Analysis of VoC data

Look at data provided to SPI-M on the B.1.351 VoC.

Data file not included in the repo, so change path if running independently.

This version:

- Data is that provided on 21 April.
- Go back to piecewise exp methodology.
- Clearer labels of calendar dates

In [1]:

```
%matplotlib inline
import numpy as np
import scipy.stats as st
import scipy.special as sp
import matplotlib.pyplot as plt
import pandas as pd
import datetime
import scipy.optimize as op
```

In [2]:

```
df = pd.read_excel(
    '/Volumes/COVID19_Epi_modelling/DstlDailyData/2021-04-21/VOC202012_02_linelist_2
    sheet_name=1,
)
df.dropna(subset=['earliest_specimen_date'],inplace=True)
```

In [3]:

```
def todays(x):
    return np.array((pd.to_datetime(x['earliest_specimen_date'], format='%Y-%m-%d')
```

In [4]:

```
monthstarts = pd.DataFrame([
    '2020-10-01','2020-11-01','2020-12-01','2021-01-01','2021-02-01','2021-03-01','2
],columns=['earliest_specimen_date'])
monthstarts
```

Out[4]:

	earliest_specimen_date
0	2020-10-01
1	2020-11-01
2	2020-12-01
3	2021-01-01
4	2021-02-01
5	2021-03-01
6	2021-04-01

In [5]:

```
def tods(x):
    return pd.to_datetime(x['earliest_specimen_date'], format='%Y-%m-%d').dt.strftim
```

In [6]:

```
keydates = pd.DataFrame([
    ['2021-01-01','Start of 2021'],
    ['2021-03-08','Start of Roadmap'],
    ['2021-03-24','Sequence Delay'],
],columns=['earliest_specimen_date','date_name'])
keydates
```

Out[6]:

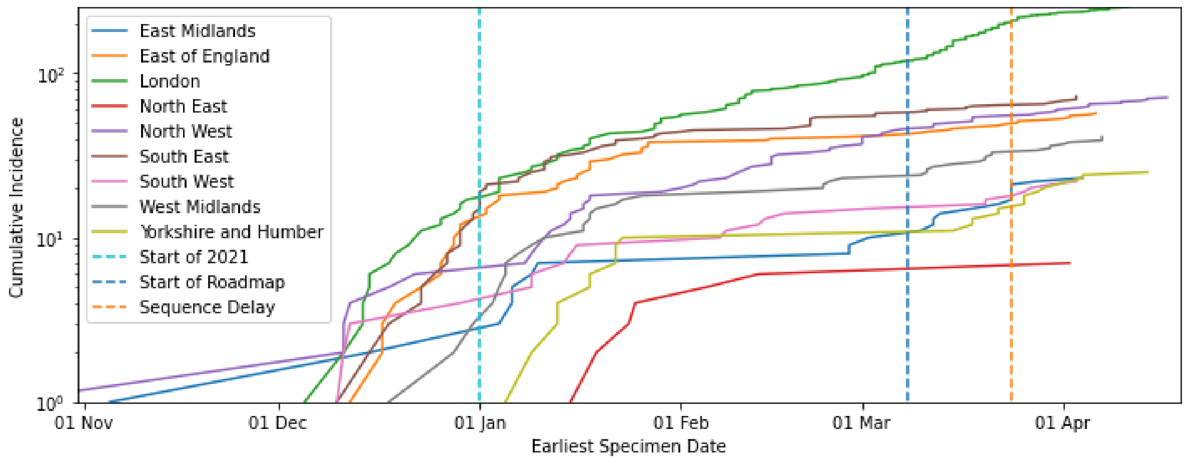
	earliest_specimen_date	date_name
0	2021-01-01	Start of 2021
1	2021-03-08	Start of Roadmap
2	2021-03-24	Sequence Delay

In [7]:

```
dfr = df.groupby('PHEC_name').apply(todays)
```

In [8]:

```
plt.figure(figsize=(10,4))
for i in range(0,len(dfr.index)):
    x = dfr[i]
    plt.plot(np.sort(x),np.arange(1,len(x)+1),label=dfr.index[i])
for i, d in keydates.iterrows():
    plt.plot(todays(d)*np.ones(2),np.array([0,250]),ls='--',label=d.date_name)
plt.legend()
plt.xlabel('Earliest Specimen Date')
plt.ylabel('Cumulative Incidence')
plt.xticks(todays(monthstarts),tods(monthstarts))
plt.xlim([30,200])
plt.ylim([1,250])
plt.yscale('log')
plt.tight_layout()
#plt.savefig('./voc_region_log.pdf')
```



