# Octave CSV : Report

By: Ryan Minneo

\*\*\*As with the report on the Java CSV files, all of the data will be on one google sheet. However the graphs will be taken as png snips from Octave itself. Other than a combined graph on Google Sheets.

[Octave: Plotter, Salter, Smoother](https://docs.google.com/spreadsheets/d/1EOKjxomycR6GkwuGbq47TGhKA1qD0MQ-E_l1Vi2hKUo/edit?usp=sharing)

## Plotting: Journal

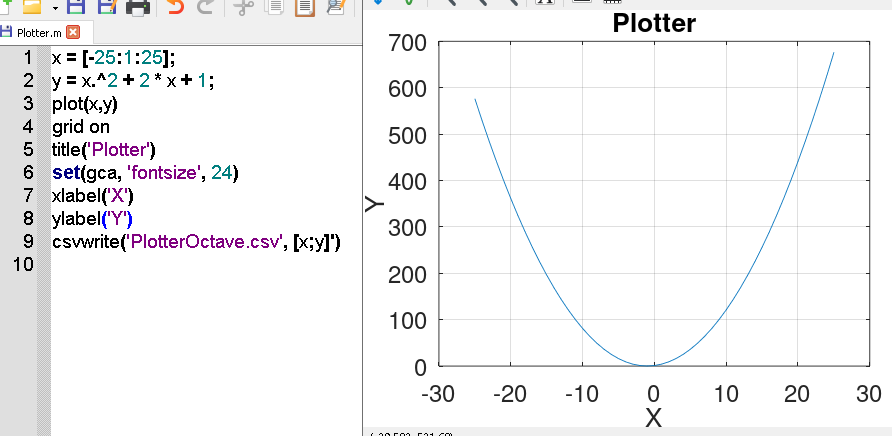
Learning the basics of Octave took such a short amount of time, and the way to create a Plotter was incredibly easy. Where the Java Plotter had just over 100 lines of code, a program with the same functionality can be made in 3 lines of code in Octave. I followed two tutorials to learn the basics. This is the [first video](https://www.youtube.com/watch?v=aD8k4pYUBOk&ab_channel=Mr.STEMEDUTV), and this is the [second video](https://www.youtube.com/watch?v=EbDJNjYZ-EA&ab_channel=MathandScience).

I followed the first video in order to see how to make the graph appear more visually appealing, but I saw in the second video that you can make an array-like dataset by stating “x = [-25:1:25]”.

After I created the graph itself, I looked up how to write/read csv files in octave. This [website](https://www.sharetechnote.com/html/Octave_Matlab_csv.html) really helped me.

## Plotting: Results

The data is in the Google Sheet linked above. Here is the script and graph:



The last line just creates a csv, identical to the Java Plotter.

## Salting: Journal

This took me way longer than I feel it should’ve. I realized and got excited by how easy it was to create the plotter, and I thought there might just be a ‘salt’ function built into Octave. Since there actually was no ‘salt’ function I figured I would just recreate what I did in the java program and make a for loop that goes through the elements and then add a random value between -50 and 50.

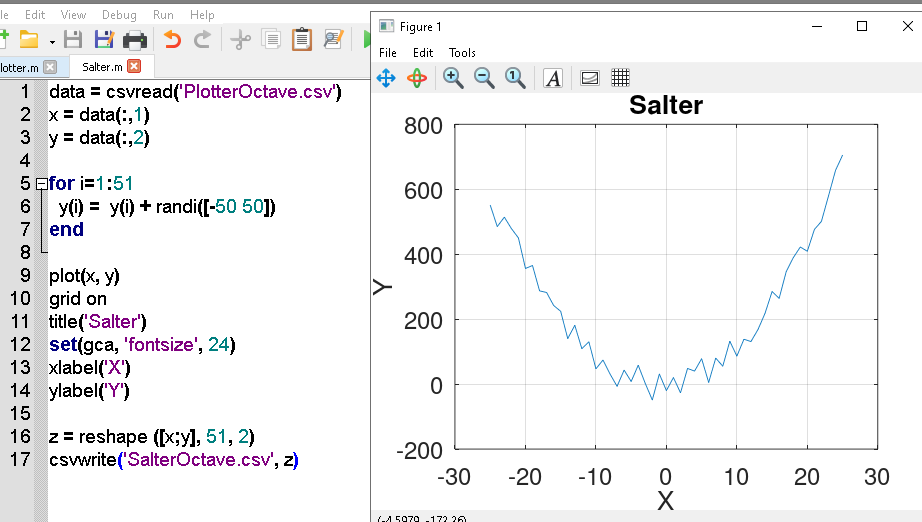
For loops weren’t much of an issue, but finding a way to create a single random integer took a little more time. I found a [website](https://docs.octave.org/latest/Distributions.html) that showed me a bunch of the different distributions when I searched up how to create a random number in octave. Eventually I found the randi function on this [website](https://docs.octave.org/v6.2.0/Random-Number-Generation.html).

After that I copied and pasted the plot~ylabel functions I did on the Plotter to make the graph look nicer. And then I finished it off by putting a “*csvwrite(‘SalterOctave.csv’, [x;y]’)*” which is just an altered version of the last line in the plotter. I thought that it should work perfectly, but of course, it had some problems.

