# Lab #5 – Interpolating and extrapolation vaccination data

***Goals of the lab:***

In this lab, we'll

- Make more use of our code that parses CSV data
- Learn to use Numpy for interpolation and extrapolation, and evaluate when extrapolation makes sense
- Reinforce our skills in drawing plots.

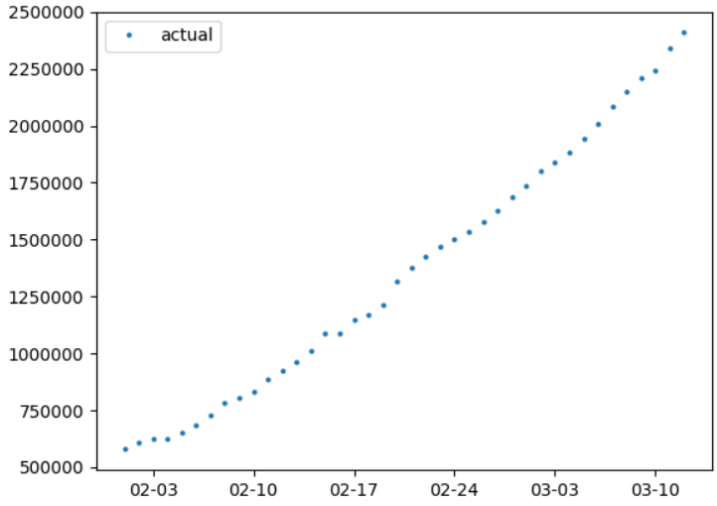
***Overview of the lab***

Our data for this lab comes from the U.S. Centers for Disease Control (the CDC). Every day they release a new data file with state-by-state vaccination data. We'll read all of the per-day data files from Feb. 1st through March 12 and analyze the data.

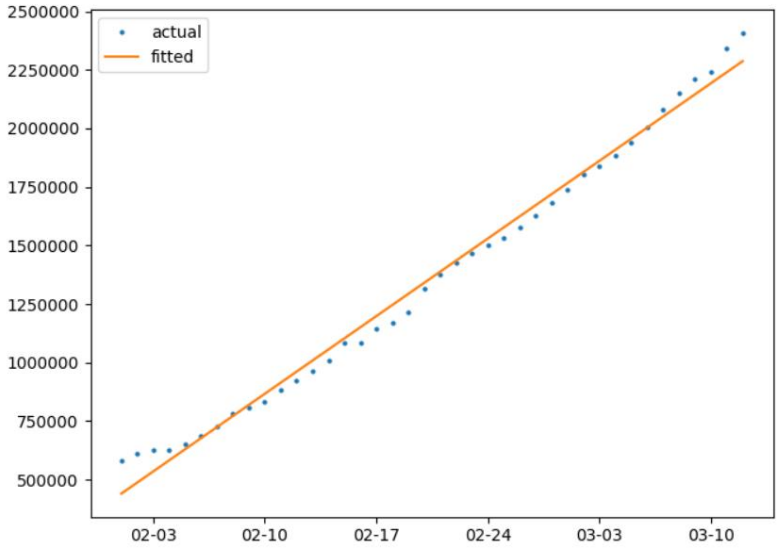
Specifically, we'll take the day-by-day Massachusetts data, curve-fit a function to it, and extrapolate that function out into the future. This will enable our best guess as to when Massachusetts will vaccine the general public – i.e., most of us.

***Extrapolation – what it is and how to do it***

Consider the graph below of the cumulative numbers of vaccinations in Massachusetts versus time (with one dot per day):

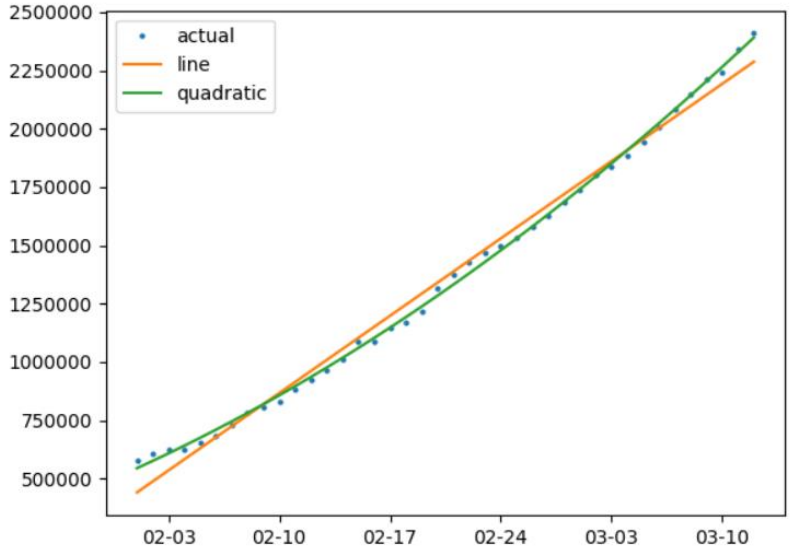



If we were vaccinating the same number of people every day, day in and day out, the points would fall on a straight line. It's not, of course – stuff happens, and some days go better than others. Still, we can come up with a *best-fit* line that fits the points:

