COVID-19 negative binomial deaths model

This notebook fits a model using Novel Coronavirus (COVID-19) cases and deaths data by country, provided by JHU CSSE at [https://github.com/CSSEGISandData/COVID-19 (https://github.com/CSSEGISandData/COVID-19)](https://github.com/CSSEGISandData/COVID-19)

We begin by estimating the new deaths at each future date t based on new cases declared on each previous date:
$$ \operatorname{E} \bigl[{nd}_t \bigr] = \sum_{i=1}^t {nc}_i \left(1-s \right) Pr\bigl[\text{ dies at } t \text{ } \mid \text{ } {nc}_i \bigr] $$

where:

${s}$ = probability of survival for a new case

${nd}_t$ = new deaths on date t

${nc}_t$ = new cases on date t

The negative binomial distribution is useful to model this conditional probability. We assume that the lag between a positive test result (i.e. creating a new case) and death due to COVID-19 follows a negative binomial distribution with parameters n and p. This can be interpreted as the probability there will be t failures until the n-th success for $t+n$ independent and identically distributed trials, each with probability of success p.

We start by combining cases and deaths data from the Johns Hopkins data for a selected country.

In [25]:

```
import pandas as pd
import numpy as np
#country='Spain'
#country='Italy'
country='United Kingdom'
#country='France'
#country='Switzerland'
#country='US'
#country='China'
#country='US'
```

In [26]:

```
#confirmed cases in time_series_covid19_confirmed_global.csv
url_c = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series
_covid19_confirmed_global.csv'
file_c = 'C:/Users/Mark/Documents/Python/code/time_series_covid19_confirmed_global.csv'
read_c = url_c #url_c or local file_c if saved already
df_c = pd.read_csv(read_c)   #global confirmed cases
df_c['Province/State'] = df_c['Province/State'].fillna('ALL')
df_cc = df_c.loc[(df_c['Country/Region']==country) & (df_c['Province/State']=='ALL')]
df_cc = df_cc.T; df_cc.columns=['cases']
df_cc = df_cc.iloc[4:]
df_cc.index = pd.to_datetime(df_cc.index)
```

In [27]:

```
#deaths in time_series_covid19_deaths_global.csv
url_d = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series
_covid19_deaths_global.csv'
file_d = 'C:/Users/Mark/Documents/Python/code/time_series_covid19_deaths_global.csv'
read_d = url_d #url_d or local file_d if saved already
df_d = pd.read_csv(read_d)   #global confirmed cases
df_d['Province/State'] = df_d['Province/State'].fillna('ALL')
df_dc = df_d.loc[(df_d['Country/Region']==country) & (df_d['Province/State']=='ALL')]
df_dc = df_dc.T; df_dc.columns=['deaths']
df_dc = df_dc.iloc[4:]
df_dc.index = pd.to_datetime(df_dc.index)
#print(df_dc)
```

In [28]:

```
df = pd.concat([df_cc,df_dc], axis=1, sort=False)
df['country'] = country
df = df[['country','cases','deaths']]
```

The 5 most recent days published are:

In [30]:

```
print(df.tail(5))
```

```
                   country  cases  deaths
2020-04-09  United Kingdom  65077    7978
2020-04-10  United Kingdom  73758    8958
2020-04-11  United Kingdom  78991    9875
2020-04-12  United Kingdom  84279   10612
2020-04-13  United Kingdom  88621   11329
```

We calculate the rate of new cases

In [31]:

```
df['new_deaths'] = df['deaths']-df['deaths'].shift(1)
df['new_cases'] = df['cases']-df['cases'].shift(1)
df.at[df.index[0],'new_deaths']=df.loc[df.index[0],'deaths']
df.at[df.index[0],'new_cases']=df.loc[df.index[0],'cases']
df['new_cases_rate'] = df['new_cases'] / (0.5*df['cases'].shift(1)+0.5*df['cases']+1e-10)
df.at[df.index[0],'new_cases_rate']=0.0
```

