME', 'AGE', 'POPEST2019_CIV']]
demos = demos.rename({'POPEST2019_CIV':'POP'}, axis=1)
demos = demos[(demos['AGE'] >= 60) & (demos['AGE'] != 999)]
demos = demos.groupby('NAME').sum()['POP']
demos = demos.reset_index()
```

[7]:
```
johns = johns.join(demos.set_index('NAME'), on='Province_State', how='inner')
johns = johns.rename({'POP':'Population Over 60'}, axis=1)
```

[8]:
```
vac = vac[vac['date'] == '2021-04-25'].dropna(axis=0)
vac = vac.reset_index()
vac = vac.drop('index', axis=1)
```

[9]:
```
johns['Population'] = 100000 * johns['Confirmed'] / johns['Incident_Rate']
johns['Death_Rate'] = 100000 * johns['Deaths'] / johns['Population']
```

[10]:
```
johns = johns.join(percs.set_index('state'), on='Province_State', how='inner')
johns = johns.reset_index()
```

[11]:
```
vac = vac.replace('New York State', 'New York')
```

[12]:
```
johns = johns.join(vac.set_index('location'), on='Province_State', how='inner')
```

[13]:
```
johns = johns.reset_index()
```

[14]:
```
johns = johns.drop(['level_0', 'index', 'Country_Region', 'Recovered',␣
↪'Active', 'usa_state', 'UID', 'ISO3', 'Hospitalization_Rate',␣
↪'People_Hospitalized', 'DEM', 'REP'], axis=1)
```

[15]:
```
pfizer = pfizer.groupby('Jurisdiction').sum()
pfizer = pfizer.rename({'1st Dose Allocations': 'Pfizer_1', '2nd Dose␣
↪Allocations': 'Pfizer_2'}, axis=1)
pfizer = pfizer.reset_index()
```

[16]:
```
moderna = moderna.groupby('Jurisdiction').sum()
moderna = moderna.rename({'1st Dose Allocations': 'Moderna_1', '2nd Dose␣
↪Allocations': 'Moderna_2'}, axis=1)
moderna = moderna.reset_index()
```

[17]:
```
janssen = janssen.groupby('Jurisdiction').sum()
janssen = janssen.rename({'1st Dose Allocations': 'Janssen_1'}, axis=1)
janssen = janssen.reset_index()
```

[18]:
```
allocs=pfizer.join(moderna.set_index('Jurisdiction'), on='Jurisdiction',␣
↪how='inner')
allocs=allocs.join(janssen.set_index('Jurisdiction'), on='Jurisdiction',␣
↪how='inner')
```

[19]:
```
johns=johns.join(allocs.set_index('Jurisdiction'), on='Province_State',␣
↪how='inner')
```

[20]:
```
vac_alloc = johns['Pfizer_1'] + johns['Pfizer_2'] + johns['Moderna_1'] +␣
↪johns['Moderna_2'] + johns['Janssen_1']
johns['Vaccine_Doses_Per_100'] = 100.0 * vac_alloc / johns['Population']
johns['Percent_Pfizer'] = 100.0 * johns['Pfizer_1'] / (johns['Pfizer_1'] +␣
↪johns['Moderna_1'] + johns['Janssen_1'])
johns['Percent_Moderna'] = 100.0 * johns['Moderna_1'] / (johns['Pfizer_1'] +␣
↪johns['Moderna_1'] + johns['Janssen_1'])
johns['Percent_Janssen'] = 100.0 * johns['Janssen_1'] / (johns['Pfizer_1'] +␣
↪johns['Moderna_1'] + johns['Janssen_1'])
```

[21]:
```
johns['Pfizer_1'] = 100.0 * johns['Pfizer_1'] / johns['Population']
johns['Pfizer_2'] = 100.0 * johns['Pfizer_2'] / johns['Population']
johns['Moderna_1'] = 100.0 * johns['Moderna_1'] / johns['Population']
johns['Moderna_2'] = 100.0 * johns['Moderna_2'] / johns['Population']
johns['Janssen_1'] = 100.0 * johns['Janssen_1'] / johns['Population']
```

[22]:
```
johns['Can_Vaccinate_Per_100'] = johns['Pfizer_1'] + johns['Moderna_1'] +␣
↪johns['Moderna_1']
```

[23]:
```
johns = johns.drop(['Last_Update', 'FIPS', 'usa_state_code', 'date'], axis=1)
johns = johns.rename({'Long_': 'Long'} ,axis=1)
```

[24]:
```
abbrevs = abbrevs[['State', 'Winner']]
johns = johns.join(abbrevs.set_index('State'), on='Province_State', how='inner')
```

[25]:
```
johns['Percent_Over_60'] = 100.0 * johns['Population Over 60'] /␣
↪johns['Population']
johns = johns.rename({'total_vaccinations_per_hundred':␣
↪'Total_Vaccinations_Per_100'}, axis=1)
```

[26]:
```
johns['Population_Density'] = johns['Population'] / johns['Area']
```

[27]:
```
johns.head()
```

[27]:

	Province_State	Lat	Long	Confirmed	Deaths	Incident_Rate	\
0	Alabama	32.3182	-86.9023	531404	10985	10837.934930	
1	Alaska	61.3707	-152.4044	69178	351	9456.424417	
2	Arizona	33.7298	-111.4312	870155	17428	11954.785440	

```
3       Arkansas  34.9697  -92.3731     337819    5770   11194.199490
4     California  36.1162 -119.6816    3761779   62365    9520.545073

   Total_Test_Results  Case_Fatality_Ratio  Testing_Rate  Unnamed: 18  …  \
0           2531623.0             2.067165   51632.21457          NaN  …
1           2169062.0             0.507387  296504.24790          NaN  …
2           9366325.0             2.002862  128680.98870          NaN  …
3           3000603.0             1.708015   99430.01600          NaN  …
4          62051055.0             1.657859  157042.68270          NaN  …

   Moderna_2  Janssen_1  Vaccine_Doses_Per_100  Percent_Pfizer  \
0  17.500869   3.261146              78.001136       48.901226
1  25.493989   4.852743             118.428805       51.398923
2  16.711736   3.109064              74.489227       48.914337
3  17.247641   3.217571              77.206472       49.106731
4  17.322741   3.221788              76.985494       48.771413

   Percent_Moderna  Percent_Janssen  Can_Vaccinate_Per_100  Winner  \
0        43.072552         8.026222              54.870865       r
1        40.829283         7.771794              83.081697       r
2        43.072434         8.013229              52.401818       b
3        42.891753         8.001516              54.242091       r
4        43.194933         8.033654              54.204594       b

   Percent_Over_60  Population_Density
0        47.772091           96.614483
1        37.374051            1.281971
2        47.884565           64.049533
3        47.176954           57.951109
4        41.080655          253.327326

[5 rows x 41 columns]
```

At this point, 'johns' has all of the information we need to explore linear regression.

Below we're just visualizing the data by plotting it and coloring by state election results

[92]:
```
johns.head(2)
```

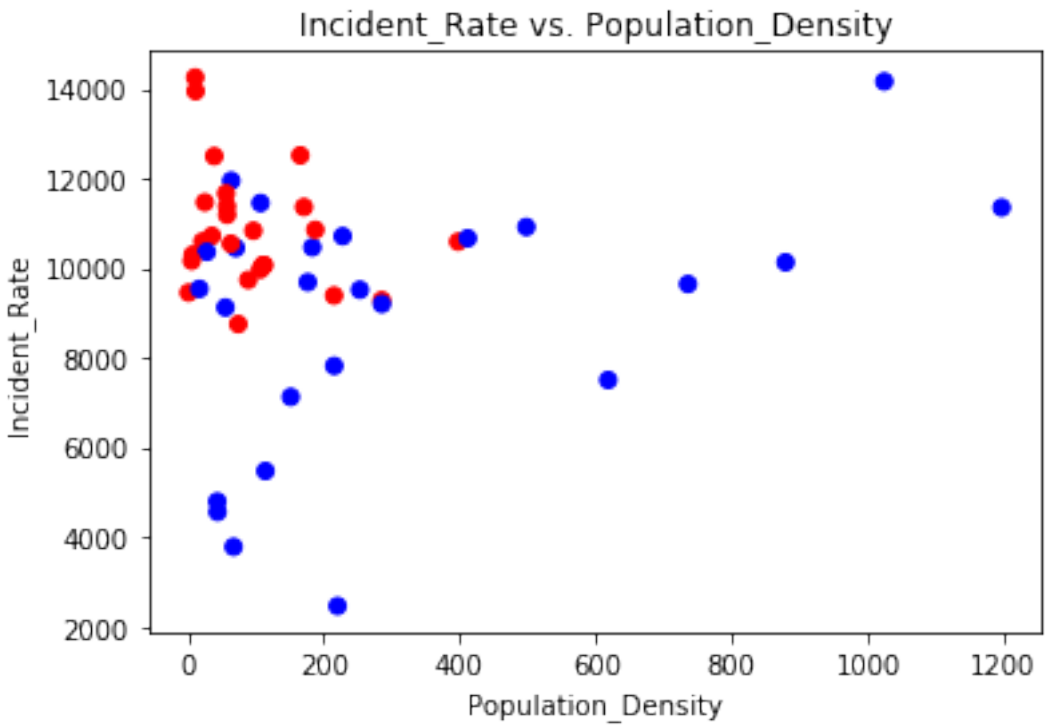
[92]:
```
  Province_State      Lat      Long  Confirmed  Deaths  Incident_Rate  \
0        Alabama  32.3182  -86.9023     531404   10985   10837.934930
1         Alaska  61.3707 -152.4044      69178     351    9456.424417

   Total_Test_Results  Case_Fatality_Ratio  Testing_Rate  Unnamed: 18  …  \
0           2531623.0             2.067165   51632.21457          NaN  …
1           2169062.0             0.507387  296504.24790          NaN  …
```

```
   Moderna_2  Janssen_1  Vaccine_Doses_Per_100  Percent_Pfizer  \
0  17.500869   3.261146              78.001136       48.901226
1  25.493989   4.852743             118.428805       51.398923

   Percent_Moderna  Percent_Janssen  Can_Vaccinate_Per_100  Winner  \
0        43.072552         8.026222              54.870865       r
1        40.829283         7.771794              83.081697       r

   Percent_Over_60  Population_Density
0        47.772091           96.614483
1        37.374051            1.281971

[2 rows x 41 columns]
```

[70]:
```
var = 'Population_Density'
target = 'Incident_Rate'
x = sm.add_constant(johns[var])
Y = johns[target]

model = sm.OLS(Y, x).fit()
Yhat = model.predict(x)
print(model.pvalues[var])

plt.scatter(x=johns[var], y=johns[target], color=list(johns['Winner'].
 ↪reset_index()['Winner']), label="Observed")
#plt.scatter(x=johns[var], y=Yhat, marker='+', label='Predicted')
plt.xlabel(var)
plt.ylabel(target)
plt.title(target + ' vs. ' + var)
#plt.legend()
```

```
0.4494230319385176
```

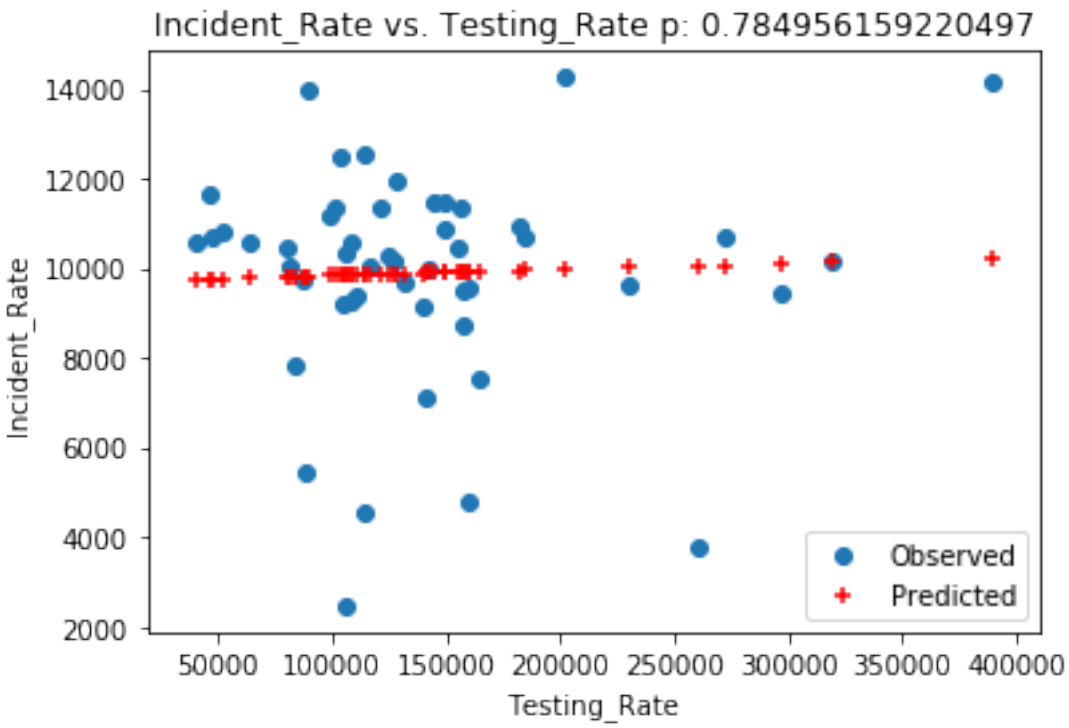
[70]:
```
Text(0.5, 1.0, 'Incident_Rate vs. Population_Density')
```



[135]:
```
var = 'Testing_Rate'
target = 'Incident_Rate'
x = sm.add_constant(johns[var])
Y = johns[target]

model = sm.OLS(Y, x).fit()
Yhat = model.predict(x)
print(model.pvalues[var])

plt.scatter(x=johns[var], y=johns[target], label="Observed")
plt.scatter(x=johns[var], y=Yhat, marker='+', color='r', label='Predicted')
plt.xlabel(var)
plt.ylabel(target)
plt.title(target + ' vs. ' + var + ' p: ' + str(model.pvalues[var]))
plt.legend()
```

```
0.784956159220497
```

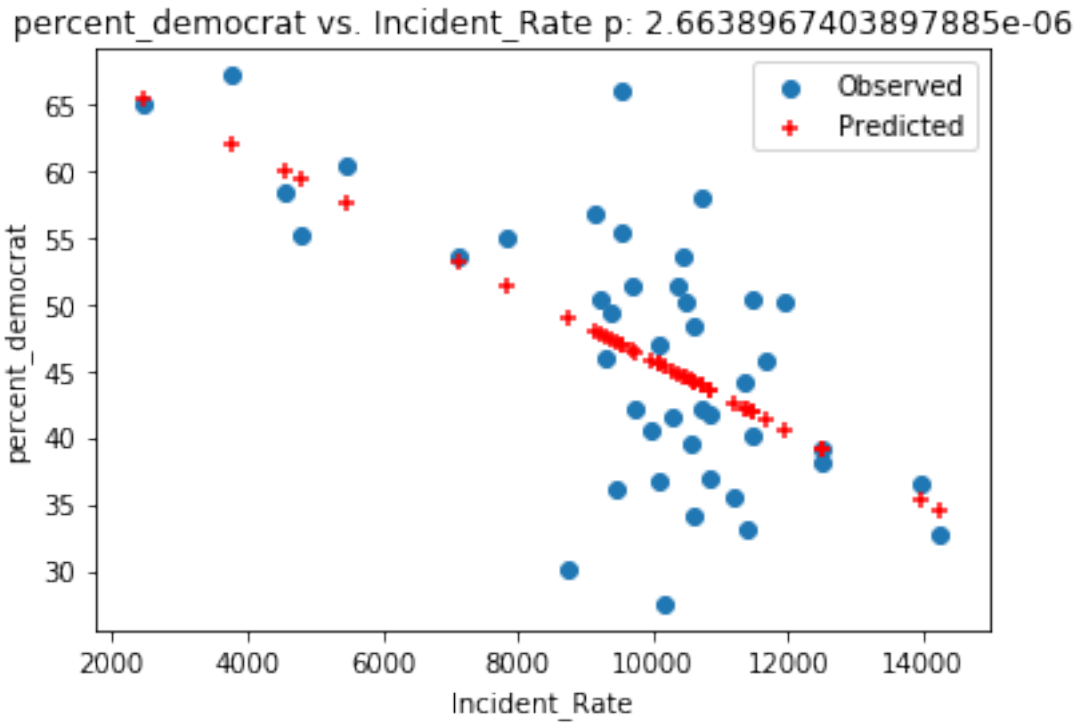
[135]:
```
<matplotlib.legend.Legend at 0x7f82e2aac610>
```



[93]:
```
j_400 = johns[johns['Population_Density'] < 400]

var = 'Incident_Rate'
target = 'percent_democrat'
x = sm.add_constant(j_400[var])
Y = j_400[target]

model = sm.OLS(Y, x).fit()
Yhat = model.predict(x)
print(model.pvalues[var])

plt.scatter(x=j_400[var], y=j_400[target], label="Observed")
plt.scatter(x=j_400[var], y=Yhat, marker='+', color='r', label='Predicted')
plt.xlabel(var)
plt.ylabel(target)
plt.title(target + ' vs. ' + var + ' p: ' + str(model.pvalues[var]))
plt.legend()
```

