testing_mortality_usa

June 1, 2020

0.1 Testing, prevalence of active cases, and mortality in Covid

This notebook considers several topics related to Covid mortality and SARS-CoV-2 virus testing, using data from the Covid Tracking project.

You can get the data by going to https://covidtracking.com/api, then downloading the CSV file under "historic state data". Make sure to save the data using file name "ctp_daily.csv". A direct link to the data is here:

https://covidtracking.com/api/v1/states/daily.csv

We will also use data about the population size for the US states. This can be obtained from various sources, we will get it directly from Wikipedia below.

Our main interest here will be to assess whether the reported testing data for active viral infections, using the PCR testing platform, are sufficient to explain the variation in Covid-related mortality. One of the main reasons to perform viral testing is to assess the prevalence of infections. As with any surveillance activity, it is not necessary to detect all cases to estimate the prevalence. It is sufficient to detect a fraction of the cases, and generalize to the full population. Unfortunately, SARS-CoV-2 testing has been performed extremely haphazardly – a very non-representative subpopulation has been tested. Therefore, we will not actually be aiming literally for the prevalence here (e.g. cases per thousand population). Instead, we take the point of view that the question of whether `enough'' testing has been done can largely be settled by assessing to what extent testing results predict mortality, which is the main`hard endpoint" for Covid.

We note that there are other possible uses of testing such as for use in contact tracing, identifying emerging disease hotspots, and assessing cumulative prevalence of past exposure to the virus. These are distinct goals from prevalence estimation, and likely would require much more extensive and more systematic testing. These goals are not considered further here.

[1]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import os
```

[2]:
```
# Below we get state population data from Wikipedia.  We only download
# the file one time and store it locally as 'states.csv'.
if not os.path.exists("states.csv"):

    # Map full state names to postal codes
```

```
   dm = {"Alabama": "AL", "Alaska": "AK", "Arizona": "AZ", "Arkansas": "AR",␣
↪"California": "CA",
       "Colorado": "CO", "Connecticut": "CT", "Delaware": "DE", "District of␣
↪Columbia": "DC",
       "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", "Puerto Rico":␣
↪"PR", "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"}

   ta = pd.read_html("https://simple.m.wikipedia.org/wiki/List_of_U.S.
↪_states_by_population",
                     attrs={"class": "wikitable sortable"})
   ta = ta[0].iloc[:, 2:4]
   ta["SA"] = [dm.get(x, "") for x in ta.State]
   ta = ta.loc[ta.SA != "", :]
   ta.columns = ["state", "pop", "SA"]
   ta = ta.loc[:, ["state", "SA", "pop"]]
   ta = ta.sort_values(by="state")
   ta.to_csv("states.csv", index=None)
```

Load and merge the Covid data and the state population data.

[3]:
```
df = pd.read_csv("ctp_daily.csv")
dp = pd.read_csv("states.csv")
dp = dp.drop("state", axis=1)
df = pd.merge(df, dp, left_on="state", right_on="SA", how="left")
df["pdate"] = pd.to_datetime(df.date, format="%Y%m%d")
dx = df.loc[:, ["date", "pdate", "state", "positive", "negative", "death",␣
 ↪"pop"]].dropna()
dx = dx.sort_values(by=["state", "pdate"])
```

The data as provided are cumulative, so we difference it to get the daily values.

[4]:
```
dx["ddeath"] = dx.groupby("state")["death"].diff()
dx["dpositive"] = dx.groupby("state")["positive"].diff()
```