In [32]:

```
df.tail(5)
```

Out[32]:

	country	cases	deaths	new_deaths	new_cases	new_cases_rate
2020-04-09	United Kingdom	65077	7978	881	4344	0.0690565
2020-04-10	United Kingdom	73758	8958	980	8681	0.125055
2020-04-11	United Kingdom	78991	9875	917	5233	0.0685176
2020-04-12	United Kingdom	84279	10612	737	5288	0.0647761
2020-04-13	United Kingdom	88621	11329	717	4342	0.0502256

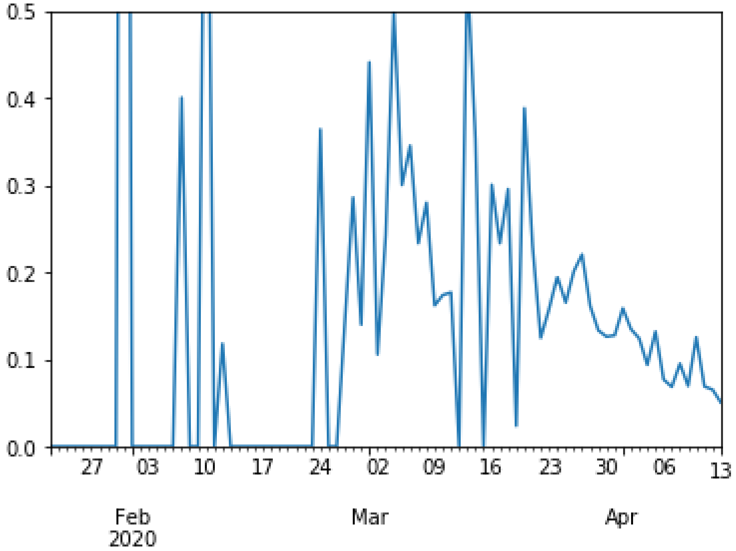
We can see the downward trend we are hoping for as measures like the lockdown take effect.

In [35]:

```
df['new_cases_rate'].plot( ylim=(0,0.5))
```

Out[35]:

<matplotlib.axes._subplots.AxesSubplot at 0x1e9542bc5c8>



This is very noisy data! For now we assume no new cases and project deaths resulting from existing cases only. Even this is a challenge, although the negative binomial model does help.

The cases and deaths data each have their own pros and cons:

- *Cases data* leads deaths data, but it has a high margin of error due to false positives and negatives and limited coverage of the population at risk; while
- *Deaths data* lags cases data, but it has a low margin of error. The sources of 'error' in modeling terms are misclassification of cause of death and deaths ascribed to COVID-19 but not previously tested.

By modeling the combinations of cases and deaths we hope to eliminate some of both of these sources of noise and get a better read on the underlying trend of new infections.

We assume that a new case has a daily probability s of survival given tests positive for COVID-19 and a negative binomial distribution for the time to death if the new case does not survive. The mean time until death is $\frac{n(1-p)}{p}$ for negative binomial parameters n and p. The three parameters s,p amd n are fitted from each country's experience using least squares.

To do this, we import our COVID19 module fit and projection functions to apply to our DataFrame **df** of experience.

In [37]:

```
#import our COVID19 module to access fit and projection functions to apply to our Data
from COVID19 import fit_err, fit_model, projection_df
#import functions for negative binomial model from scipy module
from scipy.stats import nbinom
```

We fit the model at each date and report the results, looking for trends in survival rates and $ E \left [\text{days to death} | \text{new infection} \right] $

In [38]:

```
bounds_tuple = ((0.1,0.99),(0.1,0.9),(1.0,100.0))   #we use these bounds respectively for s, p, n
max_iterations = 50
init_params_tuple = (0.5,0.35,7.0)

start_loc = 30 #pointless fitting 3 parameters to less than 30 data points
end_loc = df.shape[0]

for i in range(start_loc, end_loc):
    fit_df = df.iloc[:i]
    res=fit_model(initparams=init_params_tuple, hyperparams=fit_df, bounds=bounds_tuple, maxiter=max_iterations)
    s,p,n=res[0]
    mean = nbinom.mean(n, p) #n*p/(1-p)
    print(df.index[i].strftime('%Y-%m-%d'),'parameters=',res[0],',',round(mean,1),'days to death,', round(res[1],0),'error')
    init_params_tuple = res[0]
    s,p,n = res[0]
    df.at[df.index[i],'s']=s
    df.at[df.index[i],'p']=p
    df.at[df.index[i],'n']=n
```