```
2.6638967403897885e-06
```

[93]: <matplotlib.legend.Legend at 0x7f82d0e08b50>



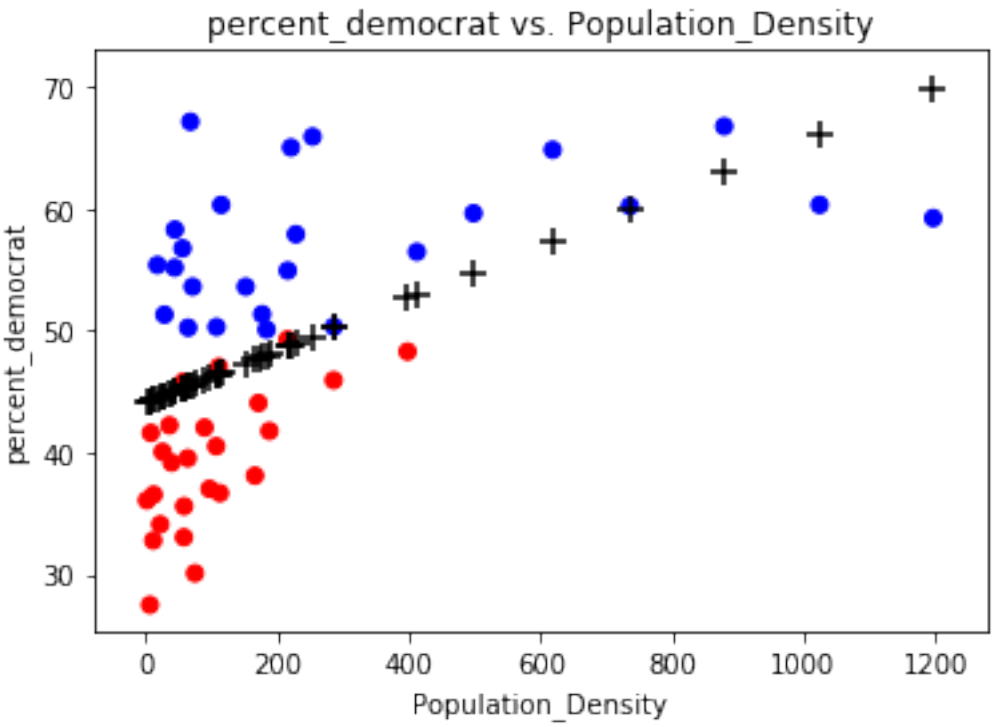
[ ]:
```
var = 'Population_Density'
target = 'Incident_Rate'
x = sm.add_constant(johns[var])
Y = johns[target]

model = sm.OLS(Y, x).fit()
Yhat = model.predict(x)
print(model.pvalues[var])

plt.scatter(x=johns[var], y=johns[target], label="Observed")
plt.scatter(x=johns[var], y=Yhat, marker='+', color='r', label='Predicted')
plt.xlabel(var)
plt.ylabel(target)
plt.title(target + ' vs. ' + var + ' p: ' + str(model.pvalues[var]))
plt.legend()
```

[30]:
```
var = 'Population_Density'
target = 'percent_democrat'
x = sm.add_constant(johns[var])
Y = johns[target]

model = sm.OLS(Y, x).fit()
Yhat = model.predict(x)
```

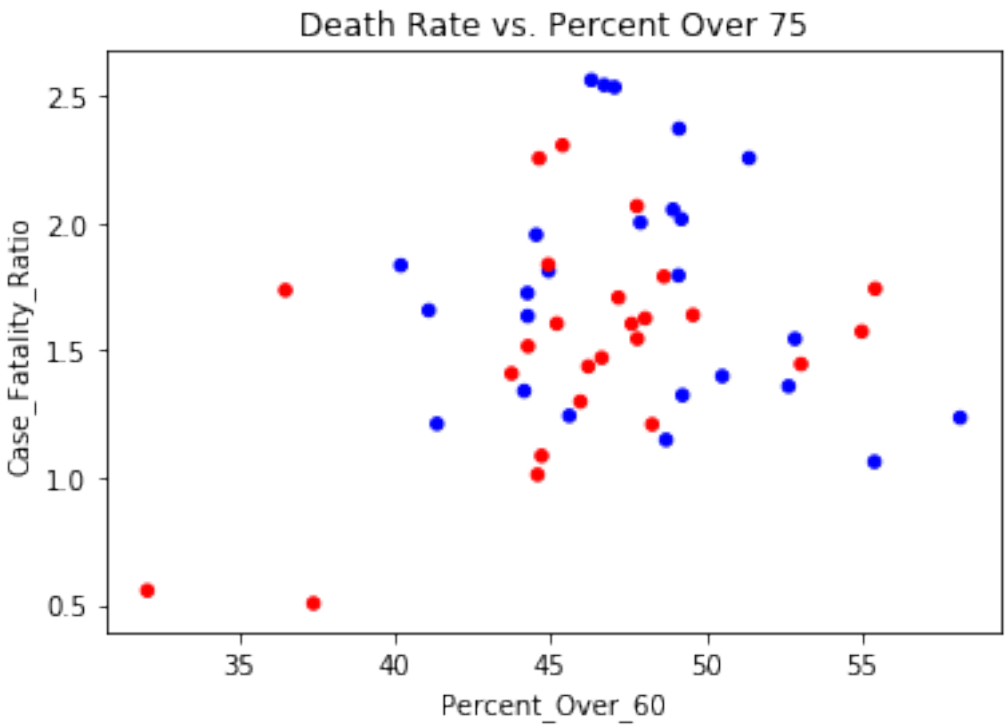
```
print(model.pvalues[var])

plt.scatter(x=johns[var], y=johns[target], color=list(johns['Winner'].
 ↪reset_index()['Winner']))
plt.scatter(x=johns[var], y=Yhat, marker='+', color='black', s=100)
plt.xlabel(var)
plt.ylabel(target)
plt.title(target + ' vs. ' + var)
```

```
5.7669816469415134e-05
```

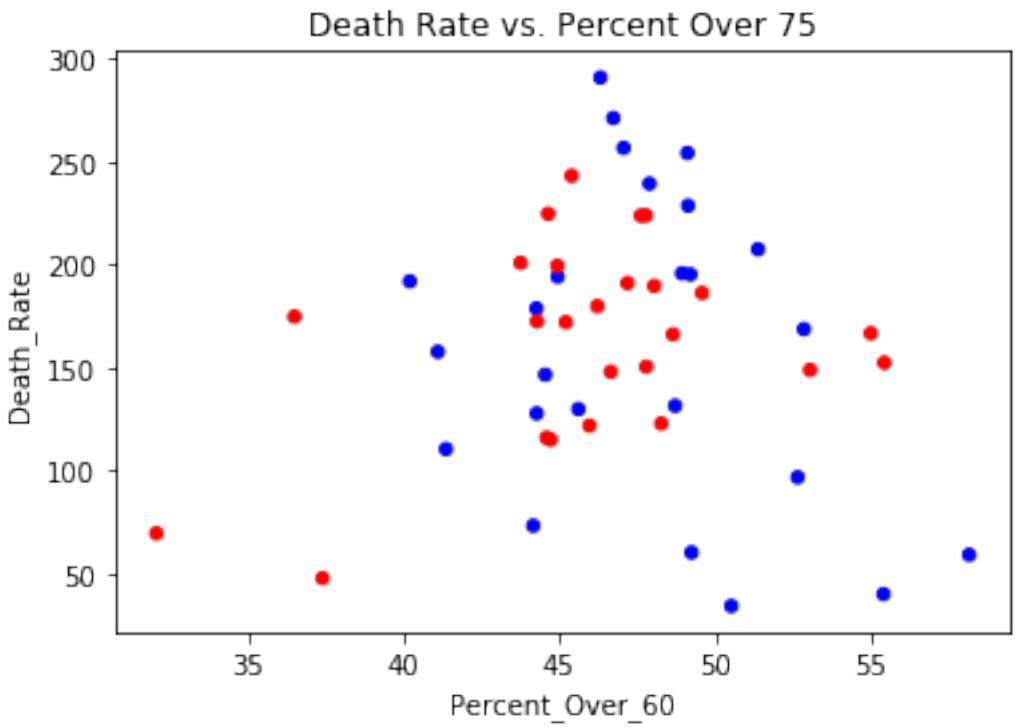
[30]:
```
Text(0.5, 1.0, 'percent_democrat vs. Population_Density')
```



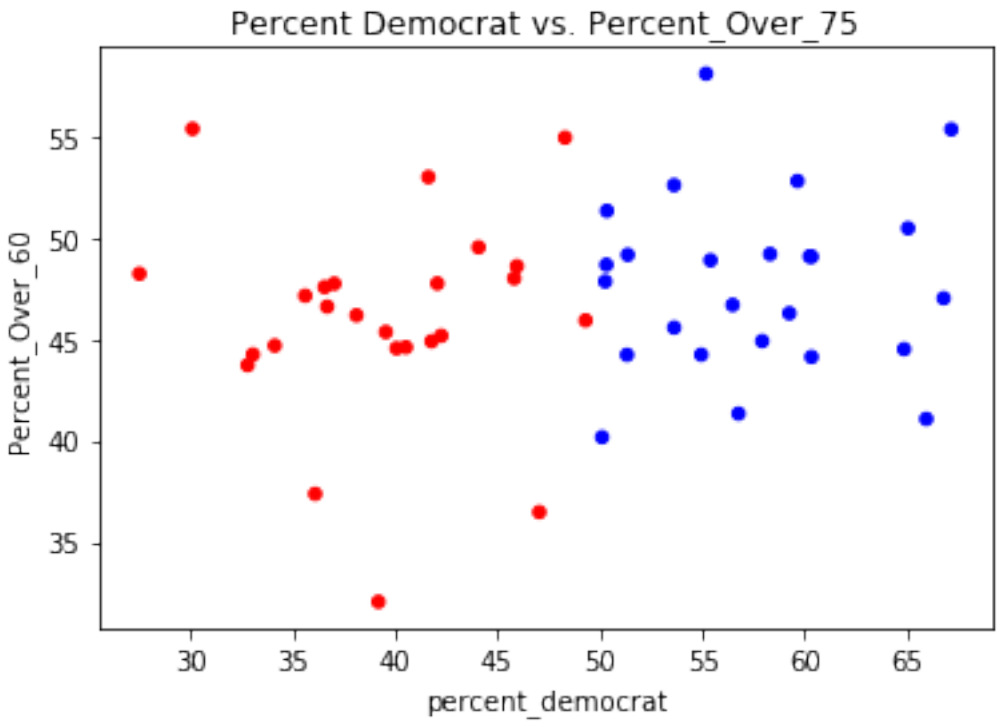
[31]:
```
p = johns.plot.scatter(x='Percent_Over_60', y='Case_Fatality_Ratio',␣
 ↪color=list(johns['Winner'].reset_index()['Winner']), title='Death Rate vs.␣
 ↪Percent Over 75')
```



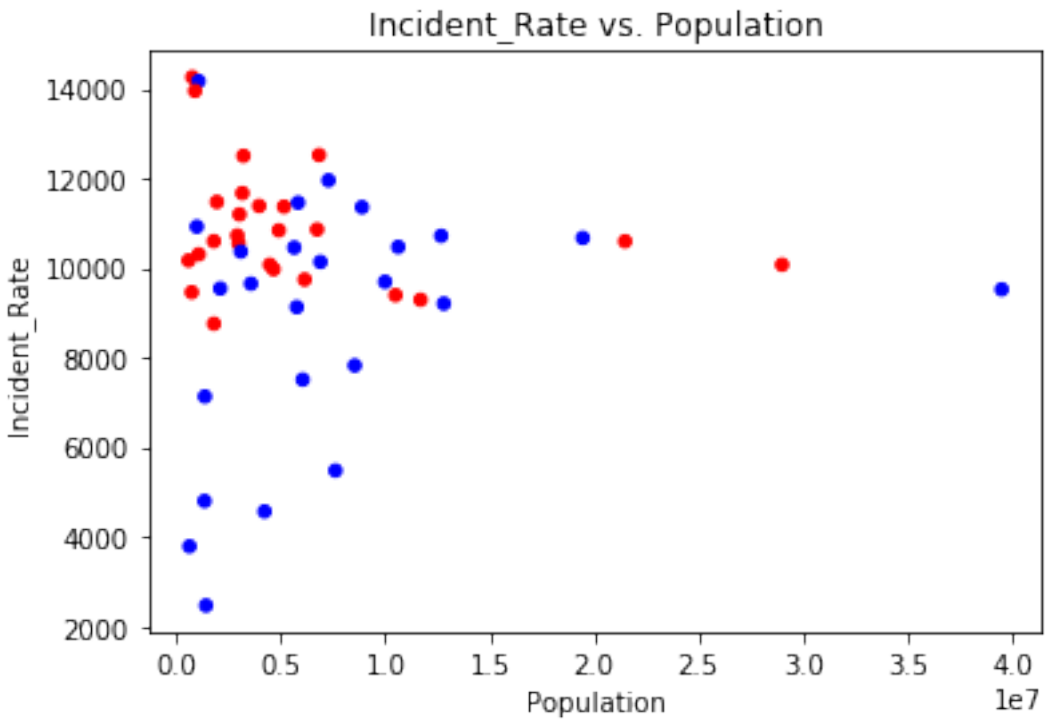
[32]:
```
p = johns.plot.scatter(x='Percent_Over_60', y='Death_Rate',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Death Rate vs.␣
↪Percent Over 75')
```



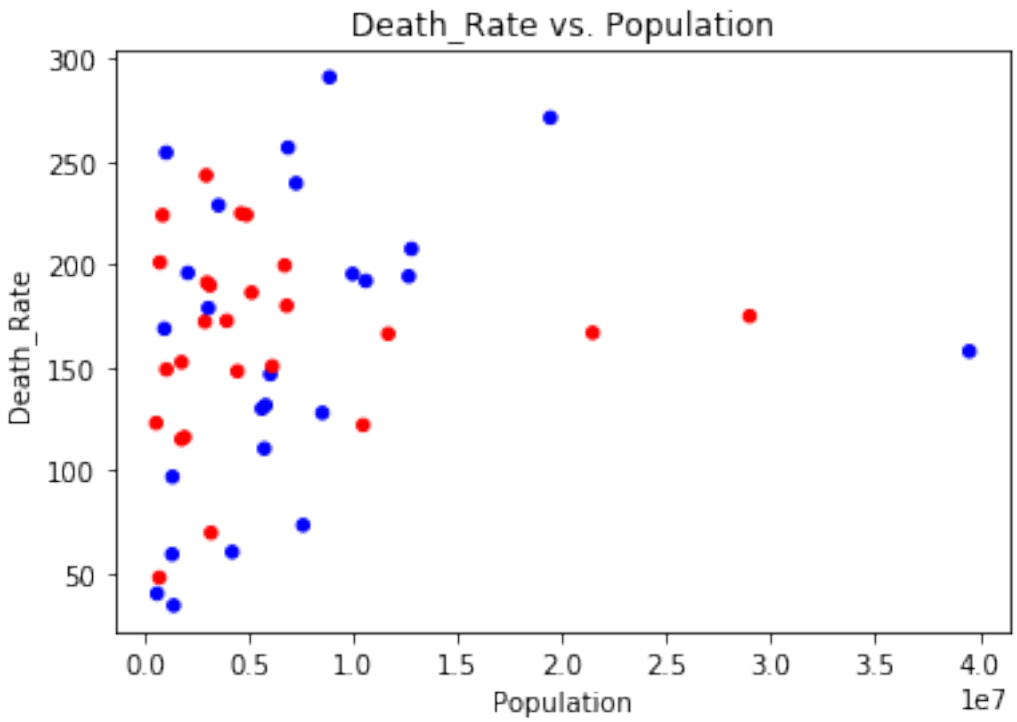
[33]:
```
p = johns.plot.scatter(x='percent_democrat', y='Percent_Over_60',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Percent Democrat␣
↪vs. Percent_Over_75')
```



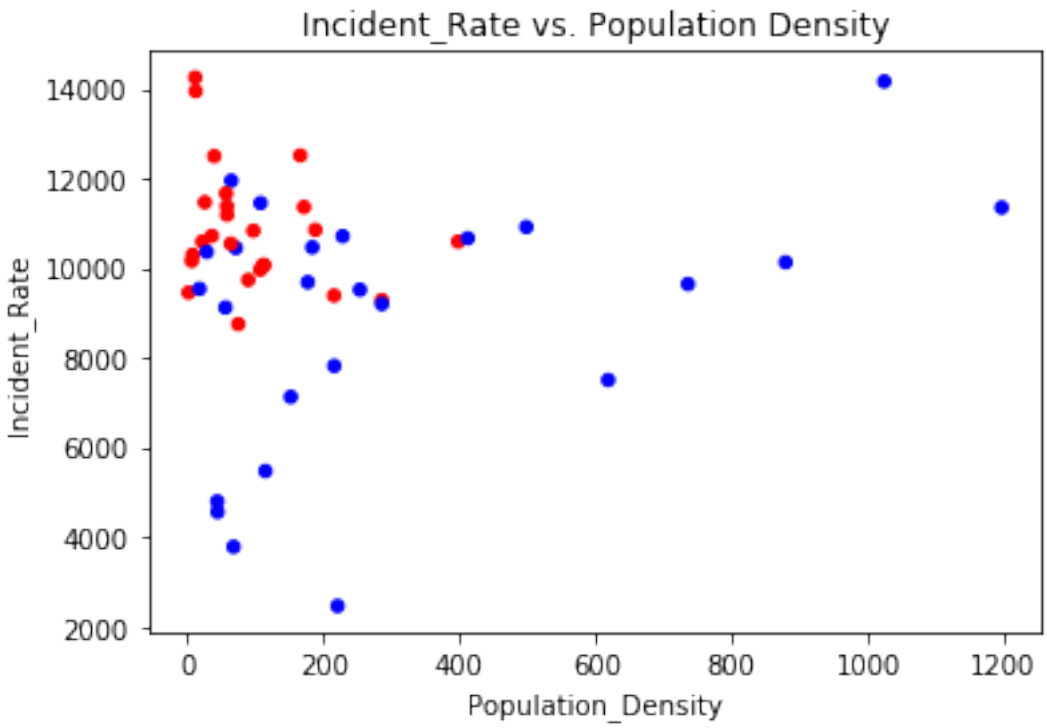
[34]:
```
p = johns.plot.scatter(x='Population', y='Incident_Rate',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Incident_Rate vs.
↪ Population')
```



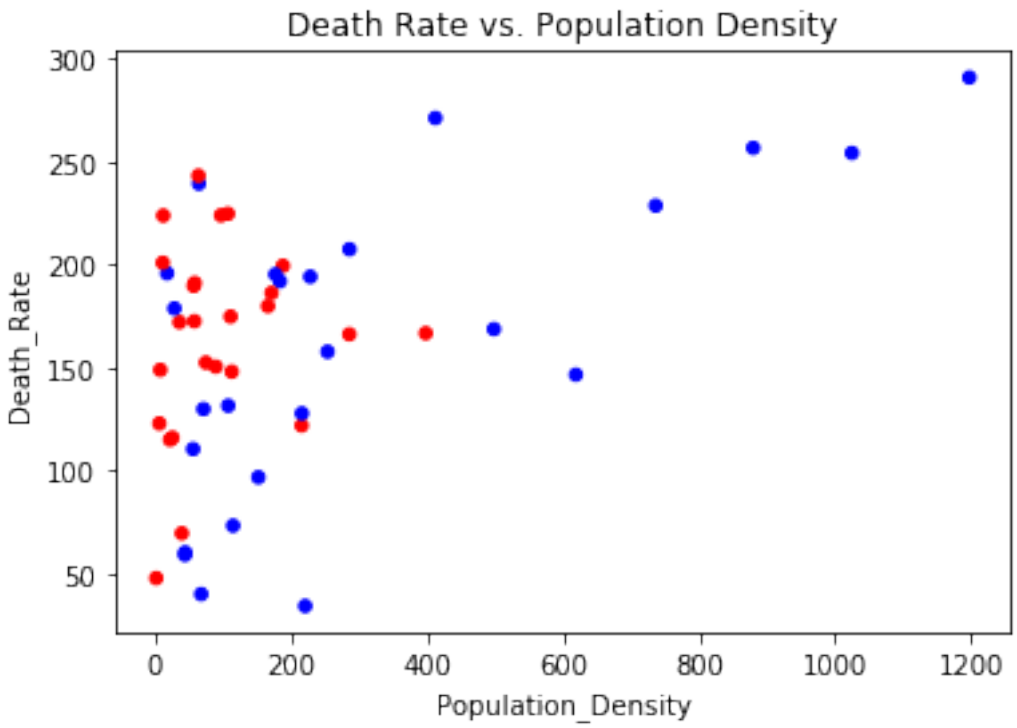
[35]:
```
p = johns.plot.scatter(x='Population', y='Death_Rate',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Death_Rate vs.␣
↪Population')
```



[36]:
```
p = johns.plot.scatter(x='Population_Density', y='Incident_Rate',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Incident_Rate vs.
↪ Population Density')
```



[37]:
```
p = johns.plot.scatter(x='Population_Density', y='Death_Rate',␣
↪color=list(johns['Winner'].reset_index()['Winner']), title='Death Rate vs.␣
↪Population Density')
```



1 Normalizing Data To Run Regression to Predict % Democrat

[111]:
```
johns.columns
```

[111]:
```
Index(['Province_State', 'Lat', 'Long', 'Confirmed', 'Deaths', 'Incident_Rate',
       'Total_Test_Results', 'Case_Fatality_Ratio', 'Testing_Rate',
       'Unnamed: 18', 'Unnamed: 19', 'Area', 'Population Over 60',
       'Population', 'Death_Rate', 'percent_democrat', 'total_vaccinations',
       'total_distributed', 'people_vaccinated',
       'people_fully_vaccinated_per_hundred', 'Total_Vaccinations_Per_100',
       'people_fully_vaccinated', 'people_vaccinated_per_hundred',
       'distributed_per_hundred', 'daily_vaccinations_raw',
       'daily_vaccinations', 'daily_vaccinations_per_million',
       'share_doses_used', 'Pfizer_1', 'Pfizer_2', 'Moderna_1', 'Moderna_2',
       'Janssen_1', 'Vaccine_Doses_Per_100', 'Percent_Pfizer',
       'Percent_Moderna', 'Percent_Janssen', 'Can_Vaccinate_Per_100', 'Winner',
       'Percent_Over_60', 'Population_Density'],
      dtype='object')
```

int_cols is just interesting columns (variables) that we want to look into in order to predict percent_democrat

[112]:
```
int_cols = ['Incident_Rate', 'Total_Vaccinations_Per_100', 'Death_Rate',␣
↪'Testing_Rate', 'Vaccine_Doses_Per_100', 'Lat', 'Long',␣
↪'Case_Fatality_Ratio', 'Percent_Over_60', 'Population_Density']
target = 'percent_democrat'
```

Normalized the data here so β's are easier to understand, not sure if this is necessary.

[113]:
```
import pandas as pd
from sklearn import preprocessing

