ds2 hw2

Nick Strayer

September 21, 2014

Code for each problem is given in `problem_number.py`

1. This was relatively straightforward. A couple of the problems that I ran into along the way:

   ### What tool to use:

   - I originally tried using **requests** but the format didnt work with Tycho (at least in the time I spent with it.) It seemed like Tycho was particular about the order of the parameters in the query and requests couldnt care less.
   - I eventually went with good-ol-fashioned **urllib.** I did this because it was the method that was used in the example on the Tycho website. I lost the ability to fail gracefully but Tycho actually nicely fails in the output anyways so it was fine.

   ### How to feed in the parameters:

   - It wasnt super clear if Tycho wanted individual states for the state query, and if it did, if it wanted them given in loc or state (seems more obvious now looking back on it.) Through trial and error this was figured out.

   ### How to store the data:

   - Did I want to aggregate the data into individual csvs for each state or disease or what? Eventually, I decided to simply get everything in as fine a resolution as possible because space is cheap and later processing would be easier if I choose to switch my methods.

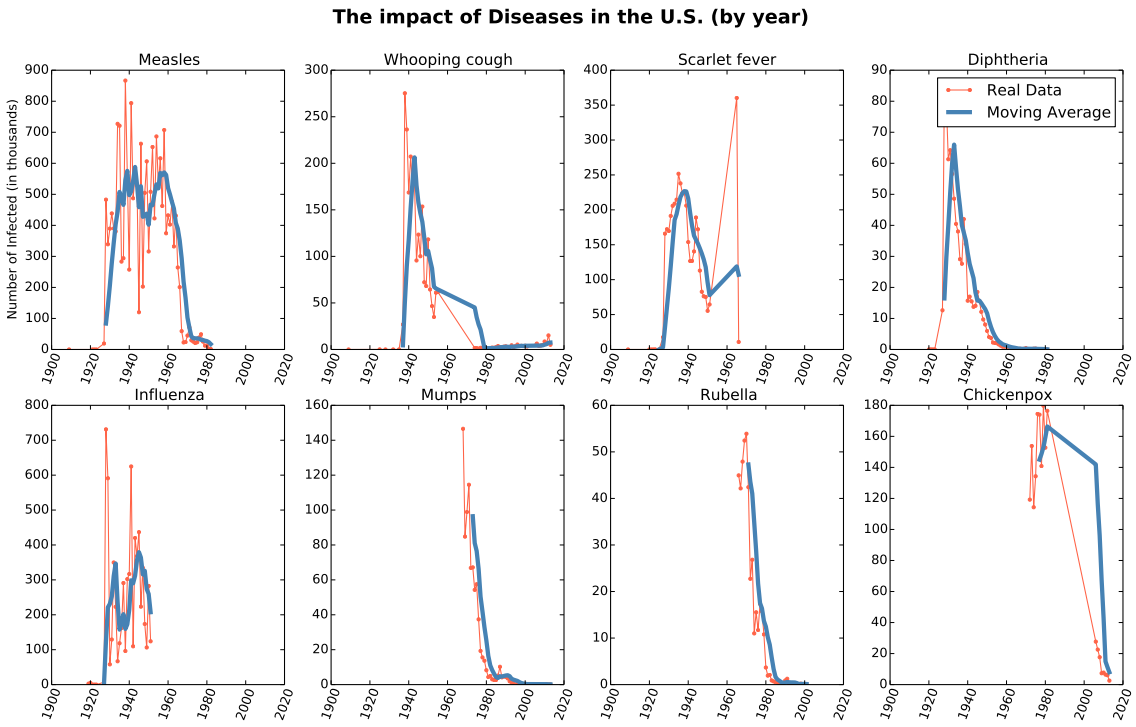
   ### Messed up disease names:

   - The first time that I ran the script I incorrectly queried for Chickenpox and Whooping cough. I didnt realize that you needed to name them explicitly (i.e. with their formal name in square brackets).
   - I changed their names and re-ran the script only looping over the messed up diseases. **Lines 43 to 48 in problem1.py**.

2. The age old question of how to represent data! I am choosing to go summed by the whole country as state boundaries, in my opinion, especially for something like this are rather arbitrary. No-one* looks back and remembers how badly a disease hit Ohio in the 1920s, (an exception would be for something like AIDS in San Fransisco in the 1980s, but we arent looking at that data.)

   I read in all of the files grabbed in problem 1, then concatenated them by disease, grouped by year, then summed to get the whole country average. **20 to 27 in problem2.py**

Output:



Note: This also made me aware of the problems with the death data. I am not sure why, but there seems to be a big gulf between the amount of info project Tycho has on cases vs. deaths. For this reason only cases were plotted. The few queries that did return death data contained too sparse a data to provide any meaningful plot or takeaway.