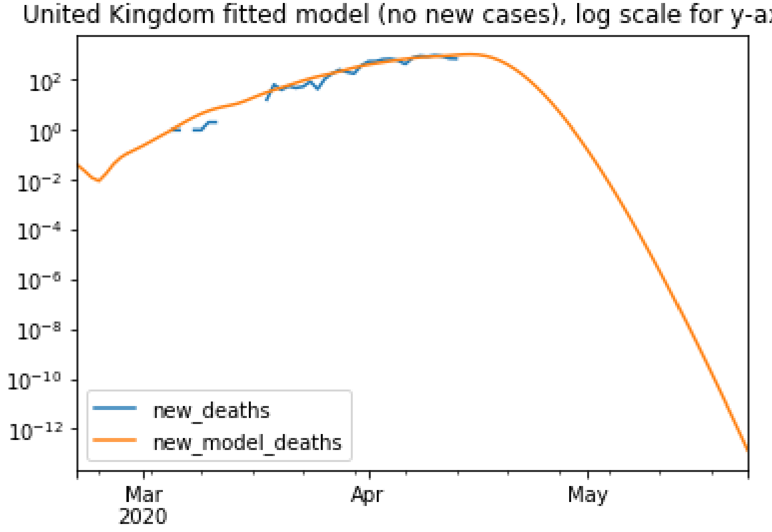
```
2020-02-21 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-22 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-23 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-24 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-25 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-26 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-27 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-28 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-02-29 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-01 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-02 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-03 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-04 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-05 parameters= [0.99       0.1        7.14140596] , 64.3 days to death, 0.0 error
2020-03-06 parameters= [ 0.97327659  0.9        14.51323101] , 1.6 days to death, 0.0 error
2020-03-07 parameters= [ 0.96879283  0.9        15.48392113] , 1.7 days to death, 0.0 error
2020-03-08 parameters= [0.98703776 0.9        3.36515494] , 0.4 days to death, 1.0 error
2020-03-09 parameters= [0.98558099 0.9        3.36513822] , 0.4 days to death, 1.0 error
2020-03-10 parameters= [0.98364877 0.82905188 3.36728576] , 0.7 days to death, 1.0 error
2020-03-11 parameters= [0.90718823 0.49223253 7.99158173] , 8.2 days to death, 2.0 error
2020-03-12 parameters= [ 0.93047915  0.72846166 18.03014091] , 6.7 days to death, 2.0 error
2020-03-13 parameters= [0.97912256 0.9        1.        ] , 0.1 days to death, 2.0 error
2020-03-14 parameters= [ 0.98208685  0.9        25.38246027] , 2.8 days to death, 6.0 error
2020-03-15 parameters= [ 0.9619949   0.9        10.84069642] , 1.2 days to death, 49.0 error
2020-03-16 parameters= [0.9752891  0.9        4.29886221] , 0.5 days to death, 70.0 error
2020-03-17 parameters= [0.95282961 0.9        1.        ] , 0.1 days to death, 521.0 error
2020-03-18 parameters= [0.9669236  0.79017289 1.        ] , 0.3 days to death, 793.0 error
2020-03-19 parameters= [0.97085878 0.9        1.        ] , 0.1 days to death, 786.0 error
2020-03-20 parameters= [ 0.75762901  0.9        62.75737922] , 7.0 days to death, 2181.0 error
2020-03-21 parameters= [ 0.76413427  0.89781125 62.82853403] , 7.2 days to death, 2219.0 error
2020-03-22 parameters= [ 0.78713918  0.9        59.70988838] , 6.6 days to death, 2218.0 error
2020-03-23 parameters= [ 0.83645797  0.9        53.05152907] , 5.9 days to death, 2499.0 error
2020-03-24 parameters= [ 0.84678463  0.8968729  53.3049009 ] , 6.1 days to death, 2819.0 error
2020-03-25 parameters= [ 0.85000586  0.9        53.36765332] , 5.9 days to death, 2826.0 error
2020-03-26 parameters= [ 0.92886493  0.9        22.1075958 ] , 2.5 days to death, 4302.0 error
2020-03-27 parameters= [ 0.92677173  0.9        19.69390997] , 2.2 days to death, 4682.0 error
2020-03-28 parameters= [0.93131845 0.87919776 7.87173567] , 1.1 days to death, 6426.0 error
2020-03-29 parameters= [ 0.91220152  0.9        14.38857369] , 1.6 days to death, 11652.0 error
2020-03-30 parameters= [ 0.91356089  0.9        13.86747951] , 1.5 days to death, 11670.0 error
2020-03-31 parameters= [ 0.9194553   0.9        11.33760783] , 1.3 days to death, 12549.0 error
2020-04-01 parameters= [ 0.28736172  0.4792983  14.30381059] , 15.5 days to death, 24382.0 error
```