x = johns[int_cols].values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
X = pd.DataFrame(x_scaled)
```

[114]:
```
d = {}
for i in range(len(X.columns)):
    d[i] = johns[int_cols].columns[i]
X = X.rename(d, axis=1)
```

[115]:
```
def minAIC(X,y):
    variables = X.columns
    model = sm.OLS(y,X[variables]).fit()
    while True:
        maxp = np.max(model.pvalues)
        newvariables = variables[model.pvalues < maxp]
        newmodel = sm.OLS(y,X[newvariables]).fit()
        if newmodel.aic < model.aic:
            model = newmodel
            variables = newvariables
        else:
            break
    return model,variables
```

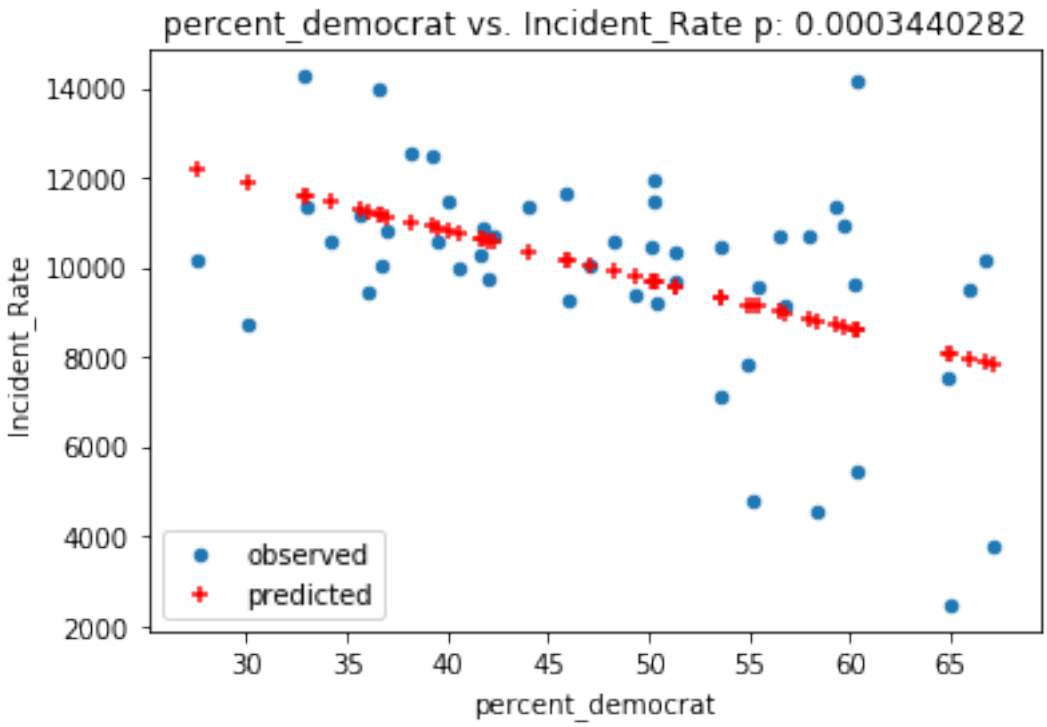
Plotting each individual variable and percent democrat, also running linear regression to see which actually might be correlated.

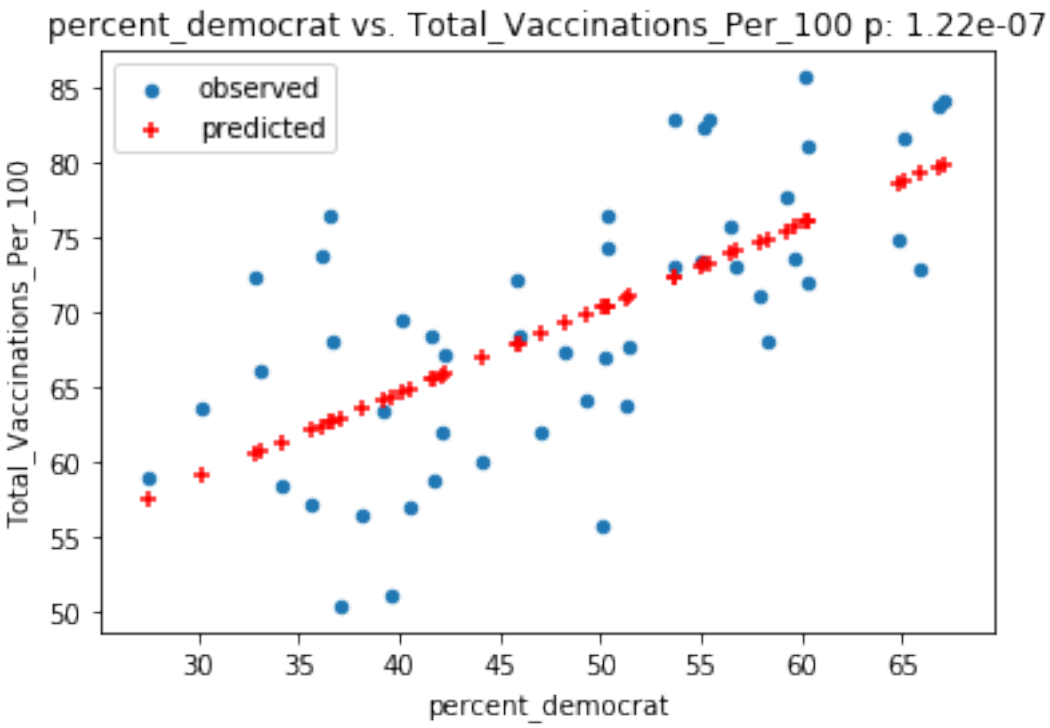
[116]:
```
j = johns[int_cols + [target]]
```

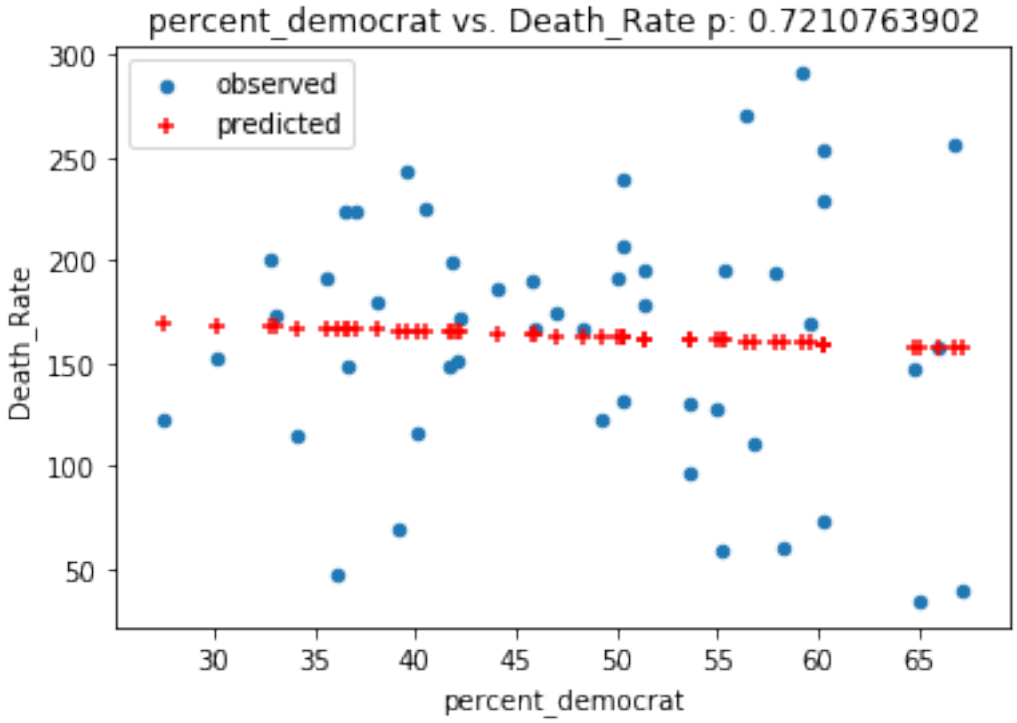
[117]:
```
plt.figure(figsize=(20,10))
for var in int_cols:
    x = sm.add_constant(j[target])
    Y = j[var]