```
dx["dnegative"] = dx.groupby("state")["negative"].diff()
dx = dx.dropna()
```

0.2 Relationship between testing results and mortality.

The Covid Tracking project reports cumulative daily counts of Covid-related deaths for each US state, and also reports the cumulative numbers of positive and negative Covid tests. The tests reported are primarily PCR tests that assess for viral RNA, which should indicate an active or recently cleared SARS-CoV-2 infection. The number of positive tests and the number of negative tests are reported for each US state and for each day.

There is a lot of discussion and debate about the characteristics of PCR testing for SARS-CoV-2. One useful reference for this topic is:

https://asm.org/Articles/2020/April/False-Negatives-and-Reinfections-the-Challenges-of

To summarize:

- It is currently unknown for how long after infection, symptom onset, or recovery a person may continue to have positive PCR test values. It is likely that this duration could be from 1 to 4 weeks in most cases.
- The PCR test is quite sensitive, approaching 100% in lab conditions, but false negatives can result from flawed sample collection.
- The PCR test is quite specific in the narrow sense – if it detects viral RNA, then viral RNA was probably present. However it is possible that PCR may detect residual non-pathogenic RNA remnants from a cleared infection.
- Viral load is highest early in the course of the infection, including during the pre-symptomatic or asymptomatic period. People with severe Covid complications that persist several weeks after symptom onset may have minimal viral load by that time, and are primarily impacted by the after-effects of their immune response to the virus.
- It is currently unknown what fraction of the many people with minimal or no symptoms following SARS-CoV-2 exposure will receive a test and then test positive via PCR.
- Antibody testing detects past infection, which in most cases will have resolved to the point where a concurrent PCR test would be negative. We only have basic summaries of the antibody testing that has been done to date. Reportedly, a few states have counted antibody tests among the PCR tests. We do not attempt to account for that here.

We should also keep in mind that each US state follows different practices for testing and reporting. For mortality data, there may be differences in which deaths are deemed to be Covid-associated. There are likely to be substantial undercounts, for example some states during some periods of time only reported deaths in hospitals. But there could be some overcounting as well, e.g. a person with multiple severe illnesses, including Covid, may not have died primarily due to Covid.

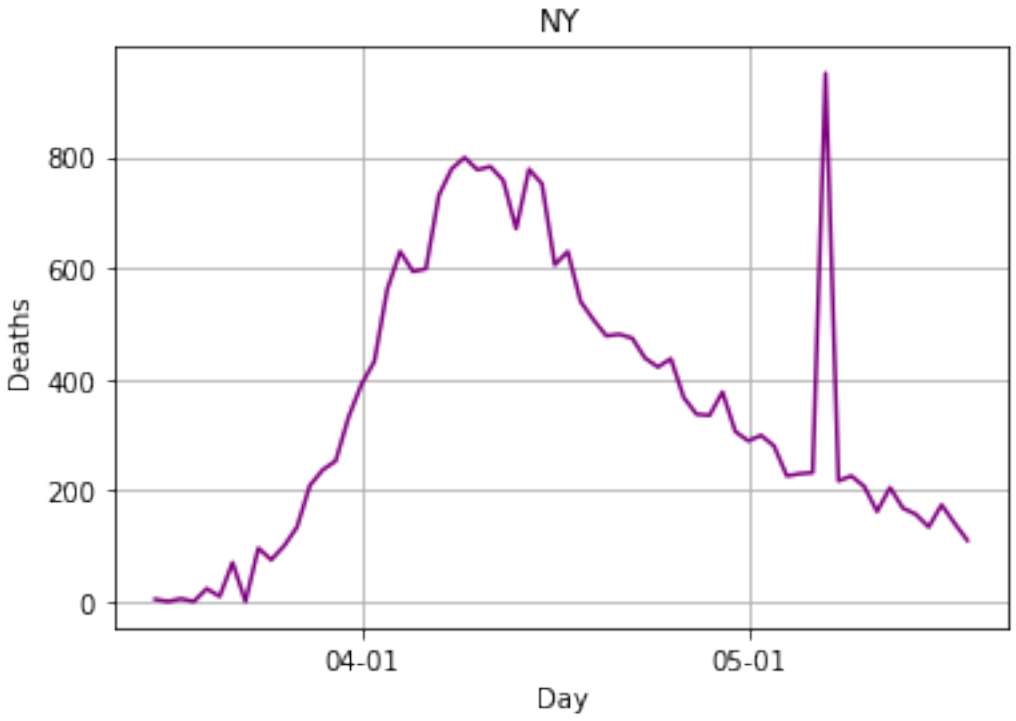
For the testing data, each state has its own policies about who can get tested. Early in the epidemic testing was severely constrained by the availability of test kits. Not all states have consistently reported negative test results, but reporting of test positives is presumably somewhat more consistent. The sample of tested people is not representative of the general population, and

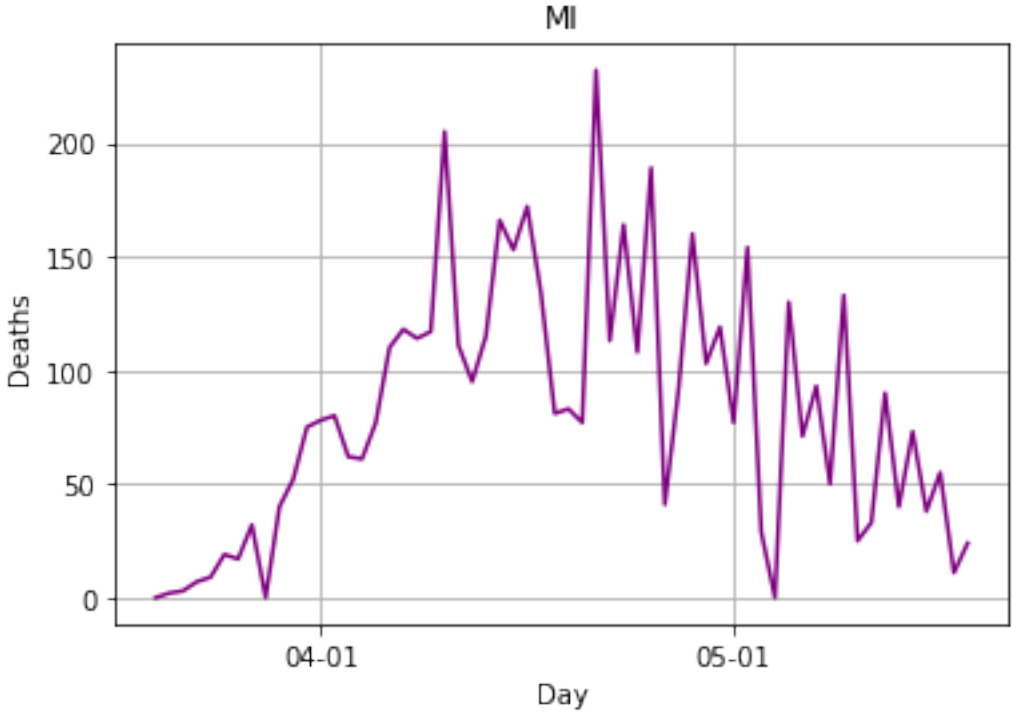
is likely a heterogeneous mix of health-care workers and people with Covid symptoms It is certainly enriched for cases.

Below are plots of the daily death counts for New York and for Michigan:

[5]:
```
fmt = mdates.DateFormatter("%m-%d")
months = mdates.MonthLocator()