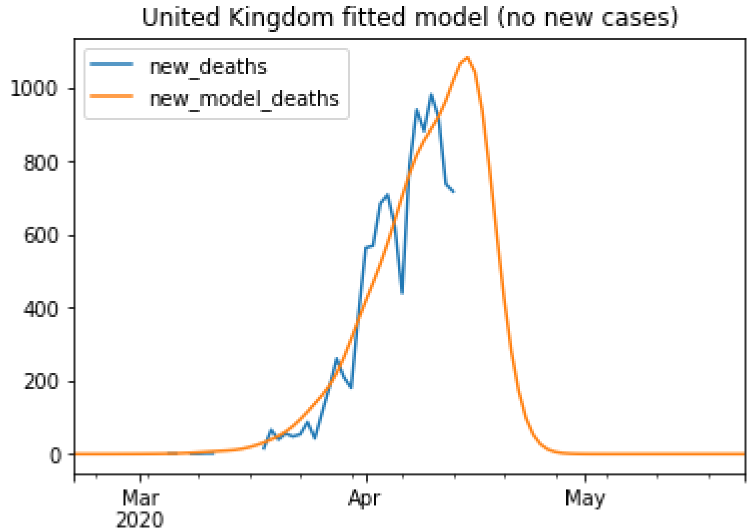
```
2020-04-02 parameters= [ 0.1         0.71677532 39.00940688] , 15.4 days to death, 41846.0 error
2020-04-03 parameters= [ 0.1         0.85503341 88.95675517] , 15.1 days to death, 41817.0 error
2020-04-04 parameters= [  0.1          0.86959606 100.        ] , 15.0 days to death, 42359.0 error
2020-04-05 parameters= [ 0.1         0.86829158 99.99999999] , 15.2 days to death, 46697.0 error
2020-04-06 parameters= [ 0.69164595  0.9        76.04713729] , 8.4 days to death, 82150.0 error
2020-04-07 parameters= [ 0.780176    0.9        58.72215928] , 6.5 days to death, 184764.0 error
2020-04-08 parameters= [ 0.77856969  0.9        59.11904678] , 6.6 days to death, 184832.0 error
2020-04-09 parameters= [ 0.76192283  0.9        63.52711871] , 7.1 days to death, 191565.0 error
2020-04-10 parameters= [ 0.76947086  0.9        61.34879353] , 6.8 days to death, 192981.0 error
2020-04-11 parameters= [ 0.76584864  0.9        62.50365869] , 6.9 days to death, 193405.0 error
2020-04-12 parameters= [ 0.77507856  0.9        59.40696874] , 6.6 days to death, 198448.0 error
2020-04-13 parameters= [ 0.79284255  0.9        53.9280128 ] , 6.0 days to death, 261303.0 error
```

In [42]:

```
#plot the projection for the model (fitted from lockdown), extending the df index 100 days into the future
proj_df = projection_df(params=res[0], df=df, cases_growth_rate=0)

proj_df['new_deaths'] = proj_df['new_deaths'].replace(0.0, np.nan) #don't plot zero values

s,p,n = res[0]
#mean = n*(1/p - 1) = n*(1-p)/p ##C:\ProgramData\Anaconda3\Lib\site-packages\scipy\stats\_discrete_distns.py line 210
mean = nbinom.mean(n, p)
print('mean days to death for those who do not survive =',round(mean,1),'days')

proj_df[['new_deaths','new_model_deaths']].iloc[30:-60].plot(title=country+' fitted model (no new cases), log scale for y-axis', l
ogy=True)
proj_df[['new_deaths','new_model_deaths']].iloc[30:-60].plot(title=country+' fitted model (no new cases)')

loc_max = proj_df.loc[ proj_df['new_model_deaths'] == proj_df['new_model_deaths'].max()].index[0]
print()
print(int(round(proj_df['new_model_deaths'].max(),0)),'max deaths expected on',loc_max.strftime('%Y-%m-%d'))
print()
```

```
the negative binomial model projected for 183 days accounts for 100.0 % of future deaths
mean days to death for those who do not survive = 6.0 days

1081 max deaths expected on 2020-04-15
```