    model = sm.OLS(Y, x).fit()
    Yhat = model.predict(x)
```

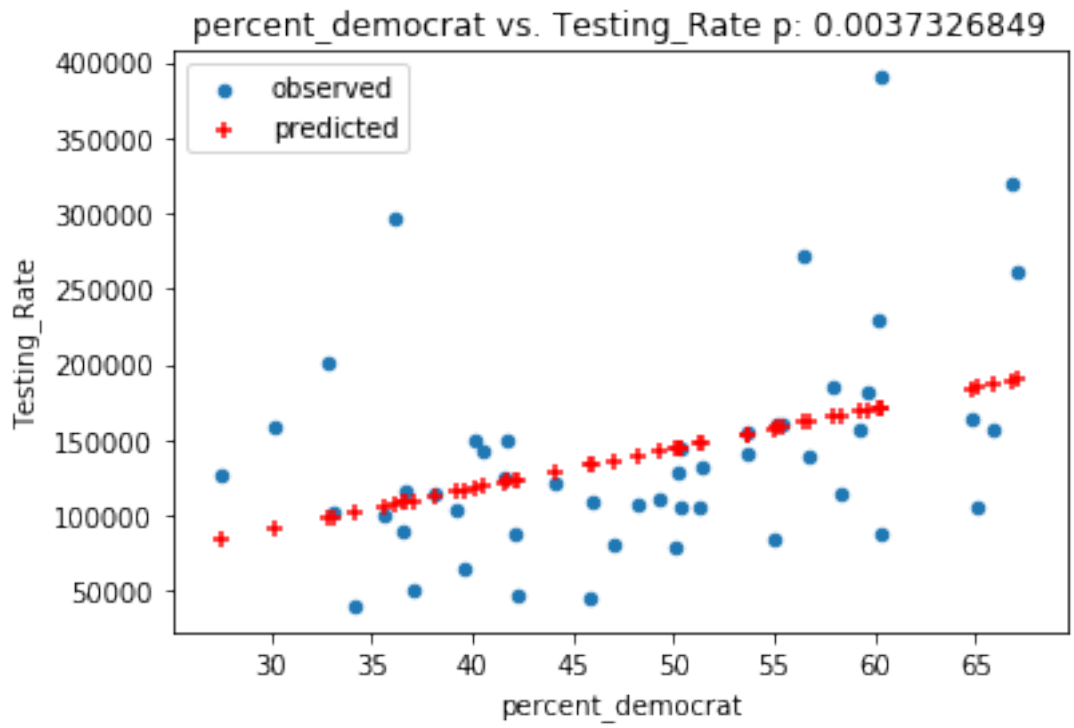
```
    p = j.plot.scatter(x=target, y=var, label='observed')
    p.scatter(x=j[target], y=Yhat, color='r', marker="+", label='predicted')
    plt.legend()
    plt.title(target + ' vs. ' + var + ' p: ' + str(round(model.
↪pvalues[target],10)))
```

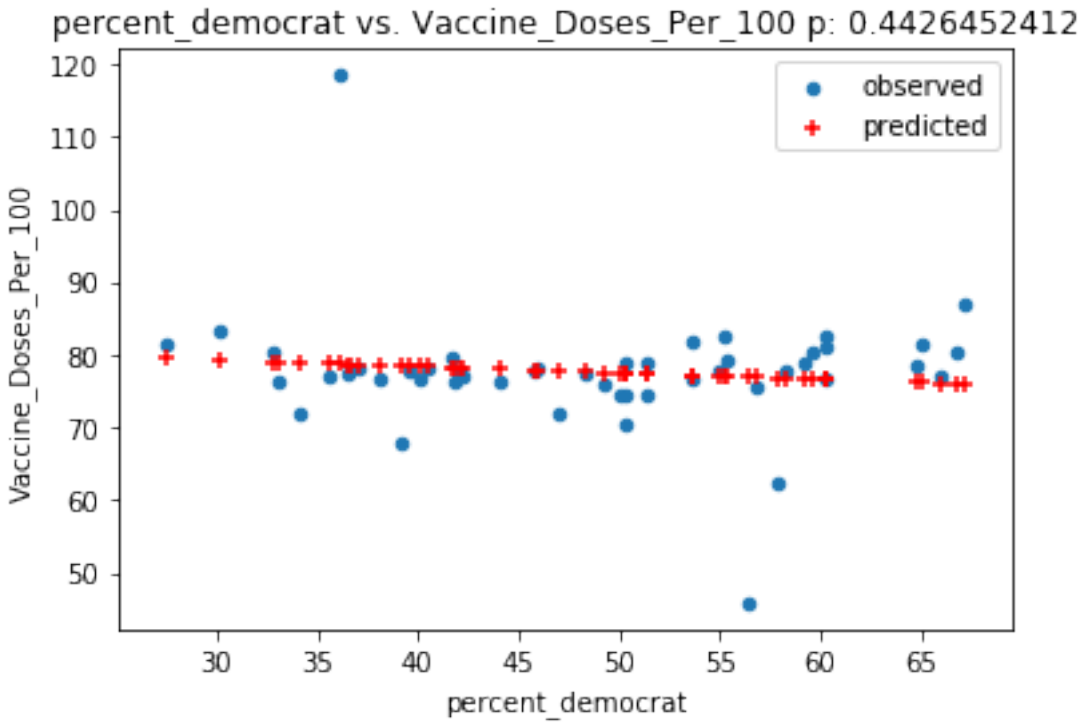
```
<Figure size 1440x720 with 0 Axes>
```

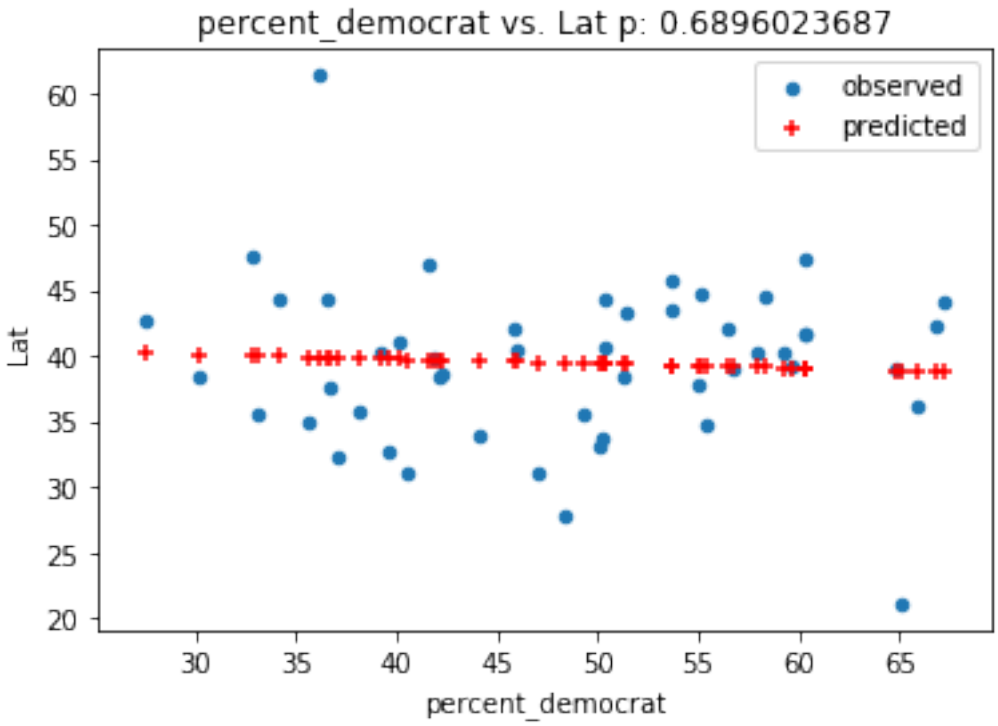


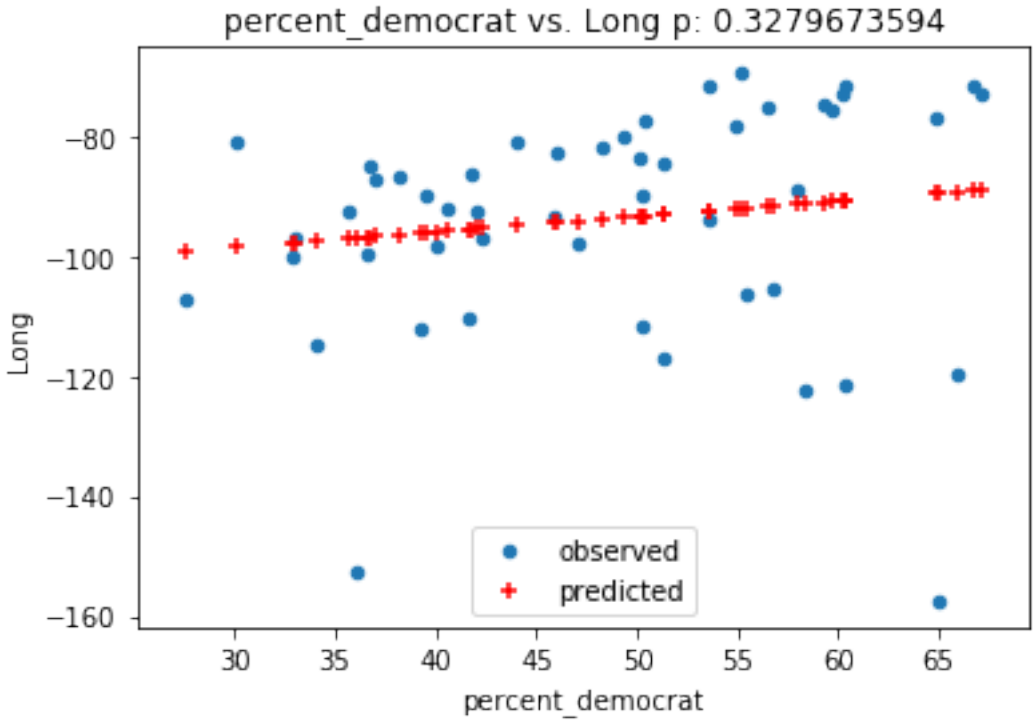

percent_democrat vs. Total_Vaccinations_Per_100 p: 1.22e-07
observed
predicted
Total_Vaccinations_Per_100
percent_democrat
85
80
75
70
65
60
55
50
30
35
40
45
50
55
60
65

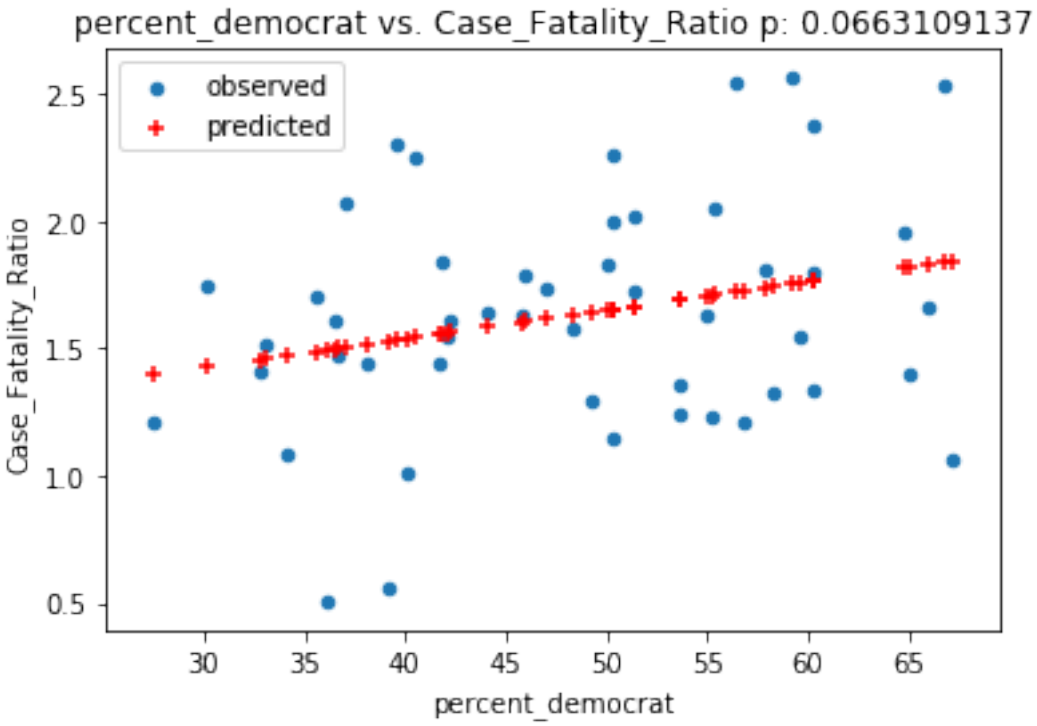

percent_democrat vs. Death_Rate p: 0.7210763902
observed
predicted
Death_Rate
percent_democrat
300
250
200
150
100
50
30
35
40
45
50
55
60
65

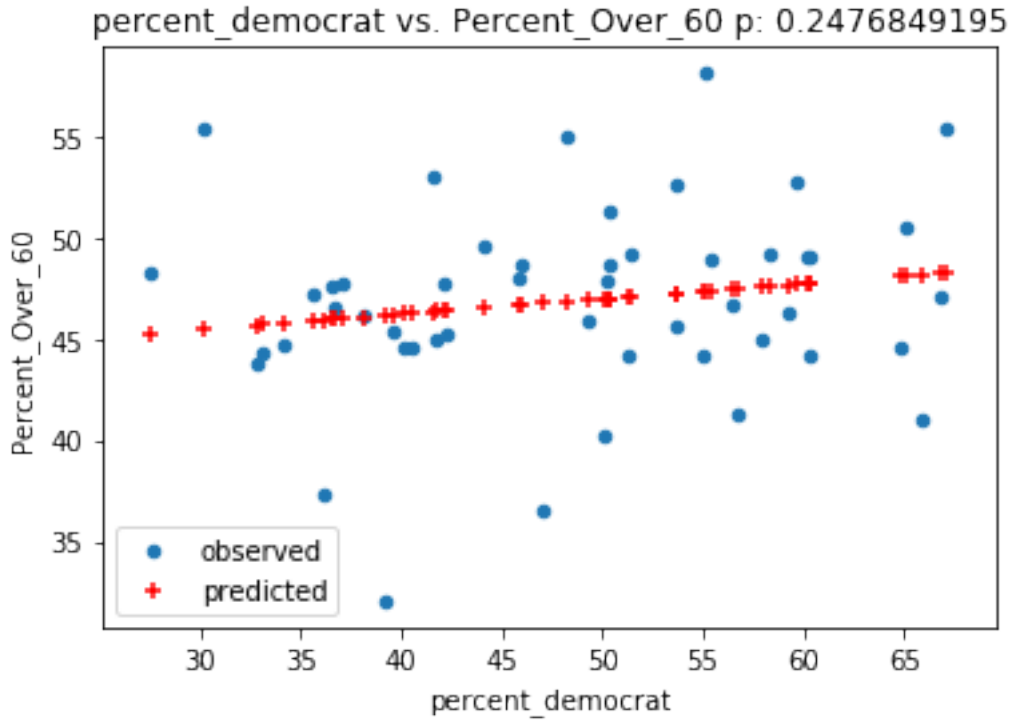

percent_democrat vs. Testing_Rate p: 0.0037326849
observed
predicted
Testing_Rate
400000
350000
300000
250000
200000
150000
100000
50000
30
35
40
45
50
55
60
65
percent_democrat

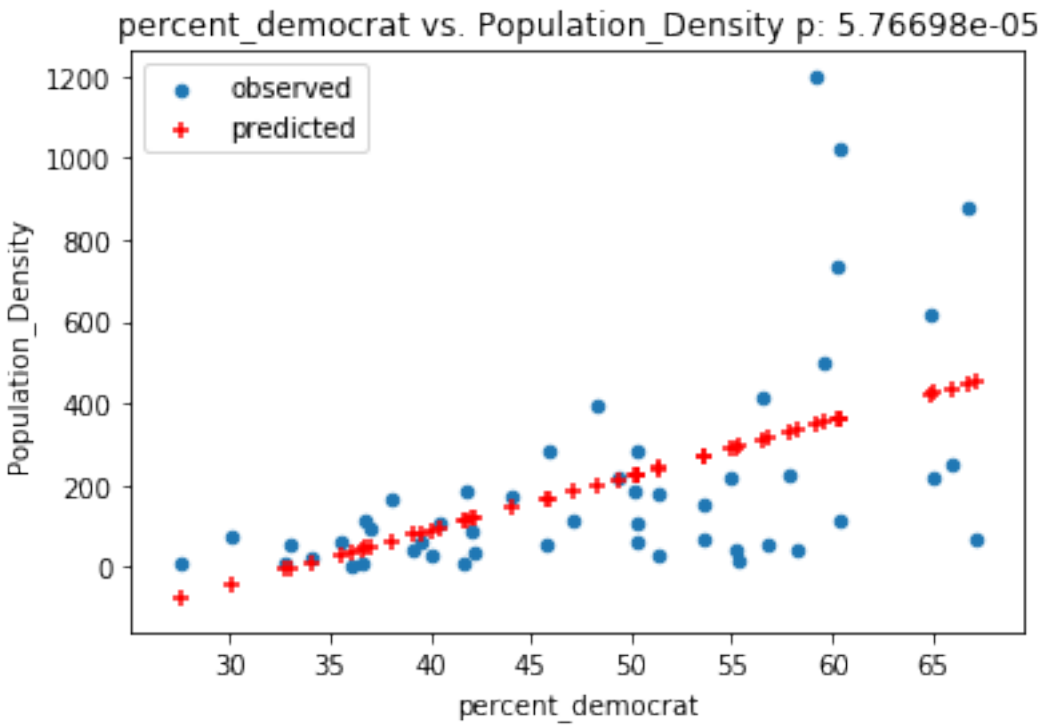

percent_democrat vs. Vaccine_Doses_Per_100 p: 0.4426452412
observed
predicted
Vaccine_Doses_Per_100
120
110
100
90
80
70
60
50
30
35
40
45
50
55
60
65
percent_democrat


percent_democrat vs. Lat p: 0.6896023687
observed
predicted
Lat
percent_democrat
60
55
50
45
40
35
30
25
20
30
35
40
45
50
55
60
65


percent_democrat vs. Long p: 0.3279673594
observed
predicted
Long
percent_democrat
−80
−100
−120
−140
−160
30
35
40
45
50
55
60
65

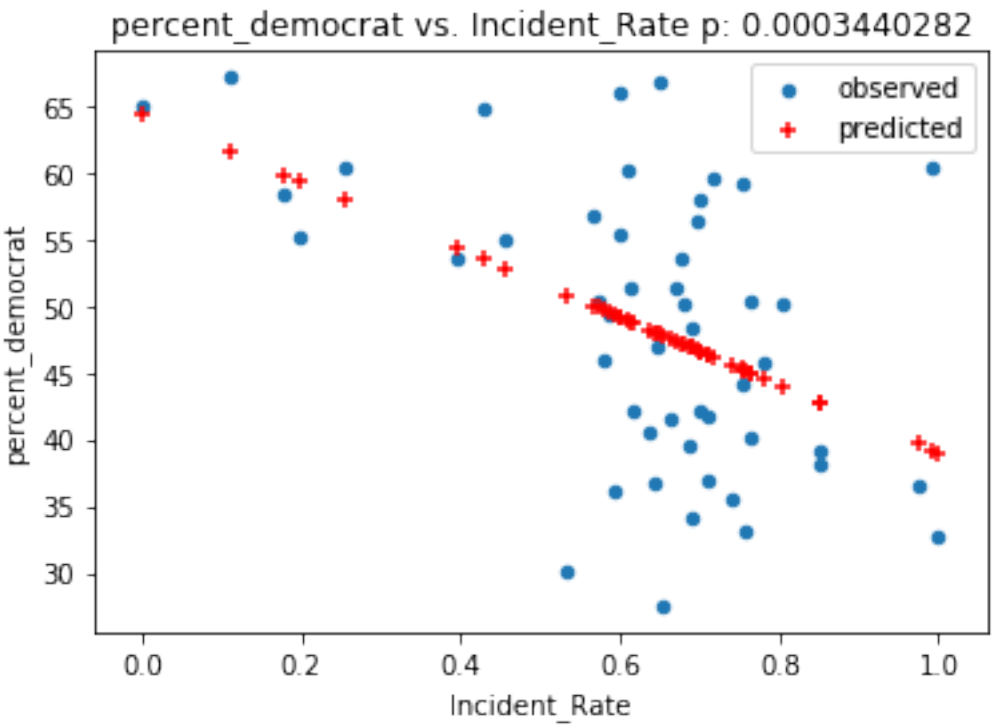

percent_democrat vs. Case_Fatality_Ratio p: 0.0663109137
observed
predicted
Case_Fatality_Ratio
percent_democrat
2.5
2.0
1.5
1.0
0.5
30
35
40
45
50
55
60
65

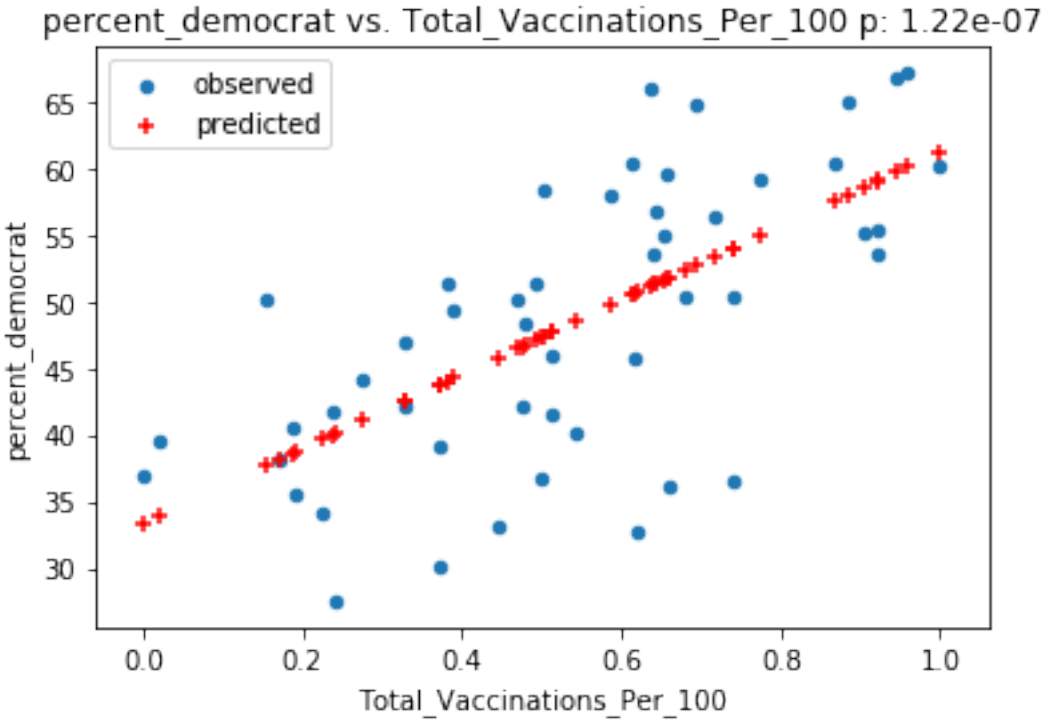

percent_democrat vs. Percent_Over_60 p: 0.2476849195
observed
predicted
Percent_Over_60
percent_democrat
55
50
45
40
35
30
35
40
45
50
55
60
65

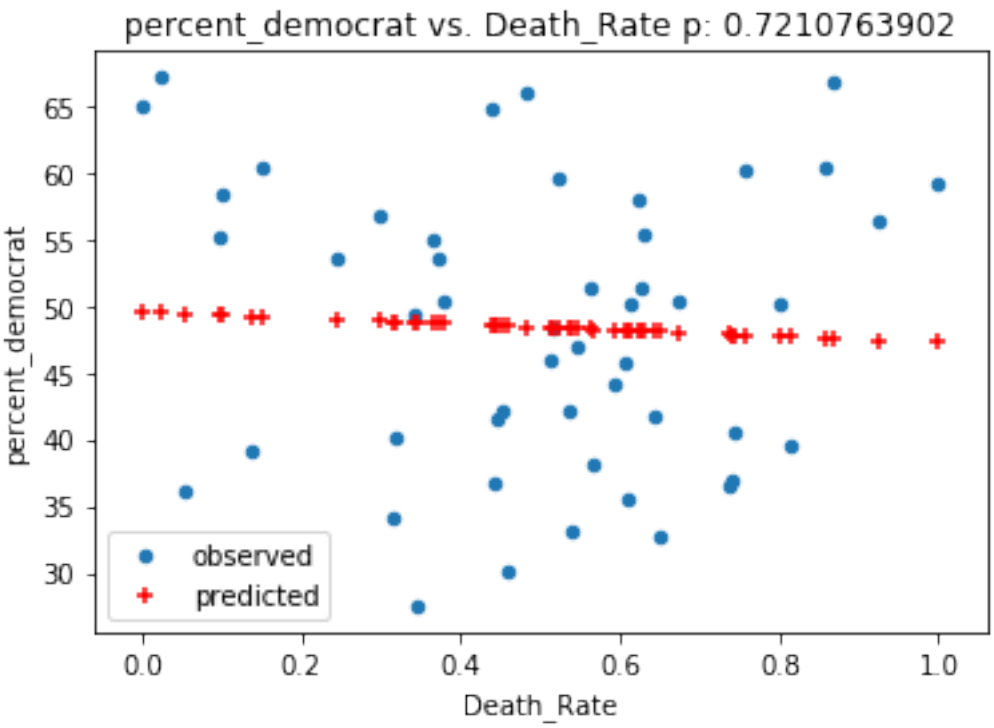


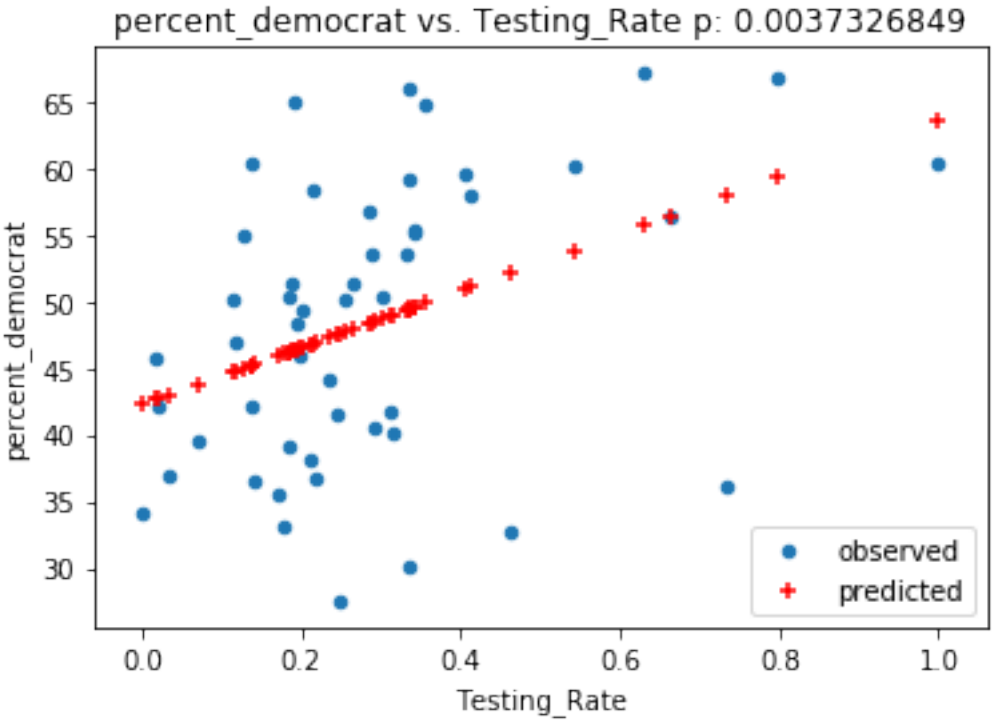
[118]:
```
plt.figure(figsize=(20,10))
X[target] = johns[target]
ps = {}
for var in int_cols:
    x = sm.add_constant(X[var])
    Y = johns[target]