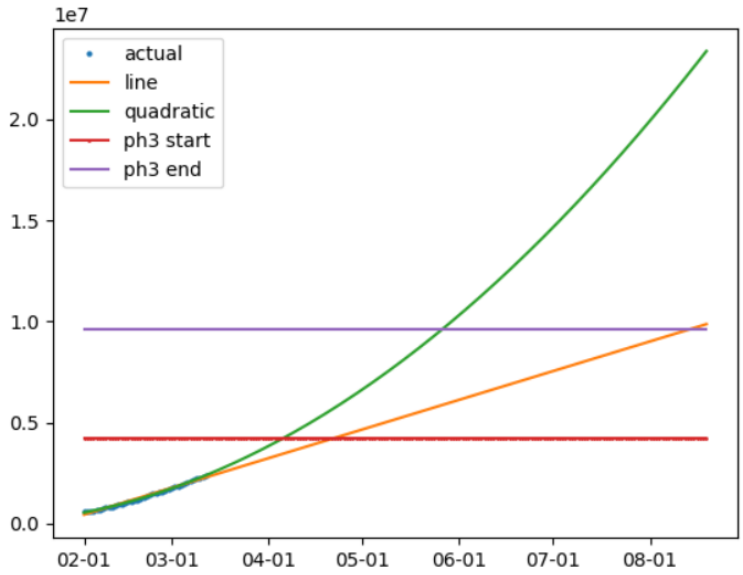



How might we come up with the best-fit line? There are many options. We could, perhaps, change the *cumulative* vaccination numbers into per-day vaccinations. Then we could just average those to get the "equivalent" single per-day number and re-plot it as a cumulative straight line.

A more general process is called *curve fitting*. We can fit a line to a set of points (like we did above), or fit a quadratic curve to the same points if we want. Typically curve-fitting programs define the "best" fit as, roughly, the one that's as close as possible to as many of our points as possible. The mathematical version of this statement is called a *least-squares* fit, and is what most curve-fitting programs give us. Here is the same vaccination data, now also showing a quadratic fit.




Curve fitting is very fun – but do we have a practical use for it? We do indeed. Once we have the best-fit line, we'll *assume* that future vaccinations follow the same trend and *extrapolate* the data forwards. This just means that we take our best-fit line or curve and extend it forwards.




What do we see on this graph? The original blue dots from Feb 1st through March 12th are now too small to easily see. But the orange best-fit line and the green best-fit quadratic are easy to see. Furthermore, we've extended them out several months. And finally, we've added two more lines – the red line at 4.2M vaccinations and the purple line at 9.6M vaccinations. Why those particular numbers? Massachusetts has 2.1 million residents slated to be vaccinated in phase 1 or 2, and another 2.7 million in phase 3. Multiply by two shots/person and you get those two lines. Thus, regardless of what Charlie Baker says, we can read the projected start and end of the state's vaccination Phase 3 from our graph 😊 .

Of course, as the mutual-fund literature says, past performance does not guarantee future results! But the hope is that past performance is nonetheless better than no information at all.

***Details of the lab***

Here is what your program should do (first, in normal English):

- Call the function *read_files* () to read all of the data files and return the Massachusetts vaccination data in a single Numpy array. *Read_files*(), as we will see soon, repeatedly calls a function *read_vacc_file*() to process a single data file and return the single number we care about from that file – the cumulative number of vaccinations through that day.
- Call a function *analyze_and_plot*() to do the relevant curve fitting and extrapolation and plot the results (specifically, to plot the past and future vaccination numbers).

