Section 2: Salting and Smoothing with JFreeCharts and Apache

This program was certainly harder than I anticipated to write. I assumed that there was an Apache library that I could just use to call a data salter and a data smoother function. Unfortunately, no such library exists. One of the first hurdles was that my IDE wouldn't quite work correctly with the JFreeCharts library or the Apache commons math library. It appears that IntelliJ has some sort of bug with one of those libraries which causes you to have to sometimes rebuild the project. That was the first hurdle.

Another problem that I ran into was finding which part of the libraries I should use, especially Apache, the documentation didn't mention anything about a graph smoother or graph salter. One great thing I found was that JFreeCharts has this object called TimeSeriesCollection. This worked well for me because one of my data points is a Date. This feature allowed the chart to format itself properly with the dates instead of treating every x axis label as if it were it's own string datatype.

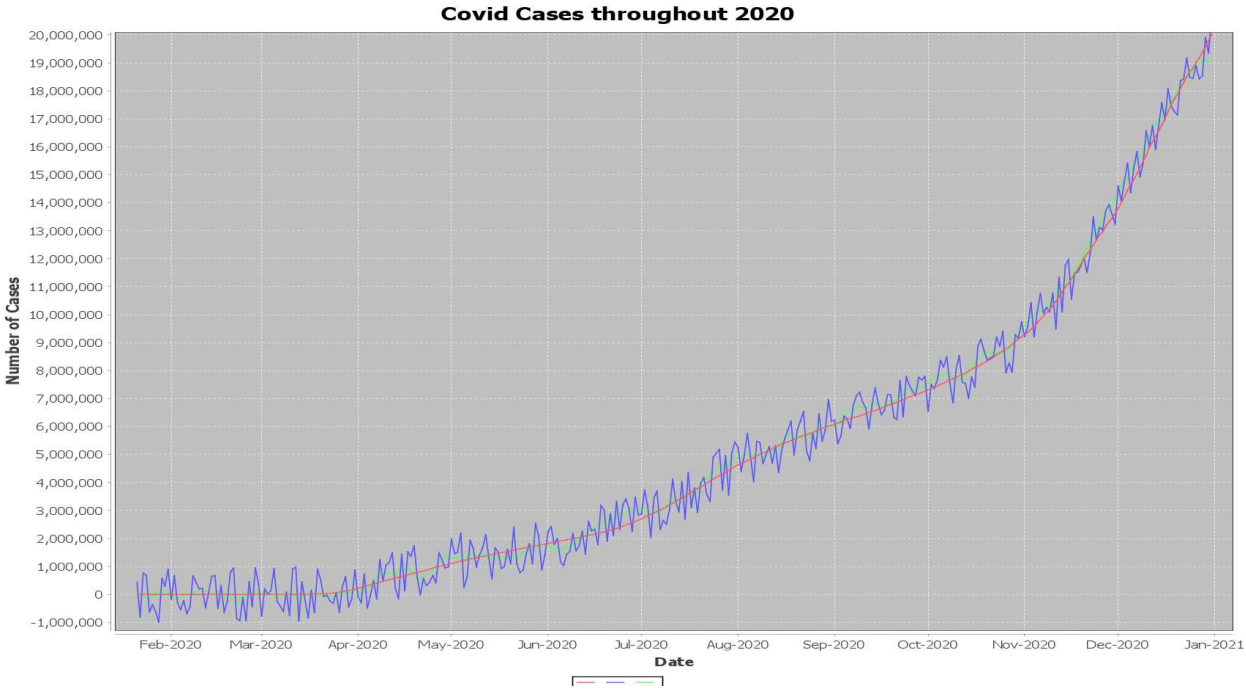
The salted data I got had to write by myself. I could not find any useful Apache math function that would salt the graph data for me. My Salter is very simple. It takes a piece of data and adds a random number between =1 million and 1 million to it. I have this parameter set so high because my data set gets up to almost 2 billion. One wouldn't notice the salting at all if the parameter wasn't so high.

For the Graph Smoother, I wrote a lot of the logic that manages the ArrayList objects, and I extract the values from them and feed them into an Apache mean function. Oddly enough, this function only took an array of doubles to average, and I had to make another array of all 1's because it needed a weight parameter. In the end, it probably would have been less messy, and performed better if I had written the algorithm.

One class I did implement in sort of a weird way was a class I called manageData. This class has a constructor that takes a filename and reads the first two columns from the CSV which was supplied into 2 ArrayLists of different types. Once the constructor has run the data is stored in that class, but it still needs to be parsed into a datatype that JFreeCharts can understand. There's a method in that class called addToDataset that takes the arrayLists global variables from when that method is constructed, and it creates JfreeChart data object and adds them to the dataset.

The Salt and Smoother function takes ArrayList of type integer as parameters, and they have a getter so we can access that array list. This is just a pointer though, when the data is being Salted and Smoothed it is passing the pointer of the original ArrayList object and they are editing the original arrayList. The getter method and ArrayList return type are not always necessary in this code, only when one is going from one object to another and the other object can't resolve the symbol used in the first object.

The graph that I produced is:



You'll notice that there are three lines, a red line, a green Line and a blue line. The red line represents the data without anything done to it. The blue line is the data after it has been salted by adding a random number between -1000000 and 1000000 to the y data point. Once it has been salted, you'll see the green line. The green line is the salted data after it's been run through a smoother. I suspect I could get the graph much closer to the original if I used more data points, but 10 is enough for demonstration purposes. It works well enough.

The Y axis labels on this chart were nice and easy to implement. Before I was using the Date object from JFreeCharts to create the data set, the Y axis was unreadable because it treated every single Y axis label as a String that had to be present on the graph. This date format automatically decides what should and what should not go on the axis label. The x axis label also automatically scaled, but it is just Integer values, so that's expected behavior.

JFreeCharts is something I've never used before, and I would say it was easy to learn and once you pick it up it is very intuitive, which is not something you commonly find with the java programming language. I especially liked how easy it is to go back and plot other lines on the same chart by just adding another series of data.