I ran into a ‘vertical dimensions mismatch’ error because for whatever reason my x variable was a 51x1 2D array and my y variable was a 1x51 2D array. I attempted to redefine how I altered the y variable, as I assumed that the line “y(i) = y(i) + randi([-50 50])” was changing the y variable from a 51x1 to a 1x51. Eventually I decided to recreate both x and y into 1 variable, z, in order to write to a csv. I found this [website](https://docs.octave.org/v6.3.0/Index-Expressions.html) which displayed how to reshape a variable into the desired dimensions.

## Salting: Results

The data is in the google sheet linked above. Here is the script and graph.



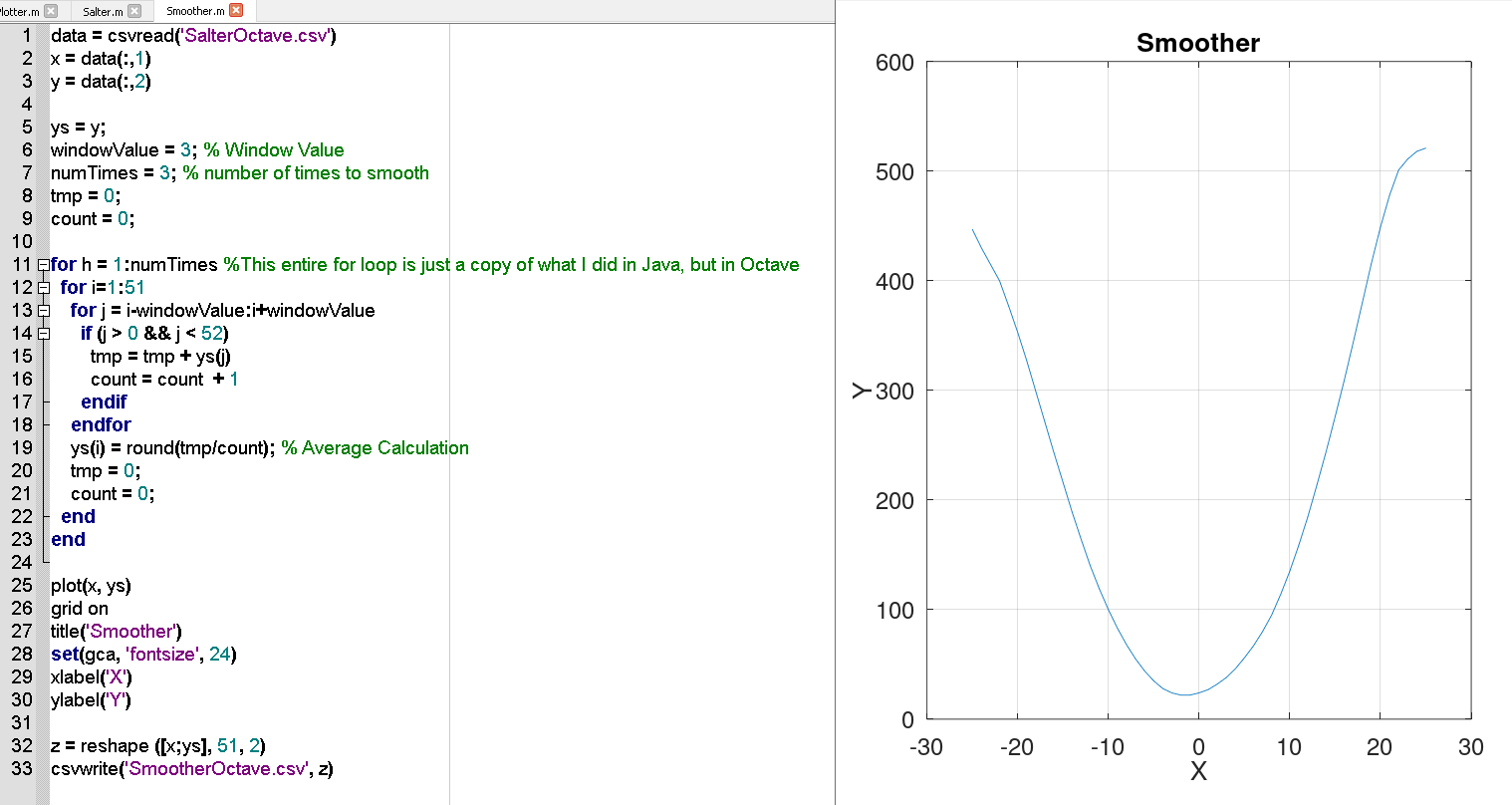
Line 17 was the line that gave me the most trouble, as I kept having that “vertical dimensions mismatch” error whenever I had “[x;y]” in place of “z”.

## Smoothing: Journal

Smoothing was easier than salting, but also harder than plotting. There seemed to be quite a few ways to implement a smoother in octave, though I liked the smoother I made in Java more than the smoothers that I found. Thus I recreated my Java code in Octave. Also, I followed along with this [youtube video](https://www.youtube.com/watch?v=l-P0QlJKGRU&ab_channel=DSPcorner), but then I decided to go with a recreation of my java code rather than implementing his idea.

## Smoothing: Results

The data is in the google sheet linked above. Here is the script and graph.



## Overall Results

This project has given me a new perspective on programming. Of course I enjoy programming in Java, but if I’m assigned a task geared towards plotting graphs, using a program (such as matlab/octave) geared towards plotting graphs will be quicker and likely more effective. It took me less time overall to learn the basics of coding in Octave and create the PSS than it did to create the PSS in Java, a language that I have long known the basics for. Overall, this portion of the project was a lot of fun. Here is the combined graph from my google sheet:

