Essays and Documentation for Project 2

In order of Completion

By Alexander Mortillite

Contents

[Salt Plot Smoother Java, No Libraries! 2](#_Toc133482623)

[Outputs in .csv files, Standard, Salted then Smoothed. 2](#_Toc133482624)

[Salt Plot Smoother Octave, No libraries 3](#_Toc133482625)

[Octave Demonstration 3](#_Toc133482626)

[Salt Plot Smoother Java, Using Apache Math and JFreeChart! 4](#_Toc133482627)

[Figure A: The smoother code 4](#_Toc133482628)

[Figure B: The Salter Code: 4](#_Toc133482629)

[Configurations for Graphs 5](#_Toc133482630)

[Figures C, D, E: Graphs 5](#_Toc133482631)

[.png exporter 5](#_Toc133482632)

[Statistics Library 2 5](#_Toc133482633)

[GUI Demonstration 6](#_Toc133482634)

[Usage of the GUI 7](#_Toc133482635)

[References/ Work Cited: 9](#_Toc133482636)

# Salt Plot Smoother Java, No Libraries!

The concept of this program was seemingly simple: pick a formula, run that formula through some sort of loop, and commit the resulting x and y values to a csv file that can be later read from. This seemingly simple program offered some of the most painstaking troubleshooting I have ever dealt with.

Step 1: Storing the data points of the quadratic formula; This part of the code was the easiest to set up and get running, a tutorial on (reference 1) was all that was needed to refresh my memory on the process of writing to a file, and a small amount of experimentation proved fruitful, enabling the proper writing to the .csv file holding all of the original values.

Step 2: The Salting portion of the program was easier to implement than step 1 because it was mostly the same code but with the additional option to write a given amount of noise to the y-data points. I used a homemade helper method to create a csv based on a fed in Array List, so all I had to do was read in the original .csv I made, salt the values of the list by a predetermined amount and then pass them through the method, turning them back into a different .csv file.

Step 3, which was smoothing was by far the most difficult part of the entire program. I had several ideas about how to go about it, the first of which was to run through the array, take multiple values, average them out and then change all the values averaged to their average. I learned quickly that wasn’t the appropriate approach because of the strange plateaus in the graph which makes sense, but it was an oversight that I had at the time. After learning that was not the right path, I had a change of pace and changed my strategy. Rather than changing entire groups of data to the average of themselves, I should change single data points to the average of their surroundings. I learned a valuable lesson in documentation when I took a hiatus after getting a semi-working model of what I had wanted but had gotten tired of troubleshooting. Going back to the drawing board I figured out a new way of logically going about the moving average before which worked so much better than the 2nd attempt, I had done but still had strange bugs inside it around the edge cases, because I wasn’t omitting non-existent edge values from the denominator when. Then after implementing my solution, it worked like a charm. Lastly, I just had to add the intensity of the smoothing modifier which was just a for-loop surrounding my original for loop which ran the number of times specified by the user.

## Outputs in .csv files, Standard, Salted then Smoothed.

Text

Description automatically generatedA picture containing text, device, meter

Description automatically generatedText

Description automatically generated

# Salt Plot Smoother Octave, No libraries

Octave as a programming language is similar yet so foreign when moving from pure java, there are many things about it that are very interesting, one of which was that omitting a semicolon from a line won’t throw a compiler error, but rather print out the action that you are taking. I found that to be incredibly useful because it made following the program much easier. The built in libraries found in octave also made the program much easier to write as well. As an example, the built in movmean () function did what my hard work in java performed at a fraction of the thought power to implement. The largest complaint that I have with the programming language is that the documentation regarding the random function which I used for random wasn’t very clear because there are a few depreciated methods that just no longer work.

To learn how to use octave I used a few online sites, such as…., but mainly I followed their documentation whenever I ran into an issue. As I mentioned previously the random function gave me the largest amount of trouble, I had found several implementations that would either always give me the same output (Defeats the point of a random method) or would require a helper method to update the system time, and even then, I couldn’t get it to work. This was until I found the randi () method in the documentation. Like a savior coming from the heavens, it provided me with what I needed, the ability to implement noise into the graph. From there I learned how to export datasets as .csv files and read them using the dlmread function. Lastly, I used the hold on and hold off to plot all three .csv files onto the same graph which worked well.

There is no user input in the script that I created, rather the values regarding the calculations can be modified in the script before it is run again. I have several variables, x which represents the low and high range of applied salting. windowSize which is used to determine the quantity of numbers within the mean calculation. Finally grit, which can be found in all my smoothing programs, represents the number of times that the smoother will run through the data, I decided on the name grit because I related the smoother to sandpaper and grit is used to determine how fine the paper is, so I felt there is a solid connection.