Our deaths projection may give a cleaner indication of the trend in infections.

Here are the rates of new deaths for last 15 days:

In [43]:

```
proj_df['new model deaths rate'] = proj_df['new_model_deaths']/proj_df['deaths']
print(proj_df.tail(115).head(15))
```

	country	cases	deaths	new_deaths	new_cases	\
2020-03-30	United Kingdom	22141.0	1408.0	180.0	2619.0	
2020-03-31	United Kingdom	25150.0	1789.0	381.0	3009.0	
2020-04-01	United Kingdom	29474.0	2352.0	563.0	4324.0	
2020-04-02	United Kingdom	33718.0	2921.0	569.0	4244.0	
2020-04-03	United Kingdom	38168.0	3605.0	684.0	4450.0	
2020-04-04	United Kingdom	41903.0	4313.0	708.0	3735.0	
2020-04-05	United Kingdom	47806.0	4934.0	621.0	5903.0	
2020-04-06	United Kingdom	51608.0	5373.0	439.0	3802.0	
2020-04-07	United Kingdom	55242.0	6159.0	786.0	3634.0	
2020-04-08	United Kingdom	60733.0	7097.0	938.0	5491.0	
2020-04-09	United Kingdom	65077.0	7978.0	881.0	4344.0	
2020-04-10	United Kingdom	73758.0	8958.0	980.0	8681.0	
2020-04-11	United Kingdom	78991.0	9875.0	917.0	5233.0	
2020-04-12	United Kingdom	84279.0	10612.0	737.0	5288.0	
2020-04-13	United Kingdom	88621.0	11329.0	717.0	4342.0	

	new_cases_rate	s	p	n	new_model_deaths	\
2020-03-30	0.125723	0.913561	0.900000	13.867480	313.824169	
2020-03-31	0.127255	0.919455	0.900000	11.337608	366.755089	
2020-04-01	0.158319	0.287362	0.479298	14.303811	417.744863	
2020-04-02	0.134321	0.100000	0.716775	39.009407	467.027076	
2020-04-03	0.123807	0.100000	0.855033	88.956755	518.702494	
2020-04-04	0.093292	0.100000	0.869596	100.000000	576.047194	
2020-04-05	0.131603	0.100000	0.868292	100.000000	638.685512	
2020-04-06	0.076488	0.691646	0.900000	76.047137	703.319155	
2020-04-07	0.068021	0.780176	0.900000	58.722159	764.532398	
2020-04-08	0.094693	0.778570	0.900000	59.119047	816.442352	
2020-04-09	0.069057	0.761923	0.900000	63.527119	855.811249	
2020-04-10	0.125055	0.769471	0.900000	61.348794	886.405150	
2020-04-11	0.068518	0.765849	0.900000	62.503659	918.917100	
2020-04-12	0.064776	0.775079	0.900000	59.406969	962.811134	
2020-04-13	0.050226	0.792843	0.900000	53.928013	1016.872747	

	new model deaths rate
2020-03-30	0.222886
2020-03-31	0.205006
2020-04-01	0.177613
2020-04-02	0.159886
2020-04-03	0.143884
2020-04-04	0.133561

```
2020-04-05    0.129446
2020-04-06    0.130899
2020-04-07    0.124133
2020-04-08    0.115040
2020-04-09    0.107271
2020-04-10    0.098951
2020-04-11    0.093055
2020-04-12    0.090729
2020-04-13    0.089758
```

One idea would be to project future deaths straight from this smoothed trend and compare it against a projection of deaths resulting from assumed new cases.