    model = sm.OLS(Y, x).fit()
    Yhat = model.predict(x)

    p = X.plot.scatter(x=var, y=target, label='observed')
    p.scatter(x=X[var], y=Yhat, color='r', marker="+", label='predicted')
    plt.legend()
    plt.title(target + ' vs. ' + var + ' p: ' + str(round(model.
 ↪pvalues[var],10)))
    ps[var] = model.pvalues[var]
```

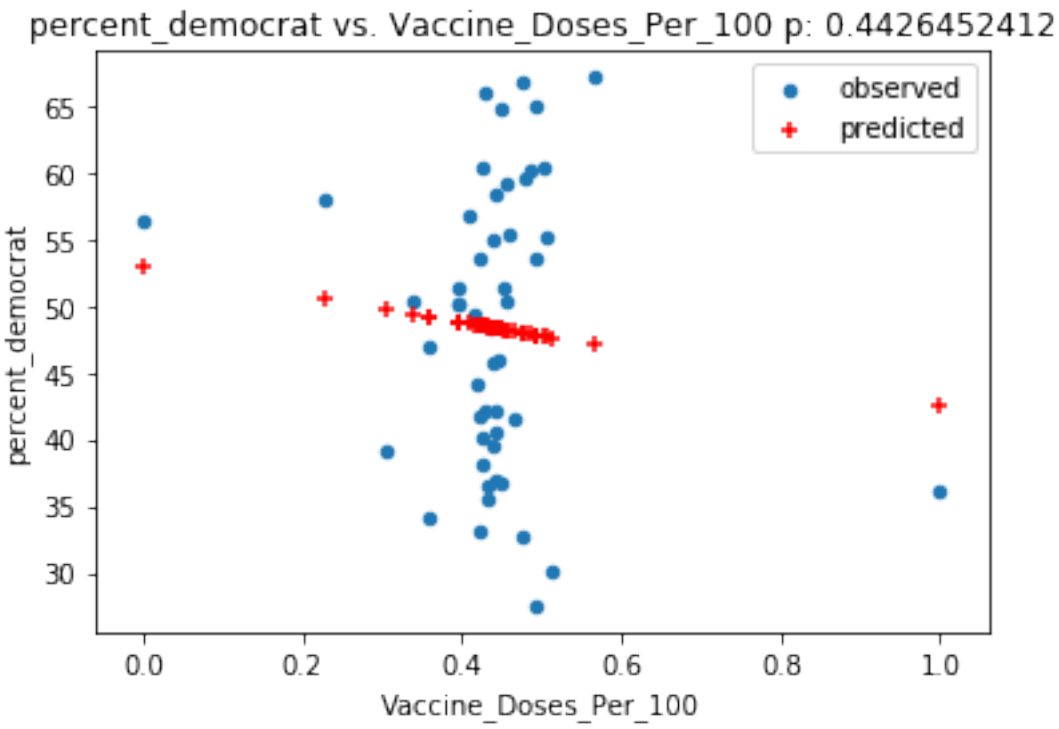
```
<Figure size 1440x720 with 0 Axes>
```

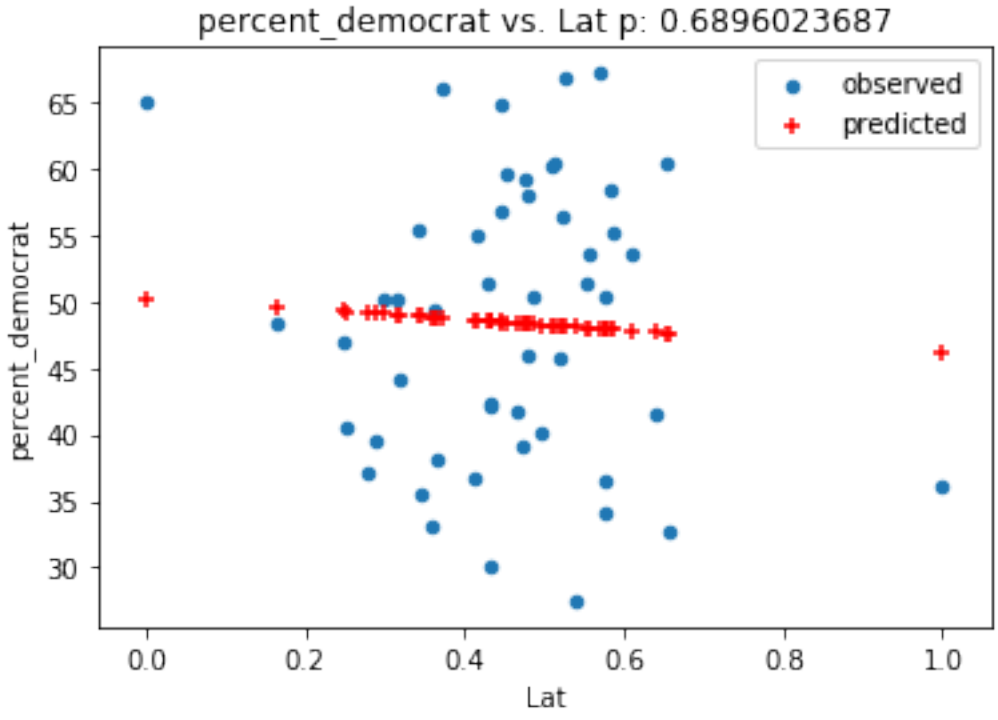

percent_democrat vs. Incident_Rate p: 0.0003440282
observed
predicted
percent_democrat
Incident_Rate
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0

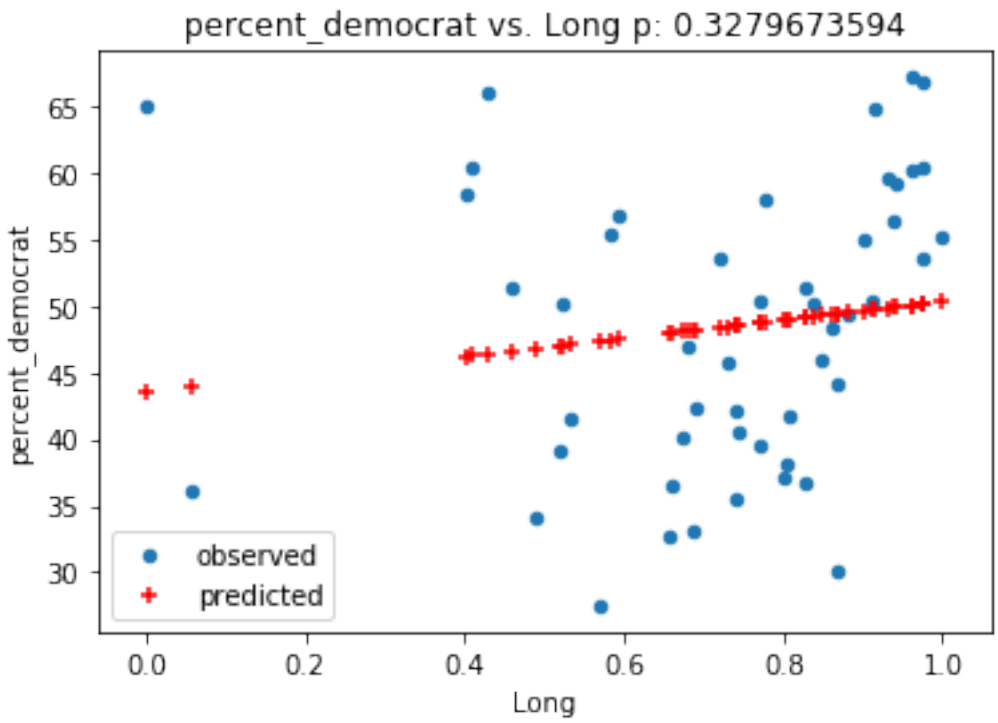

percent_democrat vs. Total_Vaccinations_Per_100 p: 1.22e-07
observed
predicted
percent_democrat
Total_Vaccinations_Per_100
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0

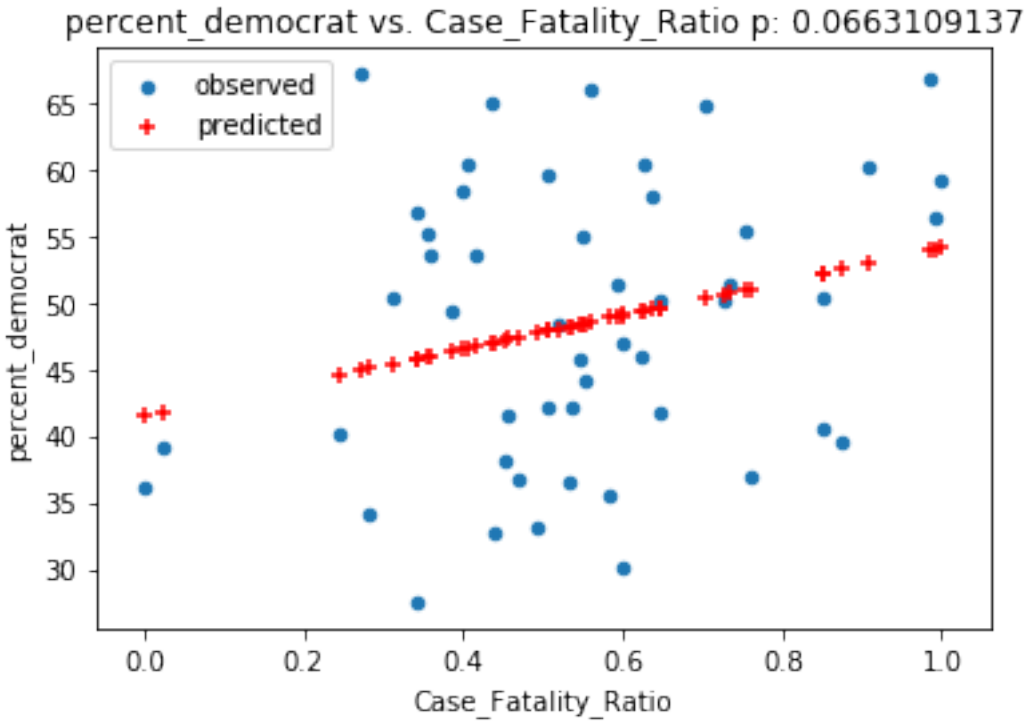

percent_democrat vs. Death_Rate p: 0.7210763902
percent_democrat
Death_Rate
observed
predicted
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0

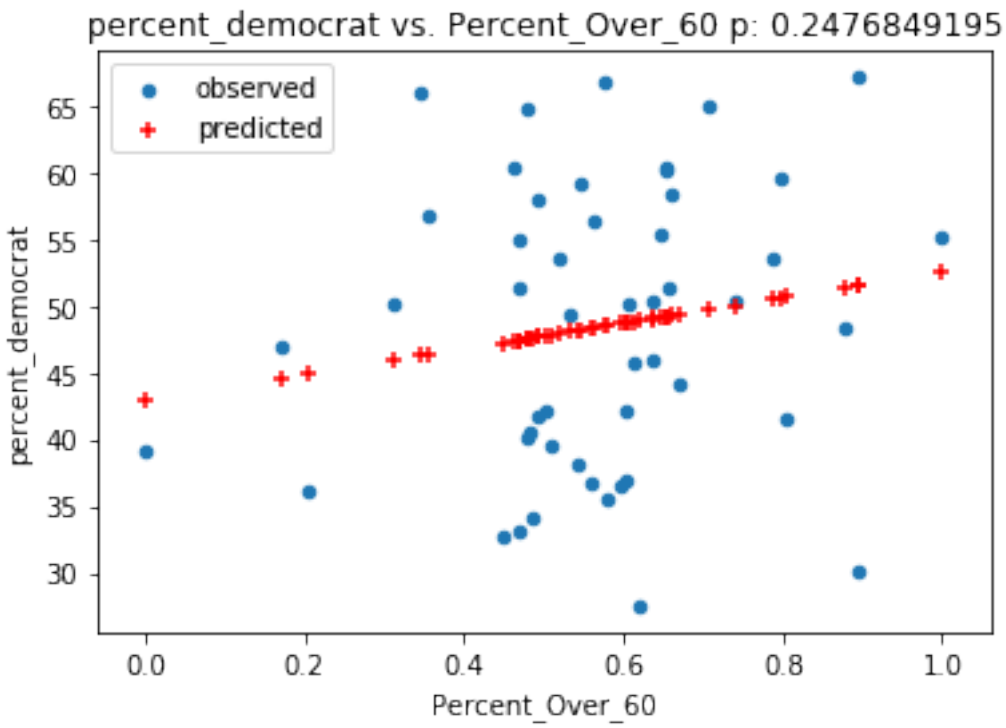

percent_democrat vs. Testing_Rate p: 0.0037326849
percent_democrat
Testing_Rate
observed
predicted
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0

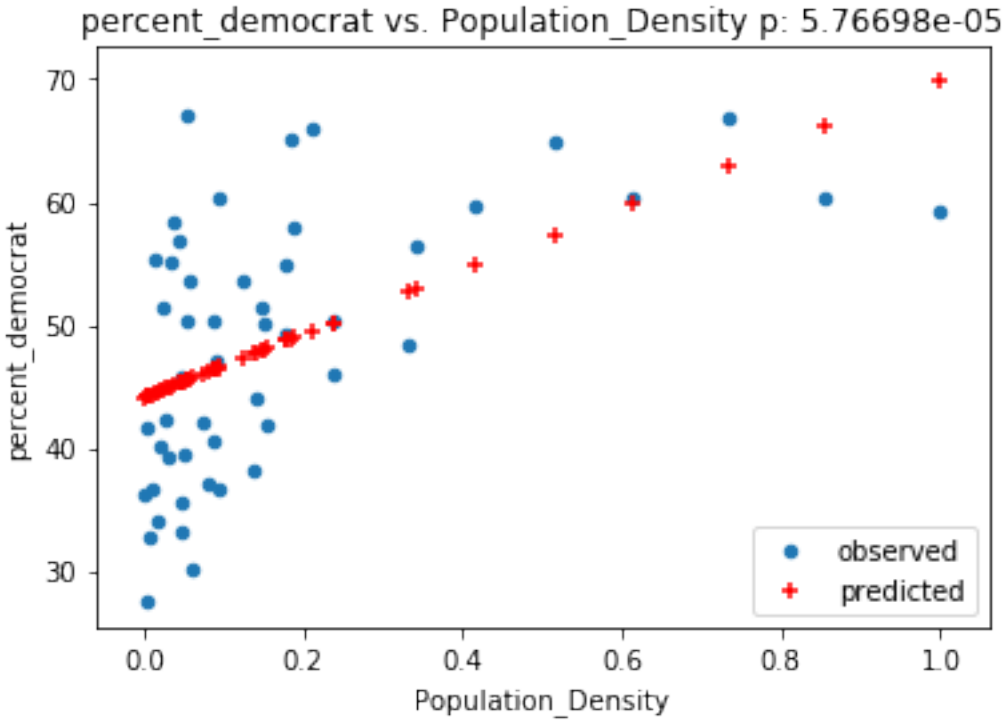

percent_democrat vs. Vaccine_Doses_Per_100 p: 0.4426452412
observed
predicted
percent_democrat
Vaccine_Doses_Per_100
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0


percent_democrat vs. Lat p: 0.6896023687
observed
predicted
percent_democrat
Lat
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0


percent_democrat vs. Long p: 0.3279673594
percent_democrat
Long
observed
predicted
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0


percent_democrat vs. Case_Fatality_Ratio p: 0.0663109137
percent_democrat
Case_Fatality_Ratio
observed
predicted
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0


percent_democrat vs. Percent_Over_60 p: 0.2476849195
observed
predicted
percent_democrat
Percent_Over_60
65
60
55
50
45
40
35
30
0.0
0.2
0.4
0.6
0.8
1.0


percent_democrat vs. Population_Density p: 5.76698e-05
observed
predicted
percent_democrat
Population_Density
70
60
50
40
30
0.0
0.2
0.4
0.6
0.8
1.0

Dropping anything that's not correlated whatsoever – don't even wanna run min AIC on these because they might affect the stuff that actually has an impact

[119]:
```
X = X.drop(target, axis=1)
for var in int_cols:
    if ps[var] > 0.05 and var != target:
        X = X.drop(var, axis=1)
```

These are the variables that might be correlated:

[120]:
```
X.columns
```

[120]:
```
Index(['Incident_Rate', 'Total_Vaccinations_Per_100', 'Testing_Rate',
       'Population_Density'],
      dtype='object')
```

[121]:
```
X = sm.add_constant(X)
y = johns[target]
```

[122]:
```
m = sm.OLS(y,X).fit()
m.summary()
```

[122]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:       percent_democrat   R-squared:                       0.676
Model:                           OLS   Adj. R-squared:                  0.648
Method:                Least Squares   F-statistic:                     23.52
Date:               Sun, 16 May 2021   Prob (F-statistic):           1.53e-10
Time:                       18:33:31   Log-Likelihood:                -160.43
No. Observations:                 50   AIC:                             330.9
Df Residuals:                     45   BIC:                             340.4
Df Model:                          4
Covariance Type:           nonrobust
==============================================================================
==============
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------
const                         50.6608      4.339     11.675      0.000
41.921      59.400
Incident_Rate                -21.6682      4.964     -4.365      0.000
-31.667     -11.670
Total_Vaccinations_Per_100    15.4991      4.901      3.162      0.003
5.627      25.371
Testing_Rate                  -1.9354      6.135     -0.315      0.754
```

```
-14.291      10.421
Population_Density            21.5955      4.955      4.359      0.000
11.617      31.574
==============================================================================
Omnibus:                        0.646   Durbin-Watson:                   2.270
Prob(Omnibus):                  0.724   Jarque-Bera (JB):                0.288
Skew:                          -0.181   Prob(JB):                        0.866
Kurtosis:                       3.085   Cond. No.                         11.8
==============================================================================

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

[123]:
```
model,variables = minAIC(X,y)
model.summary()
```

[123]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:       percent_democrat   R-squared:                       0.676
Model:                           OLS   Adj. R-squared:                  0.655
Method:                Least Squares   F-statistic:                     31.95
Date:               Sun, 16 May 2021   Prob (F-statistic):           2.58e-11
Time:                       18:33:31   Log-Likelihood:                -160.48
No. Observations:                 50   AIC:                             329.0
Df Residuals:                     46   BIC:                             336.6
Df Model:                          3
Covariance Type:            nonrobust
==============================================================================
==============