## Octave Demonstration

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generatedThere isn’t much to demonstrate, so I will include the inputs and the outputs

# Salt Plot Smoother Java, Using Apache Math and JFreeChart!

Introducing the APIs made me re-think my program entirely, initially I had just copied the non-API version of the code so that I could just. In previous iterations of my codebase, I had been changing everything into .csv files and then turning them into specific files names that could be read and written to. With the catch that it could only utilize pre-specified pathnames. In the newest version of the code, this has changed slightly, now you can enter in pathnames for any .csv file with the proper formatting. The added benefit is that now it is possible to quickly adapt the code for any type of formula to be entered in its place. This code is very modular and can be adapted very quickly, if you have the pathname of the .csv you are pulling data from.

Learning how to import .jar files into my IDE was more of a hassle than it should have been but eventually I got it to work. The only catch was that I had to use older versions of the APIs we had to use. I was never able to figure out the reason why the newest version of Apache commons would refuse to import its sub libraries, but I know that the math3 library still works! With this rendition of the code, I also made the decision to create a second class called SaltPlotSmootherLibrary whose sole purpose is to supply common methods that allow me to turn arrays into .csv files and then subsequently clean the .csv file and create an array from the data. The only downside to this process is that I lose the x-axis upper and lower bounds data, so I need to include that in the method calls to ensure that no data is lost.

The hardest part about this code was how the Apache Commons library was going to be useful within the smoother. That was until I learned that DescriptiveStatistics was essentially a stack that would pop the information out when getMean () was called.

## Figure A: The smoother code

Text

Description automatically generated

## Figure B: The Salter Code:

Text

Description automatically generated

I know this is small, it will be in the SaltPlotSmootherAPI class if you would like to have a closer look at it.

I wanted to highlight what I meant by modular, and not being tied to a specific formula.

## Configurations for Graphs

Text

Description automatically generated

I picked these numbers because I thought they were a good demonstration of what the smoother can do

## Figures C, D, E: Graphs

Chart, line chart

Description automatically generated

Chart

Description automatically generated

Chart, line chart

Description automatically generated

## .png exporter

Chart

Description automatically generatedThe other thing this code does is exports the generated graphs as .png files, I did this to keep in line with the style of the programming assignments in this project.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

I made these small because they are identical to the previous ones, but I wanted you to see that they exist.

# Statistics Library 2

Within my StatsLibrary2 I have three key methods

* Hyper Geometric
* Poisson Distribution
* tchebysheffsTheorem

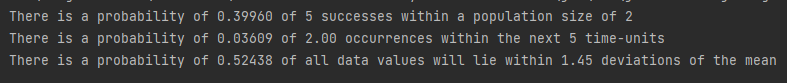
The only interesting is that the Hyper-Geometric formula, which uses the Big Integer Class to ensure that large numbers can be calculated when using combinations.

A screenshot of a computer

Description automatically generated

Graphical user interface

Description automatically generated



# GUI Demonstration

To spice up the API program, I decided to add a GUI to it. From this GUI you can quickly modify the values of the .csv files and rapidly test different results. While missing the more complex features like changing the pathname or entering in your own pathname, I still think this addition to the code is very useful for rapid feature demonstration. Another benefit is that it also saves a .png of the graphs after the project closes.

Graphical user interface

Description automatically generated The GUI is built on the principles of MVC with an emphasis on being able to change the model out quickly, it would take 7 lines of code to be different to switch the GUI from using the API version of SaltPlotSmooth to the non-API version, and for that matter, any other type of program that can salt and smooth. To run this GUI, it can be started by calling the start GUI method in Tester class.

1. Range fields: lower bound, then upper bound
2. Salt/Grit fields: Salt intensity then grit applied.
3. Window Size field: Allows user to input the window size.
4. Create button: Creates a standard .csv file based on upper and lower bound.
5. Salt button: Salts the .csv created by the create button.
6. Smooth button: Smooths the salted .csv file when this button.
7. Graph button: Will create 3 graphs based on results of the buttons.
8. Clear: Quickly wipes the data from the fields, for ease of use

## Usage of the GUI

Graphical user interface

Description automatically generated

Chart, line chart

Description automatically generatedDemonstration, input the values as shown, press create, Salt, then smooth and then the Graph button in that order

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Graphical user interface

Description automatically generatedThis is the result of pressing the clear button.

# References/ Work Cited:

1. [https://www.geeksforgeeks.org/filewriter-class-in-java/#](https://www.geeksforgeeks.org/filewriter-class-in-java/)
2. <https://zetcode.com/java/jfreechart/>
3. <https://commons.apache.org/proper/commons-math/userguide/stat.html>
4. <https://docs.octave.org/v6.3.0/Statistics-on-Sliding-Windows-of-Data.html#XREFmovmean>
5. <https://www.mathworks.com/matlabcentral/answers/23591-how-do-i-combine-multiple-plots-in-one-graph>
6. <https://www.mathworks.com/matlabcentral/answers/292226-how-to-plot-multiple-lines-in-a-graph>
7. <https://octave.sourceforge.io/octave/function/csvread.html>
8. <https://docs.octave.org/v6.3.0/Statistics-on-Sliding-Windows-of-Data.html>