for st in "NY", "MI":
    da = dx.loc[dx.state==st, :]
    plt.clf()
    plt.grid(True)
    plt.plot(da.pdate.values, da.ddeath.values, '-', color='purple')
    plt.title(st)
    plt.xlabel("Day")
    plt.ylabel("Deaths")
    plt.gca().xaxis.set_major_formatter(fmt)
    plt.gca().xaxis.set_major_locator(months)
    plt.show()
```

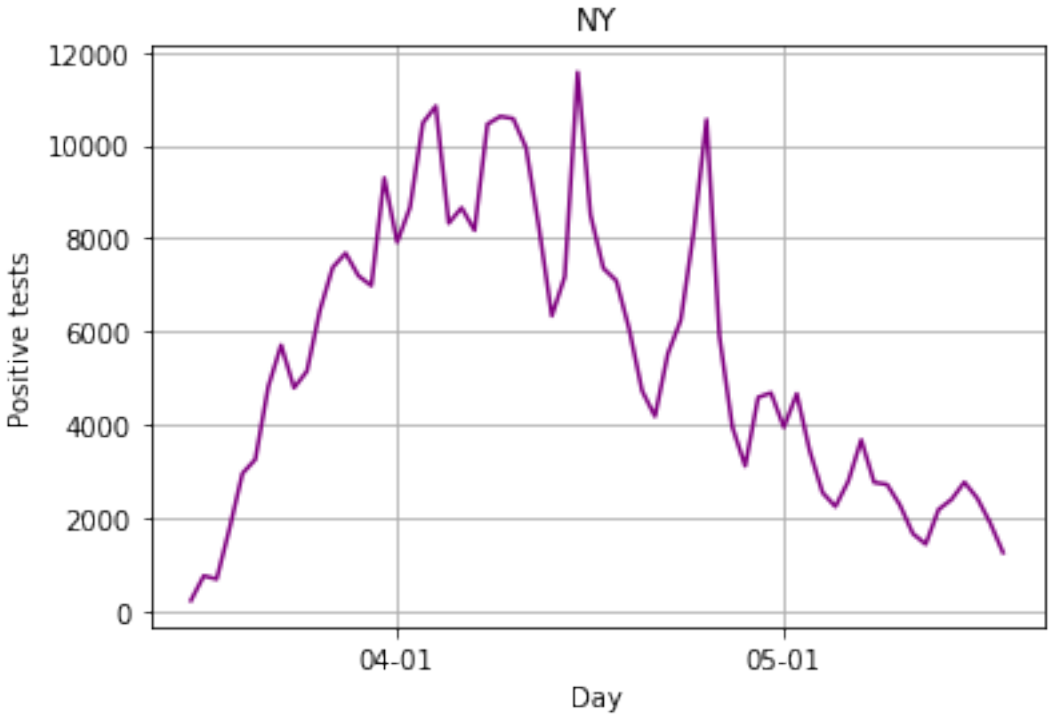
```
/nfs/kshedden/python3/lib/python3.7/site-
packages/pandas/plotting/_matplotlib/converter.py:103: FutureWarning: Using an
implicitly registered datetime converter for a matplotlib plotting method. The
converter was registered by pandas on import. Future versions of pandas will
require you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

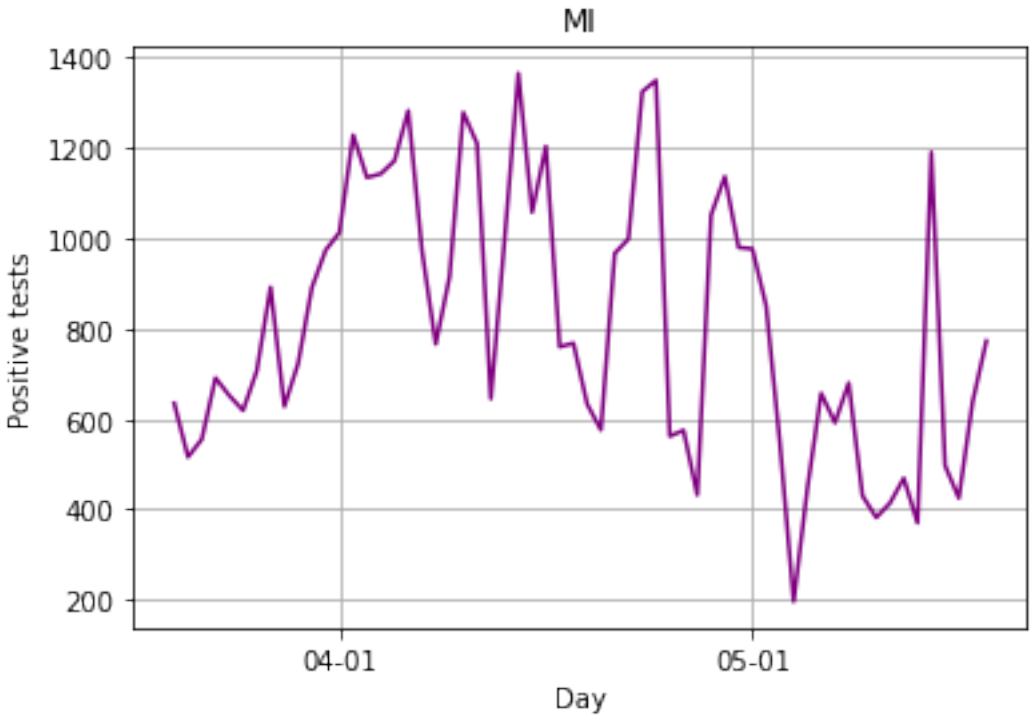




Below are plots of the daily number of positive tests for New York and for Michigan:

[6]:
```
for st in "NY", "MI":
    da = dx.loc[dx.state==st, :]
    plt.clf()
    plt.grid(True)
    plt.plot(da.pdate.values, da.dpositive.values, '-', color='purple')
    plt.title(st)
    plt.xlabel("Day")
    plt.ylabel("Positive tests")
    plt.gca().xaxis.set_major_formatter(fmt)
    plt.gca().xaxis.set_major_locator(months)
    plt.show()
```





Our main analysis will use a regression approach to relate SARS-CoV-2 testing results to Covid mortality. As noted above, the motivation for this analysis is that the people who test positive can be viewed in some way as representing the prevalence of active infections (with many caveats, to be discussed below). Thus, understanding the statistical relationship between testing results and Covid mortality can be informative about the relationship between exposure and mortality.

Since people who die from Covid typically were infected several weeks prior to their death, we will create counts of testing positives and negatives in several windows of width one week, lagging behind the mortality count.

[7]:
```
# Sum x from d2 days back in time to d1 days back in time, inclusive of
# both endpoints.  d2 must be greater than d1.
def wsum(x, d1, d2):
    w = np.ones(d2 + 1)
    if d1 > 0:
        w[-d1:] = 0
    y = np.zeros_like(x)
    y[d2:] = np.convolve(x.values, w[::-1], mode='valid')
    return y

for j in range(4):
    dx["cumpos%d" % j] = dx.groupby("state").dpositive.transform(lambda x:␣
 ↪wsum(x, 7*j, 7*j+6))
    dx["cumneg%d" % j] = dx.groupby("state").dnegative.transform(lambda x:␣
 ↪wsum(x, 7*j, 7*j+6))
    dx["logcumpos%d" % j] = np.log(dx["cumpos%d" % j] + 1)
```

```
    dx["logcumneg%d" % j] = np.log(dx["cumneg%d" % j] + 1)
```

```
/nfs/kshedden/python3/lib/python3.7/site-packages/pandas/core/series.py:853:
RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Below we will use regression analysis to try to understand how SARS-CoV-2 testing results relate to Covid mortality. A reasonable hypothesis would be that a greater number of positive test results predicts greater mortality, at a 1-4 week lag. More specifically, all else held equal, comparing two places/times where the Covid prevalence differs by a factor of two over the past month, we expect there to be a factor of two difference in Covid mortality today.

Another reasonable hypothesis would be that the ratio of positive to negative test results, rather than the absolute number of positive test results, could be a stronger predictor of mortality. This is based on the logic that taking this ratio corrects for the fact that states may increase their testing when a big surge in cases is expected.

There are several other factors that we should consider and account for if possible:

- The virus arrived in different states at different times, e.g. it arrived in New York before it arrived in South Dakota.
- States vary greatly in terms of population size. It is reasonable to expect death counts to scale with population size.
- Transmission rates may vary by state, e.g. due to population density.
- The seasons were changing just as the epidemic in the US reached its peak. Spring weather may reduce transmission rates.
- The infection/fatality ratio (IFR) may vary by state due to demographic characteristics of the population and prevalence of comorbidities.

To account for differences in the time when the disease arrived in each state, we identify the date of the first Covid death, then include only the data starting 10 days after that date.

[8]:
```
def firstdeath(x):
    if (x.death == 0).all():
        return np.inf
    ii = np.flatnonzero(x.death > 0)[0]
    return x.pdate.iloc[ii]

xx = dx.groupby("state").apply(firstdeath)
xx.name = "firstdeath"
dx = pd.merge(dx, xx, left_on="state", right_index=True)

dx["rdate"] = (dx.pdate - dx.firstdeath).dt.days
dx = dx.loc[dx.rdate >= 10, :]
```

Below is an initial regression analysis looking at mortality counts per state/day as an outcome that is predicted by testing results. We also include state level fixed effects to control for the possibility of different infection/fatality ratios among the states. The model is fit using a Poisson generalized estimating equations (GEE) approach, clustering the data by state. This allows data values in the same state to be correlated, e.g. through a serial correlation with respect to time.

[9]:
```
fml = "ddeath ~ 0 + C(state) + "
fml += " + ".join(["logcumpos%d" % j for j in range(4)])
fml += " + "
fml += " + ".join(["logcumneg%d" % j for j in range(4)])
m1 = sm.GEE.from_formula(fml, groups="state", data=dx, family=sm.families.
 ↪Poisson())
r1 = m1.fit(scale="X2")
print(r1.summary())
```