In [9]:

```
# From the above, only London seems to exhibit exponential growth so look at that
li = np.argwhere(dfr.index == 'London')[0]
x = dfr[li].values[0]
```

In [10]:

```
def mymu(x,tt,n):
    mu = np.concatenate([
        x[0]*np.exp(x[1]*tt[0:n]),
        x[0]*np.exp(x[1]*tt[n-1])*np.exp(x[3]*(tt[n:]-tt[n-1]))
    ])
    return mu

def myod(x,tt,n):
    od = np.concatenate([
        x[2]*np.ones(len(tt[0:n])),
        x[4]*np.ones(len(tt[n:]))
    ])
    return od

def mynll(y,x,tt):
    mu = x[0]*np.exp(x[1]*tt)
    od = x[2]*np.ones(len(tt))
    p = 1/od
    r = mu/(od-1)
    return -np.sum(st.nbinom.logpmf(y,r,p))

def mynll2(y,x,tt,n):
    mu = mymu(x,tt,n)
    od = myod(x,tt,n)
    p = 1/od
    r = mu/(od-1)
    return -np.sum(st.nbinom.logpmf(y,r,p))
```

In [11]:

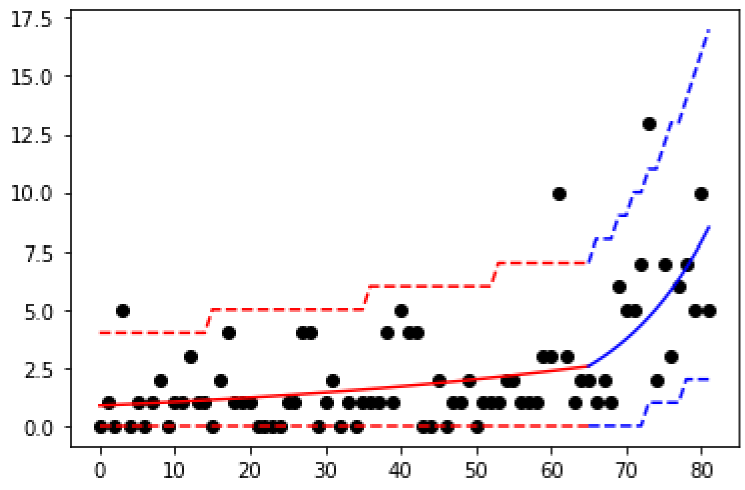
```
tk = todays(keydates)
z = np.bincount(x)
yy = z[tk[0]:tk[-1]]
tt = np.arange(0,len(yy))
n = tk[1]-tk[0]
```

In [12]:

```
x02 = np.array([1,0.25,5,0.15,5])
nll = lambda xx: mynll2(yy,xx,tt,n)
fout2 = op.minimize(nll,x02,method='Nelder-Mead')
mu = mymu(fout2.x,tt,n)
od = myod(fout2.x,tt,n)
p = 1/od
r = mu/(od-1)
dt1 = np.log(2.0)/fout2.x[1]
dt2 = np.log(2.0)/fout2.x[3]
plt.plot(tt[0:n],mu[0:n],linestyle='-',c='r',label='Doubling time {:.1f} days'.forma
plt.plot(tt[(n-1):],mu[(n-1):],linestyle='-',c='b',label='Doubling time {:.1f} days'
plt.plot(tt[0:n],st.nbinom.ppf(0.025,r[0:n],p[0:n]),linestyle='--',c='r')
plt.plot(tt[0:n],st.nbinom.ppf(0.975,r[0:n],p[0:n]),linestyle='--',c='r')
plt.plot(tt[(n-1):],st.nbinom.ppf(0.025,r[(n-1):],p[(n-1):]),linestyle='--',c='b')
plt.plot(tt[(n-1):],st.nbinom.ppf(0.975,r[(n-1):],p[(n-1):]),linestyle='--',c='b')
plt.scatter(tt,yy,marker='o',c='k',label='Data')
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x7fbdd8304250>
```



In [14]:

```
dt2
```

Out[14]:

```
9.265678111005375
```