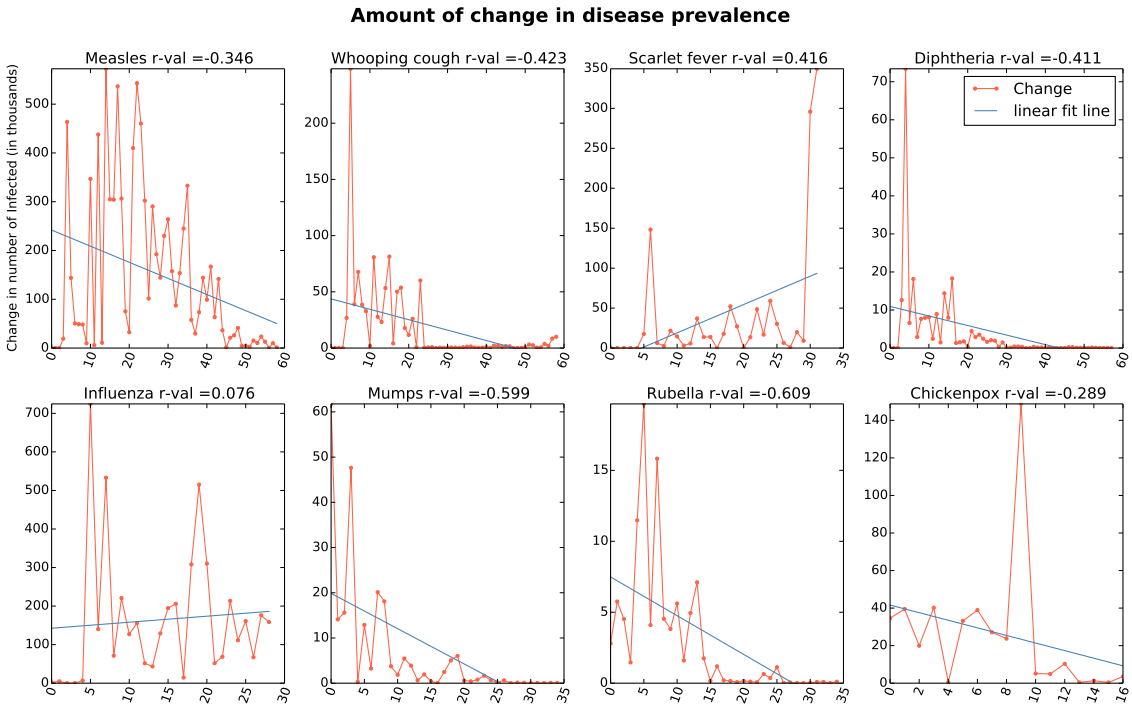
3. A quick google search shows that this question could literally fill multiple theses, so, I will heed the advice given and make something as simple as possible.

   I chose to find the difference in values between every successive measurement. I.e. `|value_i - value_i+1|`. My thought process is that this will help us see the fluctuations. The part of the resulting output (when plotted) that will indicate ability to be predicted, will be the flatness of the line. This will help the comparisons of diseases with different orders of magnitude, since the flatness is relative.

```
def predictabilityIndex(diseaseCounts):
    differenceList = []
    for i in range(len(diseaseCounts) - 1):
        change = abs(diseaseCounts[i] - diseaseCounts[i+1])
        differenceList.append(change)

    return differenceList
```

   After I did this I chose to do linear regression on the output. Unlike normal linear regression though, the goal, at least for predictability, would be to have the slope of the line be zero, so in our case, we want the smallest R-Values as possible. These are the results:



Applying to polio

I went back and to the homework 1 work and modified the plotting portion of my iPython notebook to do this same thing with the polio data. (I have ditto-ed the whole hw01 directory into this homework directory and modified it there):

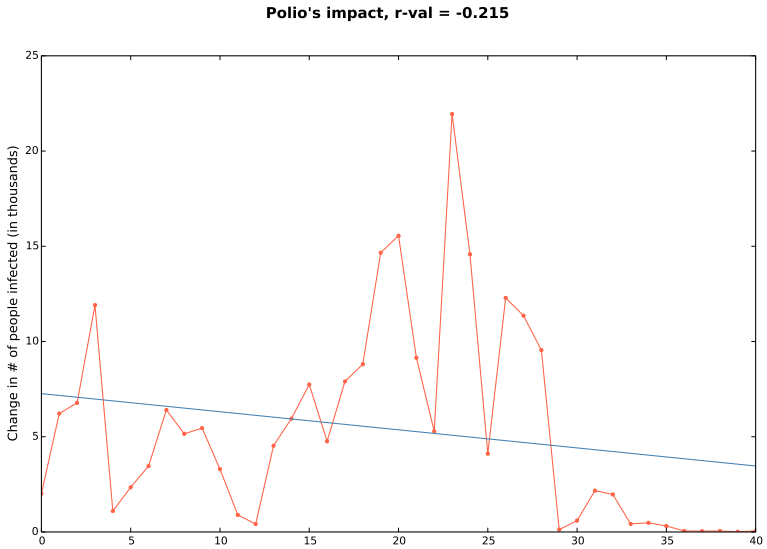
```
import scipy.stats as sp
import matplotlib.ticker as ticker
steelBlue = "#4682b4"
niceRed   = "#ff6347"

def predictabilityIndex(diseaseCounts):
    ...

def findLinearFitLine(changeAmount_list):
   ...

changeInPolio = predictabilityIndex(list(polioByYear["U.S"]))
fig = plt.figure(figsize=(12,8))
plt.plot(changeInPolio, niceRed,marker='.')
ax = fig.add_subplot(1,1,1)
rVal, pVal, linearFitLine = findLinearFitLine(changeInPolio)
plt.plot(linearFitLine, steelBlue)
```

Here is the output:



Interpretation:

From this comparison we can see that polio sits relatively in the bottom of the pack in terms of its r-val. But nothing is substantially different, and influenza for example is way more predictable.

Bonus question:

You could estimate incidence rate if you took into account the average duration of the disease and the death rate of the disease. However, given the scope of this homework this is not realistic.

For one, the death rates would have to be more robust. In their current form there simply isn't enough data to reliably estimate the incidence rates. Also, we don't have information on average disease duration. These data however, could be gotten relatively easy, but I have three more homework assignments that I need to do, so it is effectively impossible.