```
                               GEE Regression Results
==============================================================================
===
Dep. Variable:                       ddeath   No. Observations:
2509
Model:                                 GEE   No. clusters:
52
Method:                        Generalized   Min. cluster size:
26
                      Estimating Equations   Max. cluster size:
69
Family:                           Poisson   Mean cluster size:
48.2
Dependence structure:         Independence   Num. iterations:
2
Date:                     Mon, 01 Jun 2020   Scale:
10.719
Covariance type:                    robust   Time:
14:04:51
==============================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
C(state)[AK]    -3.9572      0.559     -7.084      0.000      -5.052      -2.862
C(state)[AL]    -2.3853      0.758     -3.149      0.002      -3.870      -0.900
C(state)[AR]    -3.2846      0.698     -4.704      0.000      -4.653      -1.916
C(state)[AZ]    -2.1664      0.749     -2.891      0.004      -3.635      -0.698
C(state)[CA]    -1.7785      0.904     -1.968      0.049      -3.549      -0.007
C(state)[CO]    -2.0133      0.750     -2.685      0.007      -3.483      -0.544
C(state)[CT]    -1.3431      0.783     -1.716      0.086      -2.877       0.191
C(state)[DC]    -2.3762      0.658     -3.612      0.000      -3.665      -1.087
C(state)[DE]    -2.7012      0.660     -4.096      0.000      -3.994      -1.409
C(state)[FL]    -1.8806      0.864     -2.177      0.029      -3.574      -0.187
C(state)[GA]    -2.0407      0.812     -2.514      0.012      -3.632      -0.450
C(state)[HI]    -3.2679      0.533     -6.132      0.000      -4.312      -2.223
C(state)[IA]    -2.8741      0.756     -3.800      0.000      -4.357      -1.392
C(state)[ID]    -3.0508      0.607     -5.030      0.000      -4.240      -1.862
C(state)[IL]    -1.7109      0.884     -1.935      0.053      -3.444       0.022
C(state)[IN]    -1.7935      0.784     -2.289      0.022      -3.329      -0.258
```

C(state)[KS]	-3.2214	0.707	-4.557	0.000	-4.607	-1.836
C(state)[KY]	-2.4639	0.722	-3.411	0.001	-3.880	-1.048
C(state)[LA]	-1.4980	0.811	-1.846	0.065	-3.088	0.092
C(state)[MA]	-1.3724	0.869	-1.578	0.114	-3.077	0.332
C(state)[MD]	-1.8578	0.803	-2.313	0.021	-3.432	-0.284
C(state)[ME]	-3.0739	0.509	-6.040	0.000	-4.071	-2.076
C(state)[MI]	-1.0514	0.818	-1.286	0.198	-2.654	0.551
C(state)[MN]	-2.0673	0.795	-2.600	0.009	-3.626	-0.509
C(state)[MO]	-2.1363	0.726	-2.942	0.003	-3.560	-0.713
C(state)[MS]	-2.2905	0.737	-3.108	0.002	-3.735	-0.846
C(state)[MT]	-3.1373	0.492	-6.380	0.000	-4.101	-2.174
C(state)[NC]	-2.3233	0.799	-2.907	0.004	-3.890	-0.757
C(state)[ND]	-3.4082	0.666	-5.118	0.000	-4.713	-2.103
C(state)[NE]	-3.7065	0.737	-5.032	0.000	-5.150	-2.263
C(state)[NH]	-2.5368	0.646	-3.930	0.000	-3.802	-1.272
C(state)[NJ]	-1.3087	0.846	-1.546	0.122	-2.968	0.350
C(state)[NM]	-2.3822	0.733	-3.251	0.001	-3.818	-0.946
C(state)[NV]	-2.3970	0.687	-3.489	0.000	-3.744	-1.050
C(state)[NY]	-0.9984	0.956	-1.044	0.296	-2.872	0.875
C(state)[OH]	-1.7642	0.808	-2.184	0.029	-3.348	-0.181
C(state)[OK]	-2.3560	0.678	-3.475	0.001	-3.685	-1.027
C(state)[OR]	-2.8056	0.686	-4.089	0.000	-4.150	-1.461
C(state)[PA]	-1.3981	0.847	-1.651	0.099	-3.058	0.262
C(state)[PR]	-2.7663	0.549	-5.036	0.000	-3.843	-1.690
C(state)[RI]	-2.3488	0.754	-3.113	0.002	-3.827	-0.870
C(state)[SC]	-2.4226	0.711	-3.410	0.001	-3.815	-1.030
C(state)[SD]	-4.1450	0.650	-6.381	0.000	-5.418	-2.872
C(state)[TN]	-3.0652	0.810	-3.785	0.000	-4.652	-1.478
C(state)[TX]	-2.2932	0.877	-2.616	0.009	-4.011	-0.575
C(state)[UT]	-3.7992	0.752	-5.051	0.000	-5.273	-2.325
C(state)[VA]	-2.3767	0.798	-2.976	0.003	-3.942	-0.812
C(state)[VT]	-2.7995	0.556	-5.031	0.000	-3.890	-1.709
C(state)[WA]	-2.0937	0.784	-2.669	0.008	-3.631	-0.556
C(state)[WI]	-2.4273	0.754	-3.219	0.001	-3.905	-0.950
C(state)[WV]	-2.7004	0.656	-4.118	0.000	-3.986	-1.415
C(state)[WY]	-5.3686	0.578	-9.285	0.000	-6.502	-4.235
logcumpos0	0.6193	0.084	7.370	0.000	0.455	0.784
logcumpos1	0.1648	0.035	4.644	0.000	0.095	0.234
logcumpos2	0.0361	0.033	1.102	0.271	-0.028	0.100
logcumpos3	-0.0363	0.041	-0.875	0.381	-0.118	0.045
logcumneg0	-0.0695	0.033	-2.103	0.035	-0.134	-0.005
logcumneg1	-0.0758	0.031	-2.428	0.015	-0.137	-0.015
logcumneg2	0.0027	0.026	0.103	0.918	-0.049	0.054
logcumneg3	0.0378	0.031	1.212	0.225	-0.023	0.099

Skew:	5.6410	Kurtosis:	105.1739
Centered skew:	5.6410	Centered kurtosis:	105.1739

A perfect 1-1 relationship between testing and mortality would manifest as a perfect linear relationship between the log number of cases and the log number of positive tests. In this case, we would expect the regression coefficients for the log number of positive tests to be equal to 1 (assuming that cases and mortality are 1-1 at some lag). As we see above, the coefficient for the log number of positive tests in the week immediately preceding the mortality count is around 0.56, and the coefficients for the two weeks prior to that were 0.19 and 0.04 respectively. Note that these coefficients sum to around 0.8 – if the positive tests in two states/4 week time periods differ by a factor of two, the state/time with more positive tests will have around 80% more deaths. The fact that these effects are spread across several lagged terms may reflect variation in the timing of death relative to infection.

Note also that there is a statistically significant inverse relationship between the log number of negative tests in the prior weeks and mortality. However the coefficients of the negative test result variables do not sum to a value close to 1. This would argue against using the ratio of the number of positive to negative tests as a predictor of mortality.

As noted above, only a fraction of the people who have been exposed to SARS-CoV-2 virus are tested during the period of time when they would have a positive PCR test. There is an unknown scaling factor called the "ascertainment ratio" between the number of exposed people and the number of people who are tested and found to have an active infection. The results here do not tell us anything about this ratio. Instead, the results here tell us that the testing in the U.S., with all its flaws, is sufficiently comprehensive and representative to capture around 80% of Covid mortality. More representative testing, which may or may not involve a greater volume of testing, would potentially capture 100% of Covid mortality.

0.3 State effects and variation in the IFR

The analysis above uses state-level fixed effects to account for state-to-state differences from various sources. From a statistical perspective, this is one of several ways to address this issue. Including large numbers of fixed effects in any regression model can raise concerns about there being sufficient data to estimate these parameters. This is classically known as the "Neyman-Scott" problem. Alternatives to fixed effects include mutilevel analysis, shrunken fixed effects, and marginal analysis with covariances within states (which we have also done here via GEE).

If we believe that the positive and negative testing data are sufficiently informative about the prevalence of the disease at each state/time, then the state fixed effects in the above model might reflect state-to-state differences in the infection/fatality ratio. The analysis below shows that these state effects have a range of around 4.4 on the log scale, with a standard deviation of 0.82. Since exp(0.82) ~ 2.27, this might suggest that most states have IFR values that are within a factor of around 2.5 of the mean state-level IFR – however, see below for a more refined analysis. Variation on the order of 2 or more would be quite interesting because it would suggest a major contribution of some state-level factor to Covid mortality.

Although there is substantial heterogeneity among the US states, 2-fold or greater differences in mortality to SARS-CoV-2 infection would be large. For reasons discussed below, it turns out that this factor of 2.5 is likely an over-estimate, and the true extent of state-to-state differences is likely much smaller.

[10]:
```
# Extract the state fixed effects
pa = r1.params
st = [y for x, y in zip(pa.index, pa.values) if "state" in x]
st = np.asarray(st)
```