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------
const                         50.8159      4.269     11.904      0.000
42.223      59.409
Incident_Rate                -21.9876      4.812     -4.569      0.000
-31.674     -12.301
Total_Vaccinations_Per_100    14.7257      4.202      3.504      0.001
6.267      23.185
Population_Density            21.0522      4.600      4.577      0.000
11.793      30.311
==============================================================================
Omnibus:                        1.066   Durbin-Watson:                   2.284
```

```
Prob(Omnibus):                  0.587   Jarque-Bera (JB):                0.518
Skew:                          -0.228   Prob(JB):                        0.772
Kurtosis:                       3.203   Cond. No.                         10.6
==============================================================================

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

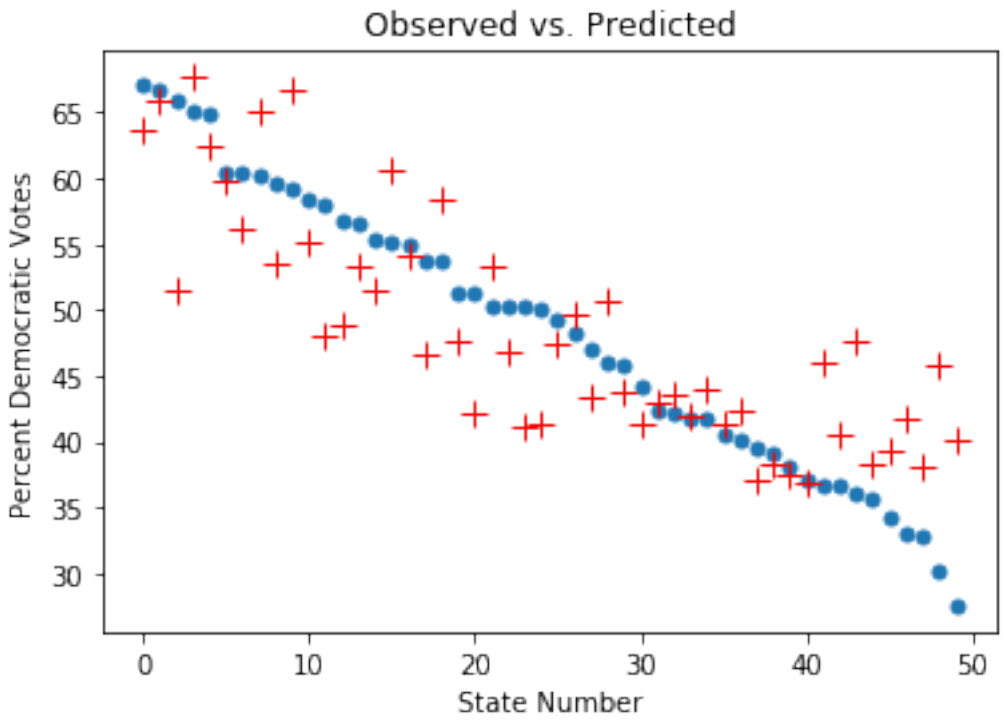
Really interesting – incidence rate and population density are opposite in sign, which means that more democrats tend to have higher state population densities but at the same time lower incident rates, I feel like this is important.

[124]:
```
X[target] = johns[target]
```

[125]:
```
jo = X.sort_values(target, ascending=False)
jo=jo.reset_index()
jo=jo.drop('index', axis=1)
```

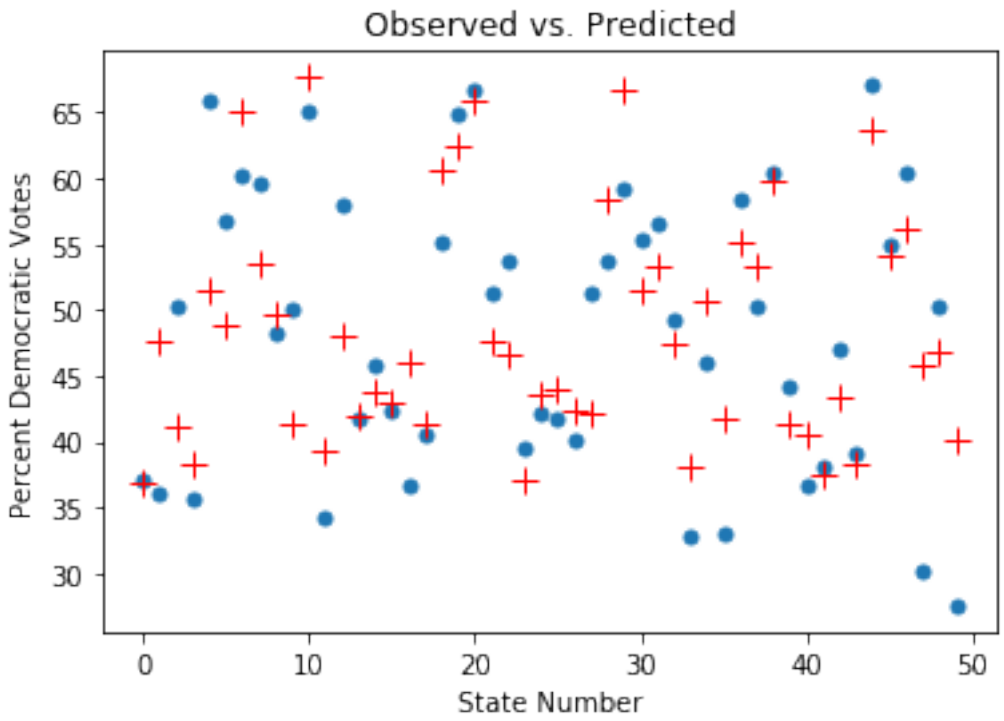
[126]:
```
plt.plot(jo.index, jo[target], '.', markersize=10, label='observed')
plt.plot(jo.index, model.predict(jo[variables]), '+', markersize=10,␣
 ↪label='predicted', color='red')
plt.title('Observed vs. Predicted')
plt.xlabel('State Number')
plt.ylabel('Percent Democratic Votes')
```

[126]:
```
Text(0, 0.5, 'Percent Democratic Votes')
```



[127]:
```
plt.plot(X.index, johns[target], '.', markersize=10, label='observed')
plt.plot(X.index, model.predict(X[variables]), '+', markersize=10,␣
 ↪label='predicted', color='red')
plt.title('Observed vs. Predicted')
plt.xlabel('State Number')
plt.ylabel('Percent Democratic Votes')
```

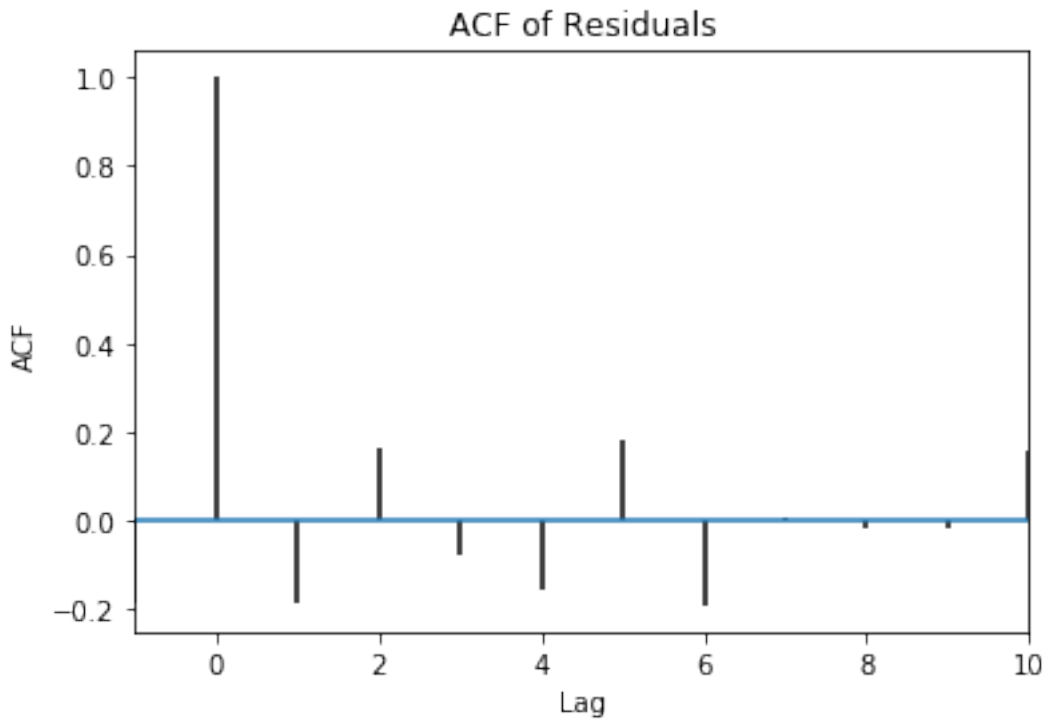
[127]:
```
Text(0, 0.5, 'Percent Democratic Votes')
```



2 Test Mutual Independence by Plotting Autocorrelations

[128]:
```
plt.acorr(model.resid)
plt.ylabel('ACF')
plt.xlim(-1,10)
plt.xlabel('Lag')
plt.title('ACF of Residuals')
```

[128]: Text(0.5, 1.0, 'ACF of Residuals')

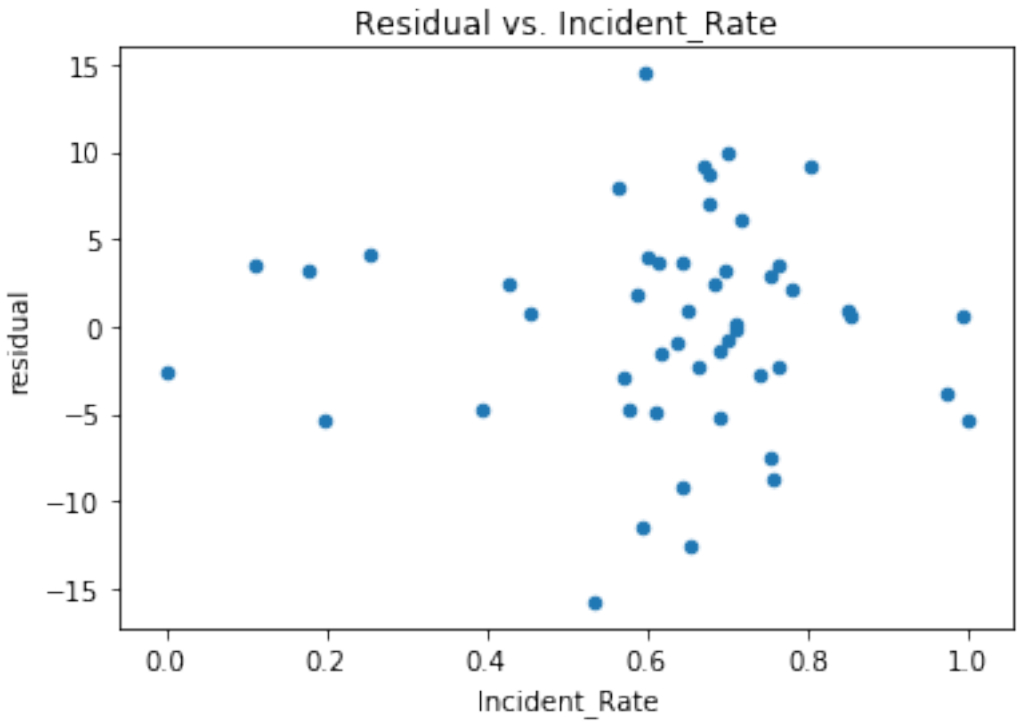


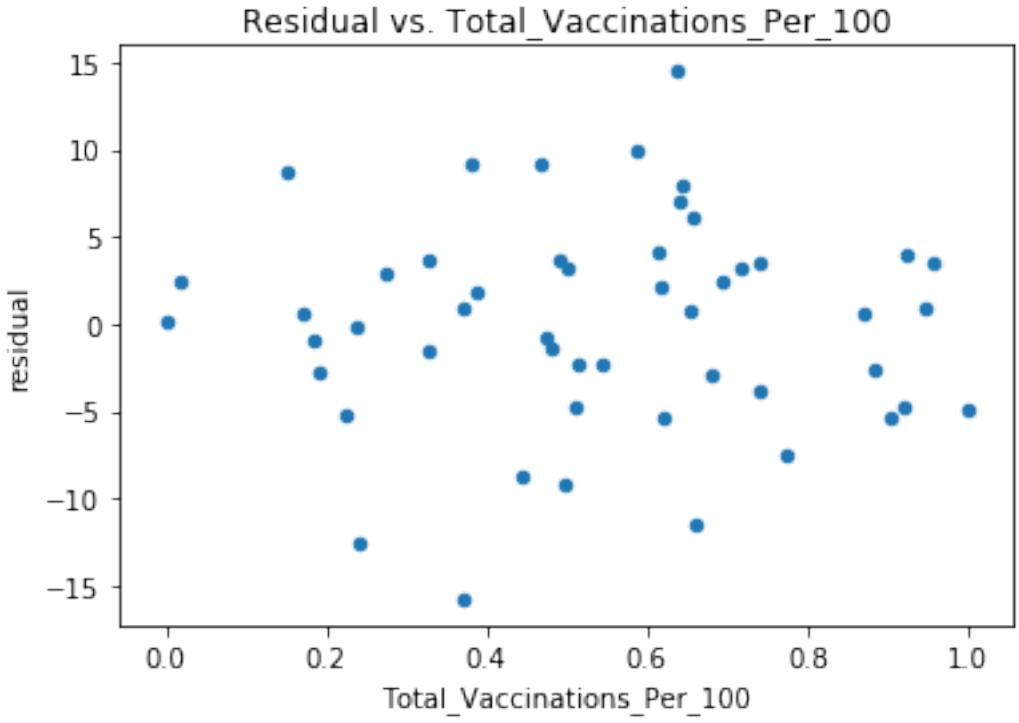
Note that all other ones are within 0.2, seem to be pretty independent.

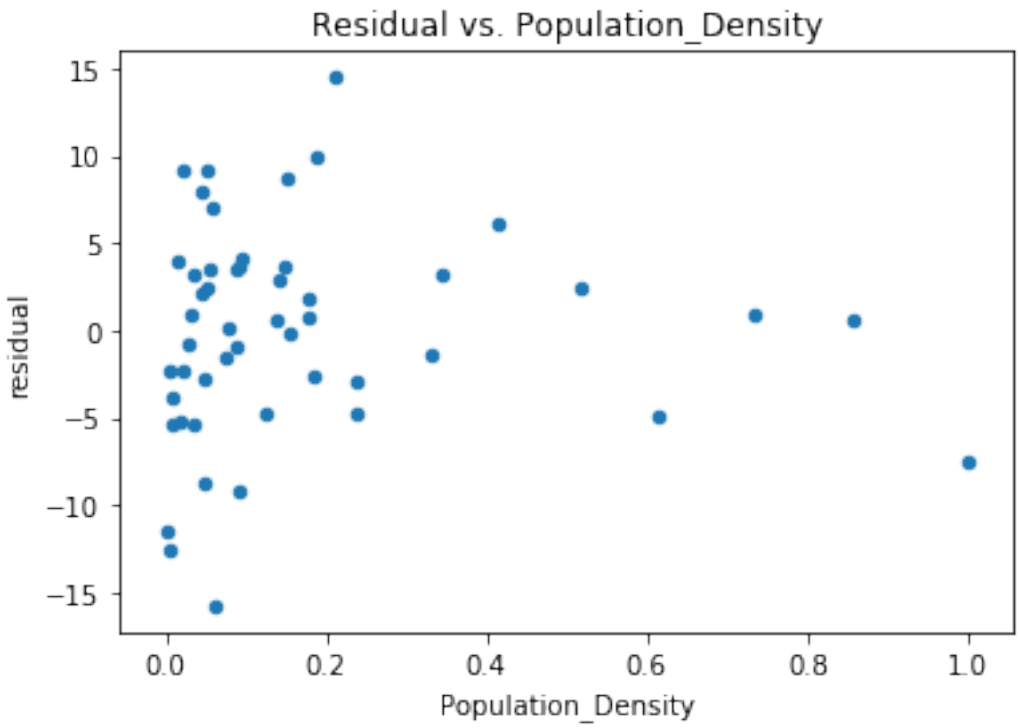
3 test linearity (equivalently, independence of residuals and covariates)

[129]:
```
X['residual'] = model.resid
```

[133]:
```
for var in variables:
    if var != 'const':
        p = X.plot.scatter(x=var, y='residual', title='Residual vs. ' + var)
```


Residual vs. Incident_Rate
residual
Incident_Rate
15
10
5
0
−5
−10
−15
0.0
0.2
0.4
0.6
0.8
1.0


Residual vs. Total_Vaccinations_Per_100
residual
Total_Vaccinations_Per_100
15
10
5
0
−5
−10
−15
0.0
0.2
0.4
0.6
0.8
1.0

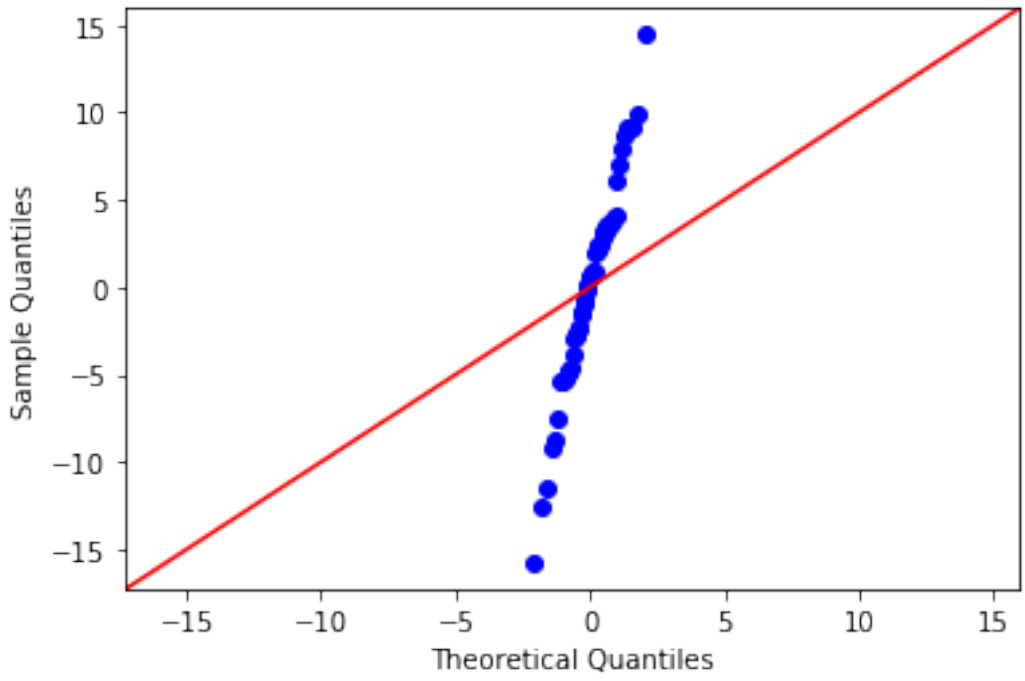


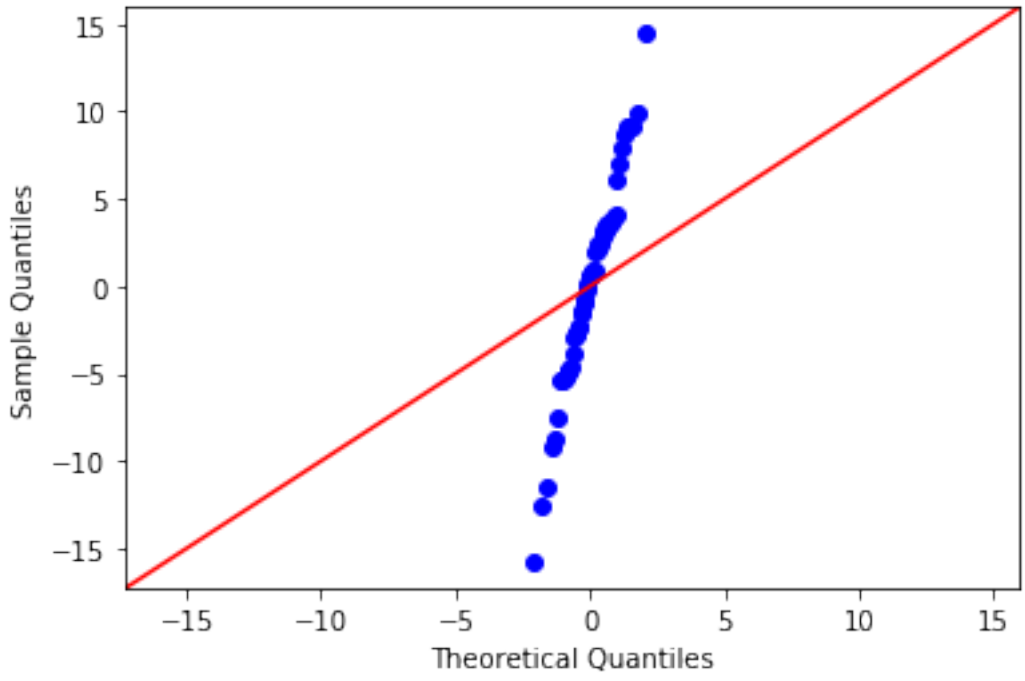
All covaritates seem to be independent of the residuals.

4 QQ plots for assessing normality

[131]:
```
sm.qqplot(model.resid, line='45')
```

[131]:

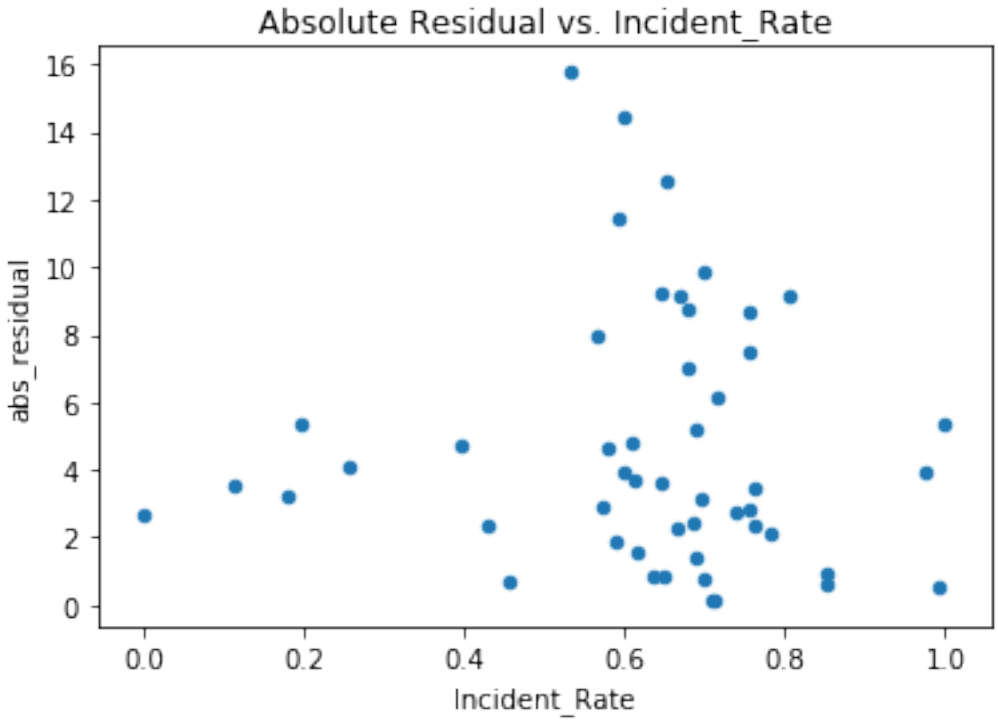


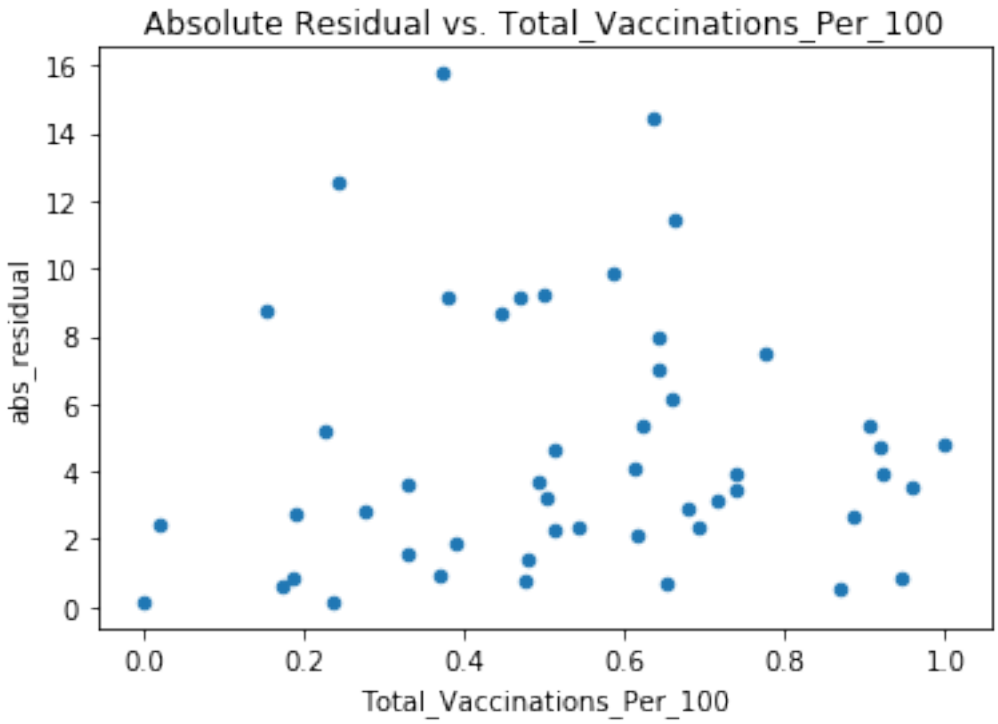


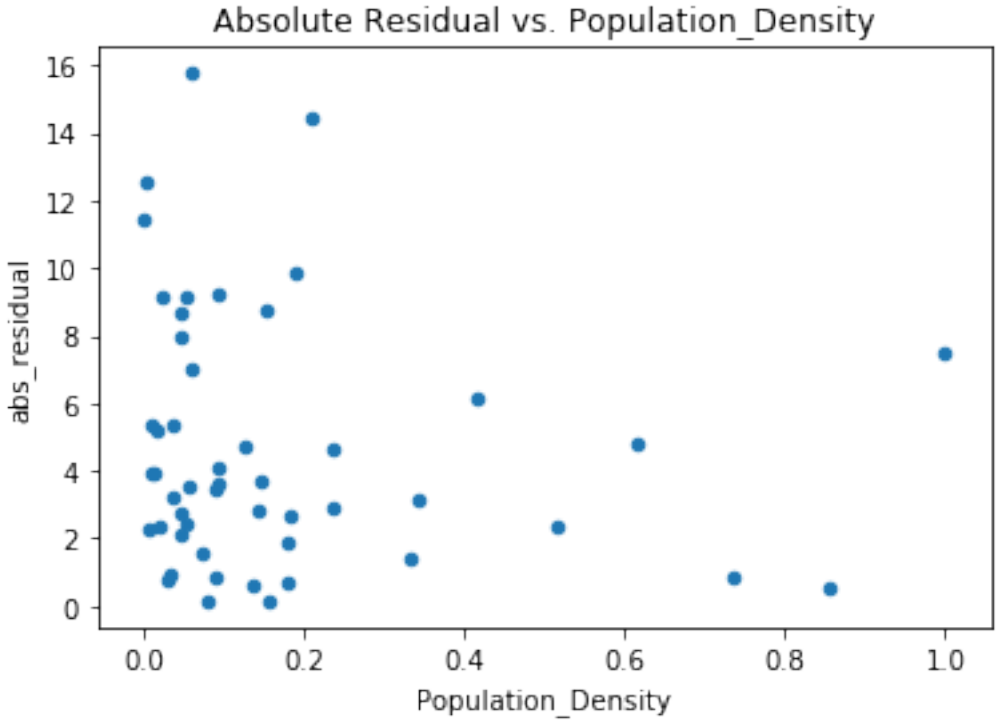
Seems pretty linear (even though not 45 degrees) so looks normal.

5 Checking homoskedasticity

[132]:
```
X['abs_residual'] = abs(X['residual'])
for var in variables:
    if var != 'const':
        p = X.plot.scatter(x=var, y='abs_residual', title='Absolute Residual vs.
 ↪ ' + var )
```




Absolute Residual vs. Total_Vaccinations_Per_100
abs_residual
Total_Vaccinations_Per_100
0
2
4
6
8
10
12
14
16
0.0
0.2
0.4
0.6
0.8
1.0


Absolute Residual vs. Population_Density
abs_residual
Population_Density
0
2
4
6
8
10
12
14
16
0.0
0.2
0.4
0.6
0.8
1.0

There seems to be no relation between the covariates and absolute residual.

[ ]:

Appendix Code Continued

May 21, 2021

1 Forecasting Now

Forecasting vaccination rates using most recent vaccination data from Kaggle.

[67]:
```
vac = pd.read_csv('vax_updated.csv')
abbrevs = pd.read_csv('abbrevs.csv')
```

[68]:
```
vac = vac.replace('New York State', 'New York')
```

[69]:
```
vac = vac.join(abbrevs.set_index('State'), on='location', how='inner')
```

[70]:
```
ny = vac[vac['location'] == 'New York']
ny.columns
```

[70]:
```
Index(['date', 'location', 'total_vaccinations', 'total_distributed',
       'people_vaccinated', 'people_fully_vaccinated_per_hundred',
       'total_vaccinations_per_hundred', 'people_fully_vaccinated',
       'people_vaccinated_per_hundred', 'distributed_per_hundred',
       'daily_vaccinations_raw', 'daily_vaccinations',
       'daily_vaccinations_per_million', 'share_doses_used', 'Abbrev', 'Code',
       'Winner'],
      dtype='object')
```

[71]:
```
measure = 'daily_vaccinations_per_million'
```

Looking just at New York right now

[72]:
```
ny=ny[['date', measure, 'Winner']]
```

[73]:
```
ny['date'] = ny['date'].apply(lambda x: pd.to_datetime(x))
```

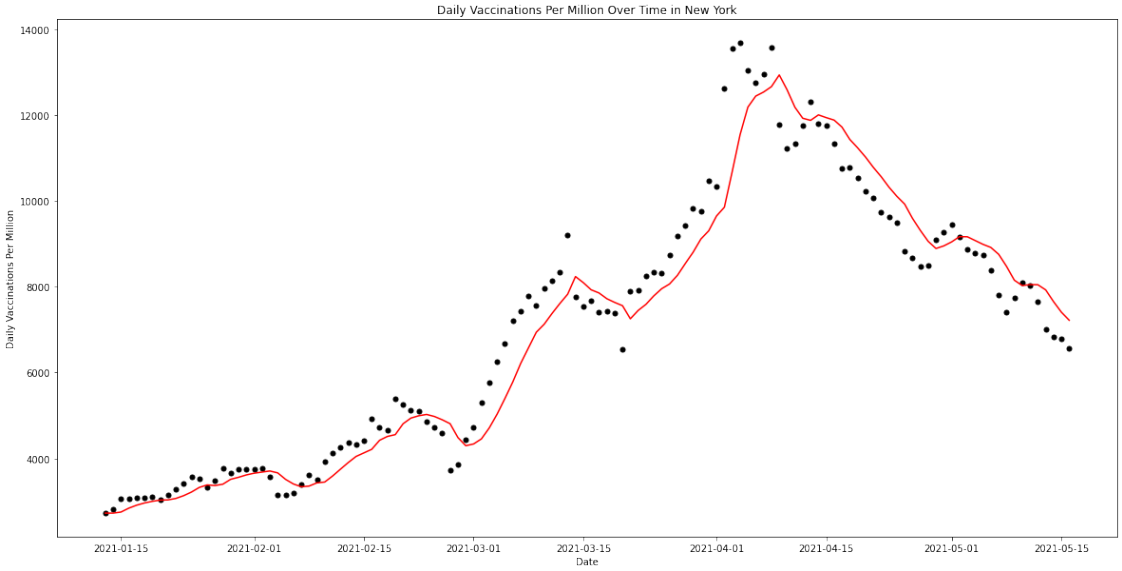
[74]:
```
ny = ny.dropna(subset=[measure])
```

Simple exponential smoothing

[75]:
```
plt.figure(figsize=(20,10))
fit = SimpleExpSmoothing(ny[measure]).fit(smoothing_level=0.3, optimized=False)
xhat = fit.fittedvalues
```