***Top-level code***

Here's the important part of the top-level code:

```
vacc_adm_past = read_files ()
analyze_and_plot (vacc_adm_past)
```

***Metacode for read_files*()**

The metacode for *read_files*() is pretty similar to what you had in lab #3. As usual, you have to flesh it out into Python. It changes slightly since we're dealing with Numpy arrays rather than lists:

```
def read_files():
      initialize vacc_adm_past[] to a Numpy array of 40 zeros
```

```
initialize month and date to Feb 1st
while (month and date haven't gone past March 12th):
      # read the data file for that day
      create the correct filename for that date
      today_cum_doses = read_vacc_file(filename)
      vacc_adm_past [the correct location] = today_cum_doses
      increment date (correctly rolling to March if needed)
return the Numpy array vacc_adm_past[]
```

Note that we've left you the small problem of how to store *today_cum_doses* at the correct location in the Numpy array.

***Metacode for read_vacc_file()***

*Read_vacc_file*() reads the U.S. vaccination data for one day. This is a CSV-format file, with one line for each state. Your code should go through the lines one by one until it finds the Massachusetts line (i.e., the one that starts with "Massachusetts"!). It should then parse that line using *str.split*(), grab the field for the total number of vaccinations so far and return it. Note that it returns a number, not a list. That's because it reads just one file, which contains the cumulative stats for just one day.

```
def read_vacc_file (fname):
      for each line in the file:
            if line starts with "Massachusetts":
                  fields = use split (",") to get the fields
                  return (the appropriate field)
```

The precise format of the lines in the file (which will help you figure out which field to grab) is explained below.

***Metacode for analyze_and_plot()***

The code so far has been quite similar to your previous labs. Now that will change – *analyze_and_plot*() is the meat of the assignment!

```
def analyze_and_plot (vacc_adm_past):
      from numpy.polynomial import Polynomial

      first plot the actual data, in vacc_adm_past. Give it the label
      "actual." Make it just dots with no line connecting them.

      # next, fit a straight line to the data. "deg=1" means to fit the
      # data with a straight line.
      n_days = len (vacc_adm_past)
      fitted = Polynomial.fit (range(n_days), vacc_adm_past, deg=1)
      # extrapolate the best-fit line out for 200 days.
      extrap = fitted (range(200)) # This will be a Numpy array
      plot the best-fit line on the same figure as the original data,
      with the label "line"

      # Next, fit a quadratic to the same data.
      similar to the above use of Polynomial.fit(), but now for a 2nd-
      degree curve.
      plot the best-fit quadratic on the same figure too! (labelled
      'quadratic')

      finally, plot two flat lines – one at y=4200000 labeled "ph3
      start" and one at y=9600000 labeled "ph3 end."
```

***Format of each data file (and where to find the files)***

As before, you can grab the files on the web. This time, they're at http://www.ece.tufts.edu/es/2/data/vaccine/ . Again, you can download the files one by one (they range from *vaccine_2_1.csv* through *vaccine_3_12.csv*) or use the compressed folder *vaccine.zip*.

Each of the files represents the total US vaccination effort up to and including that day. Each file has one line of data for each state. The lines look like the following:

```
Connecticut,847200,654766,23762,18365,462710,12978,183146,5137
```

This is the same CSV format as we've already seen, but with the fields representing different values than in the NY Times lab. The easiest way to parse it is probably to take your existing code that reads a CSV file and alter it slightly as needed.

What do the fields of this particular file mean? In order, they are:

- State/Territory/Federal Entity
- Total doses delivered
- Total doses administered
- Total doses delivered per 100K population
- Total doses administered per 100K population
- Number of people with 1+ Doses
- Number of people with 1+ Doses per 100K
- Number of people with 2 Doses
- Number of people with 2 Doses Per 100K

As mentioned, you can modify your previous code to read this file. The important point is to use only the Massachusetts data. Furthermore, you will only need the "total doses administered" field.

***Questions***

1. You did a curve fit for both a straight line and a quadratic. Which fits the original data better? Do you think this is just coincidence, or a general principle?
2. The quadratic fit predicts that phase 3 both starts and ends earlier than the straight-line fit does. Any intuition why that is? (Hint – note that the vaccination rate started slowly in early February and picked up in early March).
3. Let's say we did our straight-line curve fit, not against *all* of the data from Feb 1 through March 12, but only on the data from Feb 20 through March 12. How do you expect that the prediction would have changed? Why? Feel free to program this to verify your intuition.

***What to turn in:***

- Your program, *mass_vaccine_stats.py*
- A .pdf with your one plot and the answers to the questions.

## *Challenge problems:*

- As usual, feel free to do a plot like the ones I showed, with dates on the *x* axis.
- Add one more plot to the figure – a cubic. Does it exhibit any very odd behavior?

## *Grading*:

- Code & plot correctness: 45 pts
- Code clarity: 10 pts
- Questions: 45 pts total (15 points each)