```
print("Range of state effects:")
print(st.min(), st.max())

print("Unadjusted SD of state effects:")
print(st.std())
```

```
Range of state effects:
-5.368611589075238 -0.99837848022408
Unadjusted SD of state effects:
0.8253747958120818
```

The state fixed effects discussed above are estimates, not exact values. The estimation errors in these quantities inflate their variance when estimated naively as in the preceding cell. This can be addressed as shown below. However the results show that the bias is not great in this case, and the relationship between testing and mortality may commonly differ from state-to-state by factors on the order of 2-3.

[11]:
```
ii = [i for i, x in enumerate(pa.index) if "state" in x]
c = r1.cov_params().iloc[ii, ii]

# This is a method of moments adjustment to remove the extra variance
# due to parameter estimation.  We do not provide all the details
# here.
p = len(st)
oo = np.ones(p)
qm = np.eye(p) - np.outer(oo, oo) / p # Centering matrix
bias = np.trace(np.dot(qm, np.dot(c, qm)))
v = np.dot(st, np.dot(qm, st)) - bias
v /= p

print("Adjusted variance and SD of state effects:")
print("variance=%f, SD=%f\n" % (v, np.sqrt(v)))
```

```
Adjusted variance and SD of state effects:
variance=0.668018, SD=0.817324
```

0.4 Alternative models and confounding

As with any observational data set, there are many opportunities for confounding to mislead us. One such possibility is that all the US states have progressed through the Covid epidemic over roughly the same time period, and during this time the weather became much warmer in most of the US. In fact, weather is just one possible confounder indexed by time (e.g. standards of care have evolved and may have impacted the IFR). To address this possibility, we fit models in which calendar date, or date relative to the state's first Covid death are included as controls. We find that the coefficients for the positive and negative testing results are relatively invariant to inclusion of these terms. Also, based on a score test, the inclusion of rdate (the number of days within a state since that state's first recorded Covid death) is not statistically significant.

[12]:
```
fml = "ddeath ~ C(state) + bs(rdate, 5) + "
fml += " + ".join(["logcumpos%d" % j for j in range(4)])
fml += " + "
fml += " + ".join(["logcumneg%d" % j for j in range(4)])
m2 = sm.GEE.from_formula(fml, groups="state", data=dx, family=sm.families.
 ↪Poisson())
r2 = m2.fit(scale="X2")
print(r2.summary())
print("Score test for model 2 relative to model 1:")
print(m2.compare_score_test(r1))
```

```
                               GEE Regression Results
==============================================================================
===
Dep. Variable:                      ddeath   No. Observations:
2509
Model:                                GEE   No. clusters:
52
Method:                       Generalized   Min. cluster size:
26
                      Estimating Equations   Max. cluster size:
69
Family:                           Poisson   Mean cluster size:
48.2
Dependence structure:         Independence   Num. iterations:
2
Date:                     Mon, 01 Jun 2020   Scale:
10.553
Covariance type:                    robust   Time:
14:04:51
==============================================================================
===
                      coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---
Intercept          -3.6364      0.596     -6.101      0.000      -4.805
-2.468
C(state)[T.AL]      1.5671      0.297      5.282      0.000       0.986
2.149
C(state)[T.AR]      0.5928      0.226      2.626      0.009       0.150
1.035
C(state)[T.AZ]      1.7078      0.320      5.332      0.000       1.080
2.336
C(state)[T.CA]      2.1918      0.466      4.701      0.000       1.278
3.106
C(state)[T.CO]      1.8028      0.377      4.784      0.000       1.064
```