```
plt.scatter(ny['date'], ny[measure], marker='.', color='black', s=100)
plt.plot(ny['date'], xhat, color='red')
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time in New York')
```

[75]: Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time in New York')

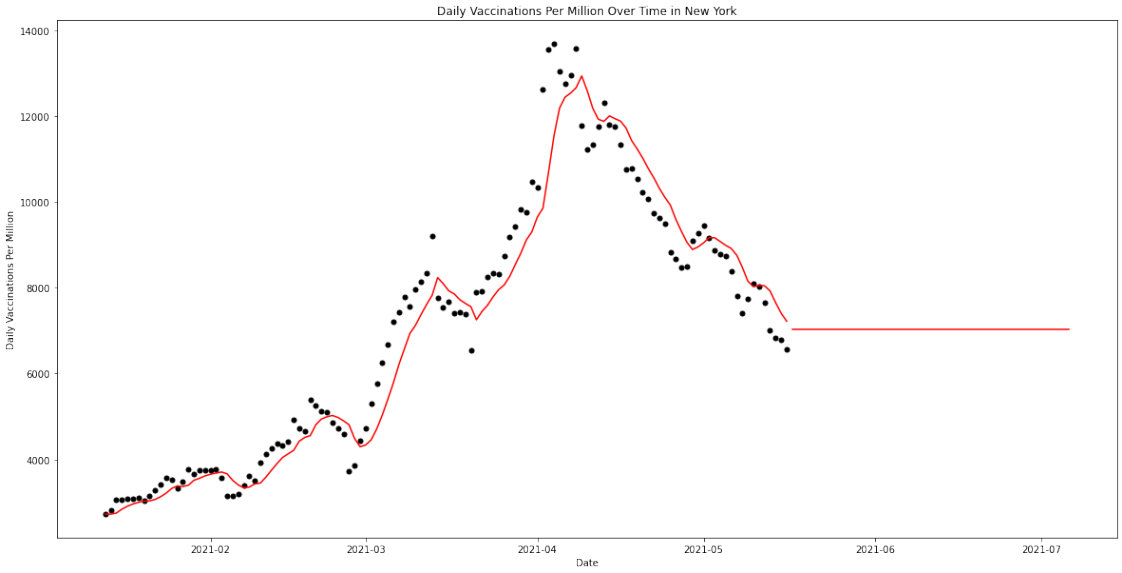


[76]:
```
L = 50
lst = [ny['date'].iloc[-1] + pd.Timedelta('1 day')]
for i in range(L):
    lst.append(lst[-1] + pd.Timedelta('1 day'))
```

Forecasting constant value for simple exponential smoothing

[77]:
```
forecast = fit.forecast(L+1)
plt.figure(figsize=(20,10))
fit = SimpleExpSmoothing(ny[measure]).fit(smoothing_level=0.3, optimized=False)
xhat = fit.fittedvalues
plt.scatter(ny['date'], ny[measure], marker='.', color='black', s=100)
plt.plot(ny['date'], xhat, color='red')
plt.plot(lst, forecast, color='red')
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time in New York')
```

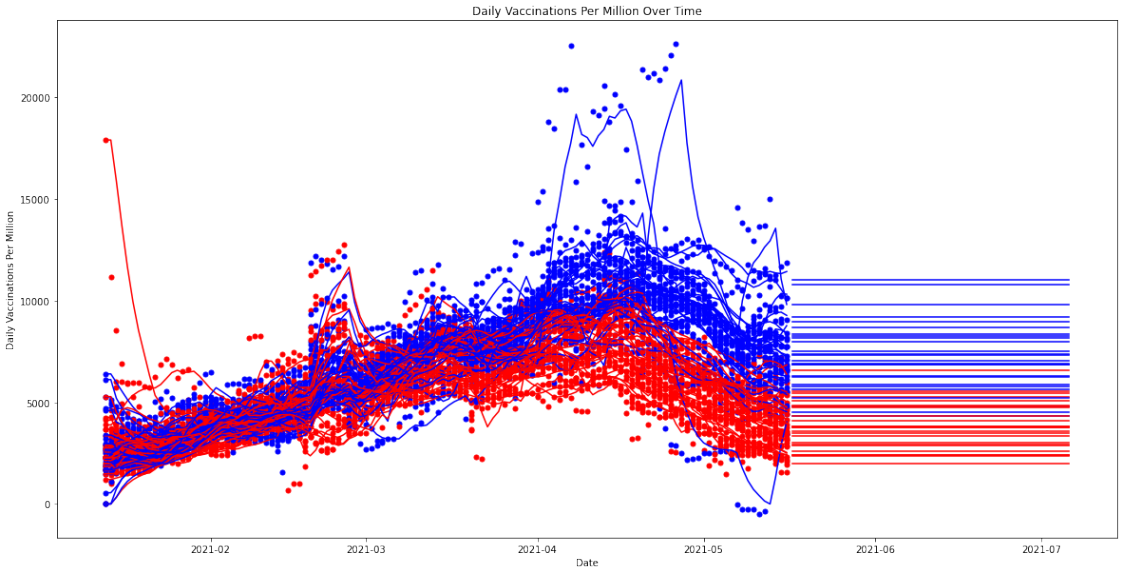
[77]: Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time in New York')



Doing the same as above but for all states, coloring by election results.

[78]:
```
plt.figure(figsize=(20,10))
for state in vac['location'].unique():
    df = vac[vac['location'] == state]
    df = df[['date', measure, 'Winner']]
    df['date'] = df['date'].apply(lambda x: pd.to_datetime(x))
    df = df.dropna(subset=[measure])
    fit = SimpleExpSmoothing(df[measure]).fit(smoothing_level=0.3,␣
 ↪optimized=False)
    xhat = fit.fittedvalues
    forecast = fit.forecast(L+1)
    plt.scatter(df['date'], df[measure], marker='.', color=df.
 ↪iloc[0]['Winner'], s=100)
    plt.plot(df['date'], xhat, color=df.iloc[0]['Winner'])
    plt.plot(lst, forecast, color=df.iloc[0]['Winner'])
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time')
```

[78]:
```
Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time')
```



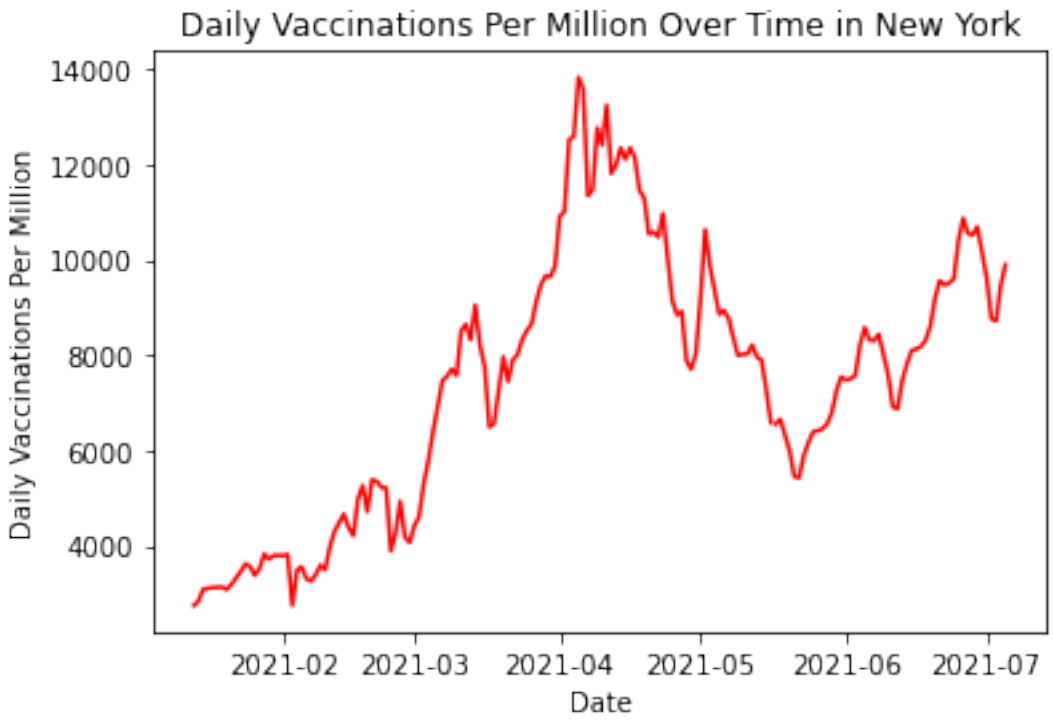
Seasonal periods for New York, might be interesting because the doeses are 3 weeks (21 periods) apart

[92]:
```
model = ExponentialSmoothing(ny[measure], trend='mul', seasonal='mul',␣
↪seasonal_periods=21).fit()
lookahead = 50
forecast = model.forecast(lookahead)
```

[93]:
```
L = lookahead
lst = [ny['date'].iloc[-1] + pd.Timedelta('1 day')]
for i in range(L-1):
    lst.append(lst[-1] + pd.Timedelta('1 day'))
```

[94]:
```
xhat = model.fittedvalues
plt.plot(ny['date'], xhat, color='red')
plt.plot(lst, forecast, 'r')
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time in New York')
```

[94]:
```
Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time in New York')
```



Finding the best model for the data

[104]:
```
# formally we could check all 4 options:
mse_min = 10000000000
t = None
s = None
period = None
for trend in ['add', 'mul']:
    for seasonal in ['add', 'mul']:
        for i in range(2,32):
            model = ExponentialSmoothing(df[measure], trend=trend,␣
 ↪seasonal=seasonal, seasonal_periods=i).fit()
            if model.sse < mse_min:
                mse_min, t, s, period = model.sse, trend, seasonal, i
            #print("Mean square error (trend="+trend+", period="+str(i)+",␣
 ↪seasonal="+seasonal+"):", model.sse)
```

[105]:
```
print(t,s,period)
```

```
add mul 6
```

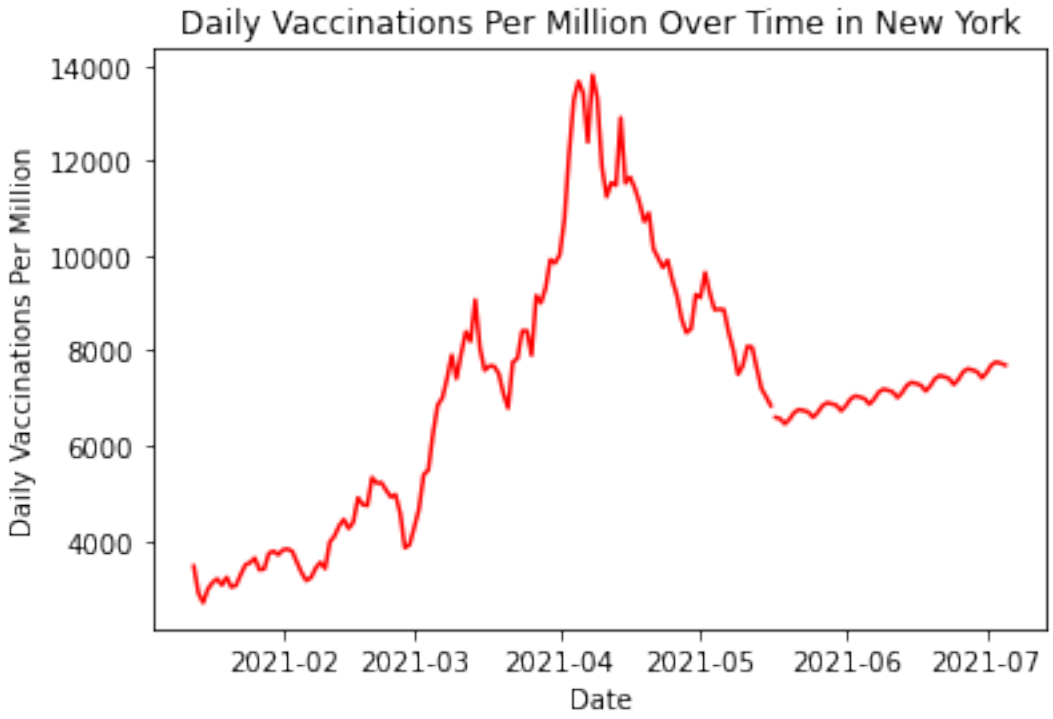
[106]:
```
model = ExponentialSmoothing(ny[measure], trend =t, seasonal=s,␣
 ↪seasonal_periods=period).fit()
lookahead = 50
```

```
forecast = model.forecast(lookahead)
```

[107]:
```
L = lookahead
lst = [ny['date'].iloc[-1] + pd.Timedelta('1 day')]
for i in range(L-1):
    lst.append(lst[-1] + pd.Timedelta('1 day'))
```

[108]:
```
xhat = model.fittedvalues
plt.plot(ny['date'], xhat, color='red')
plt.plot(lst, forecast, 'r')
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time in New York')
```

[108]:
```
Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time in New York')
```



Tried doing this for all states, having issues and not sure why (asked about it on Piazza)

[109]:
```
L = lookahead
lst = [ny['date'].iloc[-1] + pd.Timedelta('1 day')]
for i in range(L-1):
    lst.append(lst[-1] + pd.Timedelta('1 day'))
```

[115]:
```
plt.figure(figsize=(20,10))
for state in vac['location'].unique():
    df = vac[vac['location'] == state]
    df = df[['date', measure, 'Winner']]
    df['date'] = df['date'].apply(lambda x: pd.to_datetime(x))
    df = df.dropna(subset=[measure])
    try:
        model = ExponentialSmoothing(df[measure], seasonal='mul', trend='mul',␣
 ↪seasonal_periods=7).fit()
        xhat = model.fittedvalues
        forecast = model.forecast(lookahead)
        plt.scatter(df['date'], df[measure], marker='.', color=df.
 ↪iloc[0]['Winner'], s=100)
        plt.plot(df['date'], xhat, color=df.iloc[0]['Winner'])
        plt.plot(lst, forecast, color=df.iloc[0]['Winner'])
    except:
        print(state)
plt.ylabel('Daily Vaccinations Per Million')
plt.xlabel('Date')
plt.title('Daily Vaccinations Per Million Over Time')
```

```
Missouri
Nevada
New Hampshire
Wisconsin
```

[115]: Text(0.5, 1.0, 'Daily Vaccinations Per Million Over Time')

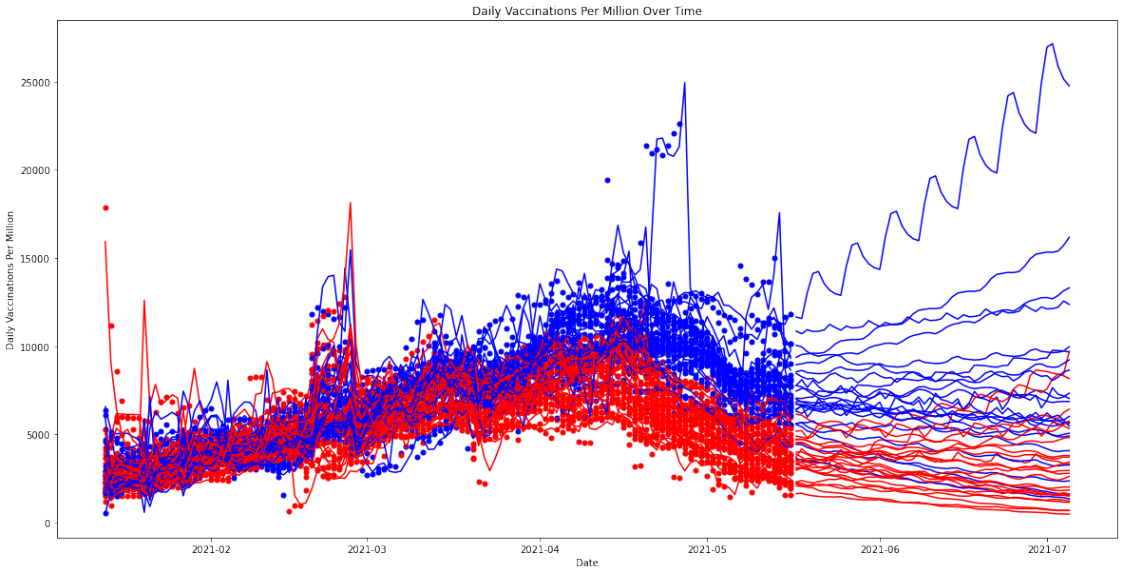


Tableau Code

May 21, 2021

[1]:
```
import pandas as pd
```

[31]:
```
df = pd.read_csv('us_state_vaccinations.csv')
vote = pd.read_csv('democratic_vs_republican_votes_by_usa_state_2020.csv')
df = df[df['date']=='2021-05-16']
combined = df.join(vote.set_index('state'), on ='location', how='inner')
combined['party'] = (combined['percent_democrat']/100).round()
combined.to_csv('for_tableau', index=False)
len(combined)
```

[31]:
```
51
```