```
2.541
C(state)[T.CT]      2.5015      0.419      5.969      0.000       1.680
3.323
C(state)[T.DC]      1.3649      0.336      4.067      0.000       0.707
2.023
C(state)[T.DE]      1.0823      0.327      3.306      0.001       0.441
1.724
C(state)[T.FL]      2.0772      0.415      5.000      0.000       1.263
2.891
C(state)[T.GA]      1.8549      0.401      4.623      0.000       1.068
2.641
C(state)[T.HI]      0.6541      0.064     10.226      0.000       0.529
0.780
C(state)[T.IA]      0.9531      0.355      2.681      0.007       0.256
1.650
C(state)[T.ID]      0.8176      0.162      5.036      0.000       0.499
1.136
C(state)[T.IL]      2.2059      0.483      4.566      0.000       1.259
3.153
C(state)[T.IN]      2.0530      0.390      5.261      0.000       1.288
2.818
C(state)[T.KS]      0.5391      0.330      1.632      0.103      -0.108
1.187
C(state)[T.KY]      1.3631      0.284      4.801      0.000       0.807
1.920
C(state)[T.LA]      2.4022      0.397      6.058      0.000       1.625
3.179
C(state)[T.MA]      2.5490      0.472      5.403      0.000       1.624
3.474
C(state)[T.MD]      1.9894      0.422      4.714      0.000       1.162
2.817
C(state)[T.ME]      0.5319      0.439      1.213      0.225      -0.328
1.392
C(state)[T.MI]      2.8596      0.415      6.891      0.000       2.046
3.673
C(state)[T.MN]      1.7617      0.365      4.820      0.000       1.045
2.478
C(state)[T.MO]      1.7374      0.297      5.842      0.000       1.154
2.320
C(state)[T.MS]      1.5470      0.316      4.895      0.000       0.928
2.166
C(state)[T.MT]      0.7337      0.085      8.641      0.000       0.567
0.900
C(state)[T.NC]      1.6344      0.337      4.854      0.000       0.975
2.294
C(state)[T.ND]      0.4776      0.162      2.943      0.003       0.160
0.796
C(state)[T.NE]      0.1073      0.344      0.312      0.755      -0.567
```

```
0.782
C(state)[T.NH]      1.2577      0.239      5.266      0.000       0.790
1.726
C(state)[T.NJ]      2.5367      0.533      4.763      0.000       1.493
3.581
C(state)[T.NM]      1.5008      0.251      5.979      0.000       1.009
1.993
C(state)[T.NV]      1.4198      0.276      5.144      0.000       0.879
1.961
C(state)[T.NY]      2.9824      0.585      5.096      0.000       1.835
4.130
C(state)[T.OH]      2.1323      0.379      5.621      0.000       1.389
2.876
C(state)[T.OK]      1.4743      0.255      5.777      0.000       0.974
1.975
C(state)[T.OR]      1.0541      0.203      5.200      0.000       0.657
1.451
C(state)[T.PA]      2.5078      0.446      5.617      0.000       1.633
3.383
C(state)[T.PR]      0.8880      0.415      2.137      0.033       0.074
1.702
C(state)[T.RI]      1.5435      0.322      4.796      0.000       0.913
2.174
C(state)[T.SC]      1.4106      0.288      4.899      0.000       0.846
1.975
C(state)[T.SD]     -0.4191      0.308     -1.361      0.173      -1.022
0.184
C(state)[T.TN]      0.8807      0.334      2.638      0.008       0.226
1.535
C(state)[T.TX]      1.6655      0.426      3.909      0.000       0.830
2.501
C(state)[T.UT]      0.1376      0.258      0.534      0.593      -0.368
0.643
C(state)[T.VA]      1.4576      0.407      3.577      0.000       0.659
2.256
C(state)[T.VT]      1.0287      0.129      7.975      0.000       0.776
1.282
C(state)[T.WA]      1.8496      0.325      5.686      0.000       1.212
2.487
C(state)[T.WI]      1.4311      0.313      4.572      0.000       0.818
2.044
C(state)[T.WV]      1.2946      0.128     10.118      0.000       1.044
1.545
C(state)[T.WY]     -1.4379      0.115    -12.483      0.000      -1.664
-1.212
bs(rdate, 5)[0]     0.0707      0.364      0.194      0.846      -0.643
0.784
bs(rdate, 5)[1]     0.4842      0.474      1.021      0.307      -0.446
```

```
1.414
bs(rdate, 5)[2]     0.4494      0.467      0.963      0.336      -0.466
1.364
bs(rdate, 5)[3]     0.8860      0.718      1.235      0.217      -0.520
2.292
bs(rdate, 5)[4]    -0.2159      0.374     -0.577      0.564      -0.949
0.517
logcumpos0          0.6344      0.101      6.260      0.000       0.436
0.833
logcumpos1          0.1770      0.045      3.912      0.000       0.088
0.266
logcumpos2          0.0334      0.031      1.082      0.279      -0.027
0.094
logcumpos3         -0.0456      0.035     -1.320      0.187      -0.113
0.022
logcumneg0         -0.1116      0.062     -1.788      0.074      -0.234
0.011
logcumneg1         -0.1004      0.051     -1.970      0.049      -0.200
-0.000
logcumneg2         -0.0121      0.027     -0.447      0.655      -0.065
0.041
logcumneg3          0.0316      0.026      1.213      0.225      -0.019
0.083
==============================================================================
Skew:                          5.4725   Kurtosis:                       104.0474
Centered skew:                 5.4725   Centered kurtosis:              104.0474
==============================================================================
Score test for model 2 relative to model 1:
{'statistic': 5.6049392193251375, 'df': 5, 'p-value': 0.34657606840474964}
```

As noted above, it makes sense to model death relative to population size. Since the Poisson regression that we will be using has a log link function, to properly account for population effects we should use the log of the total population as an offset, or as a covariate. Note however that Covid has largely arisen in geographically-limited clusters that are smaller than an entire state. For this reason, the state population may not be the perfect offset to use for this purpose. Ideally, we would have mortality and testing data at a finer geographical scale, for example by county. But that data is not available now.

[13]:
```
dx["lpop"] = np.log(dx["pop"])
```

Since population size is a state-level variable and we already have included state fixed effects in the model, a main effect of population size has already been accounted for in the models above. However we can take the question of population scaling a bit further by considering interactions between population size and the positive test counts. As shown below however, these coefficients are not statistically significant based on the score test.

[14]:
```
dx["lpop_cen"] = dx.lpop - dx.lpop.mean()
fml = "ddeath ~ C(state) + bs(rdate, 5) + "
fml += " + ".join(["lpop_cen*logcumpos%d" % j for j in range(4)])
fml += " + "
```

```
fml += " + ".join(["lpop_cen*logcumneg%d" % j for j in range(4)])
m3 = sm.GEE.from_formula(fml, groups="state", data=dx, family=sm.families.
 ↪Poisson())
r3 = m3.fit(scale="X2")
print("Score test for model 3 relative to model 2:")
print(m3.compare_score_test(r2))
```

```
Score test for model 3 relative to model 2:
{'statistic': 13.31378323202196, 'df': 9, 'p-value': 0.1489142958032137}
```

0.5 Dispersion relative to the mean and the scale parameter

Above we focused on the mean structure of the model, which is reflected in the slope parameters for the covariates. These parameters determine the expected mortality count for any given state/date pair. We should also consider to what extent the data are scattered with respect to this mean. This is captured through the scale parameter of the quasi-Poisson regression.

[15]:
```
print(r1.scale)
print(r2.scale)
print(r3.scale)
```

```
10.719056894065115
10.552970538459272
10.409096549371005
```

In an ideal Poisson regression, the scale parameter is equal to 1, meaning that the conditional variance is equal to the conditional mean. Here we are seeing that the conditional variance is around 10 times greater than the conditional mean.

The conventional approach for estimating the scale parameter is analogous to the sample variance, using the "Pearson residuals".

[16]:
```
resid = r2.resid_pearson
print(np.var(resid))

# This is how the Pearson residuals are constructed:
f = np.exp(np.dot(m2.exog, r2.params))
resid0 = (m2.endog - f) / np.sqrt(f)

assert(np.all(np.abs(resid - resid0) < 1e-10))
```

```
10.278036657228622
```

As we saw above, there are some large outliers in the mortality data, and we have good reason to believe that these outliers do not reflect actual daily death counts. Therefore it might make sense to use an alternative approach to calculating the scale parameter that is not too strongly influenced by a few outlying values. One way to do this is using "Huber's proposal 2" estimate of the scale. This estimator solves a modified ("clipped") version of the moment-based estimating equation solved by the usual sample variance.

Huber's scale estimate is around 4.15, indicating how the usual scale estimate is inflated by a few large outliers. Thus, we see that the conditional variance in the number of Covid deaths is around 4-10 times greater than the conditional mean number of Covid deaths, depending on how we estimate the scale parameter.

[17]:
```
def g(s, x=resid):
    c = 1.345
    return np.sum(np.clip(x / s, -c, c)**2)

# This is around 0.84, slightly less than 1 due
# to clipping.
gam = g(1, np.random.normal(size=50000)) / 50000

# This is the equation that we wish to solve to estimate
# the scale parameter s.
def h(s):
    return g(s).sum() - gam*(m2.nobs - r2.params.size)

# Left bracket
s0 = r2.scale
while h(s0) < 0:
    s0 /= 2

# Right bracket
s1 = r2.scale
while h(s1) > 0:
    s1 *= 2

from scipy.optimize import root_scalar

scale_huber = root_scalar(h, bracket=[s0, s1]).root**2
```

In a perfect Poisson situation, the variance would be equal to the mean. This perfect Poisson behavior would arise if we had independence and homogeneity – independence meaning that any two people living in the same state on the same day die of Covid independently of each other, and homogeneity meaning that any two people living in the same state on the same day have the same probability of dying of Covid. The independence condition is likely to hold, but the homogeneity condition is not. Our data are not stratified by known risk factors, sex and age being most well-established. Also, there are complex artifacts in the way death reports are distributed over the days of the week. Pooling data with different event probabilities will give us a scale parameter greater than 1, as is seen here.

Although we don't have access to the stratified data that we would need to characterize these sources of dispersion, we can do some sensitivity analyses to see to what extent age and sex effects might inflate the scale parameter.

The following function implements a form of sensitivity analysis in which we attempt to identify population structures that are consistent with the findings we reached above with the Covid tracking project data. Our goal here is to get a better sense for how much heterogeneity would be needed to attain a scale parameter around 10, while respecting the marginal mean structure estimated above. We can do this, in a very hypothetical way, by imagining that our population

consists of two groups with different risks. If we specify the prevalences of these two groups, and the risk ratio between them, and force the marginal mean to agree with what we found above, we can calculate the scale parameter.

This idea is implemented in the function below. The parameter 'high_risk' is a hypothetical risk ratio between a high risk group and a low risk group. The parameter 'pr_high' is the hypothetical proportion of the population in the high risk group.

[18]:
```
def scale_2group(high_risk, pr_high):
    f = np.r_[1, high_risk]
    pr = np.r_[1 - pr_high, pr_high]
    f /= np.dot(f, pr)
    ex = r1.fittedvalues
    exq = [fx*ex for fx in f]
    mcv = sum([p*b for (p, b) in zip(pr, exq)])
    vcm = sum([p*(b-ex)**2 for (p, b) in zip(pr, exq)])
    tv = mcv + vcm
    return (tv/ex).mean()
```

If we calculate the hypothetical scale parameter for an imaginary population in which 5% of the population is 4 times more likely to die than the others, we get a scale parameter of around 11, which is close to our observed scale parameter. Of course the real world does not consist of two homogeneous groups, but this may not be as far as one would imagine from reality. For example, including a third group with very low risk would not change anything. Consistent with observed data, we can suppose that the risk of dying for people under 40 is negligible.

Below we show two hypothetical 2-group scenarios that would explain the scale parameters estimated in the conventional way, and robustly. In the first case, we posit that 5% of the population has a 3.8 times greater risk of death than the others. In the second case, we posit that 2% of the population has a 3.3 times greater risk of death than the others.

[19]:
```
print(scale_2group(3.8, 0.05))
print(scale_2group(3.3, 0.02))
```

```
10.494514484697184
4.139938674770221
```

As another illustration, if half the population (i.e. males) has a 2 times greater risk of death than the others, then we get a scale parameter that is in-range with our estimated values.

[20]:
```
print(scale_2group(2.0, 0.5))
```

```
4.681546432841761
```