Chris Ricchi

November 11 2023

Probability and Applied Stats

**Comparative Analysis of Salting and Smoothing: Java and Octave**

**Abstract:** This research paper presents a comprehensive exploration of three different programs developed to handle data manipulation such as importing, salting and smoothing, and plotting. My study compares my personal development experiences and outcomes of these three programs, each implemented in different environments: the first in Java from scratch, the second in Java utilizing JFreeChart and Apache Common Math Library, and the third in Octave. My primary goal of this paper is to provide insight into the ease of implementation, simplicity of use, and efficiency of each unique programming approach. My overall findings revealed that the Octave implementation stands out as the most straightforward, mainly due to its inherent simplicity and specialized functionality. The second program, which utilizes JFreeChart and Apache, proved slightly easier due to the pre-existence of code and the simple integration of the libraries that already complete most of the work for me. The last program, created from scratch, was certainly the most challenging because it required substantial effort in coding, specifically debugging.

**Octave**

The implementation of the Octave program was very straightforward as I already have personal experience with both MATLAB and Octave. Using my previous knowledge, I completed the program relatively quickly, only needing to reference the official Octave documentation. The most challenging portion of the program was implementing a user-interface to allow selection of options, which even then after reviewing the documentation was very straightforward. The program allows users to interactively choose between salting, smoothing, exporting data, or exiting. The functionality seamlessly loads data from a specified .CSV file, enabling users to import their own datasets with ease and manipulate the points.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

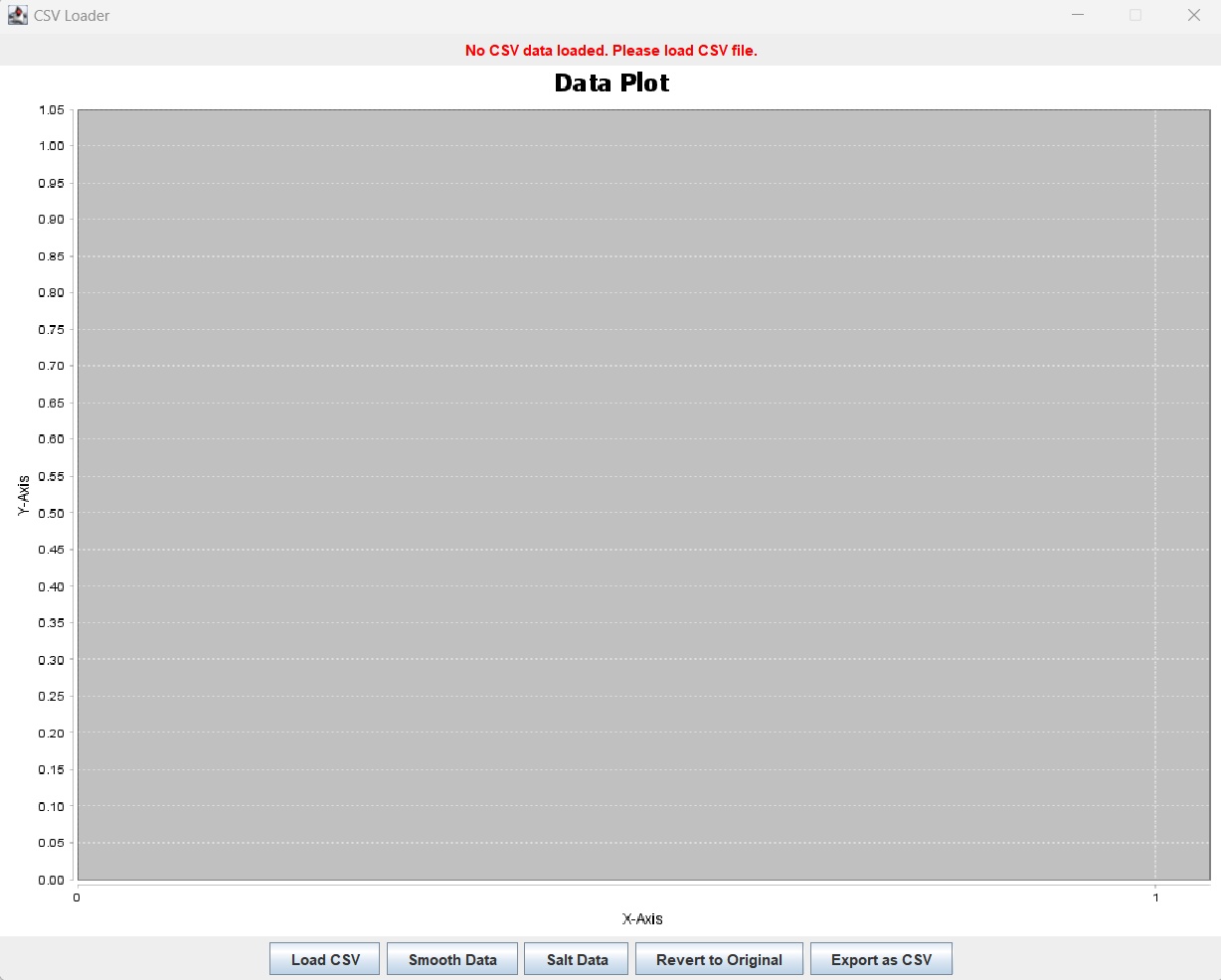
Description automatically generatedA screenshot of a computer screen

Description automatically generated

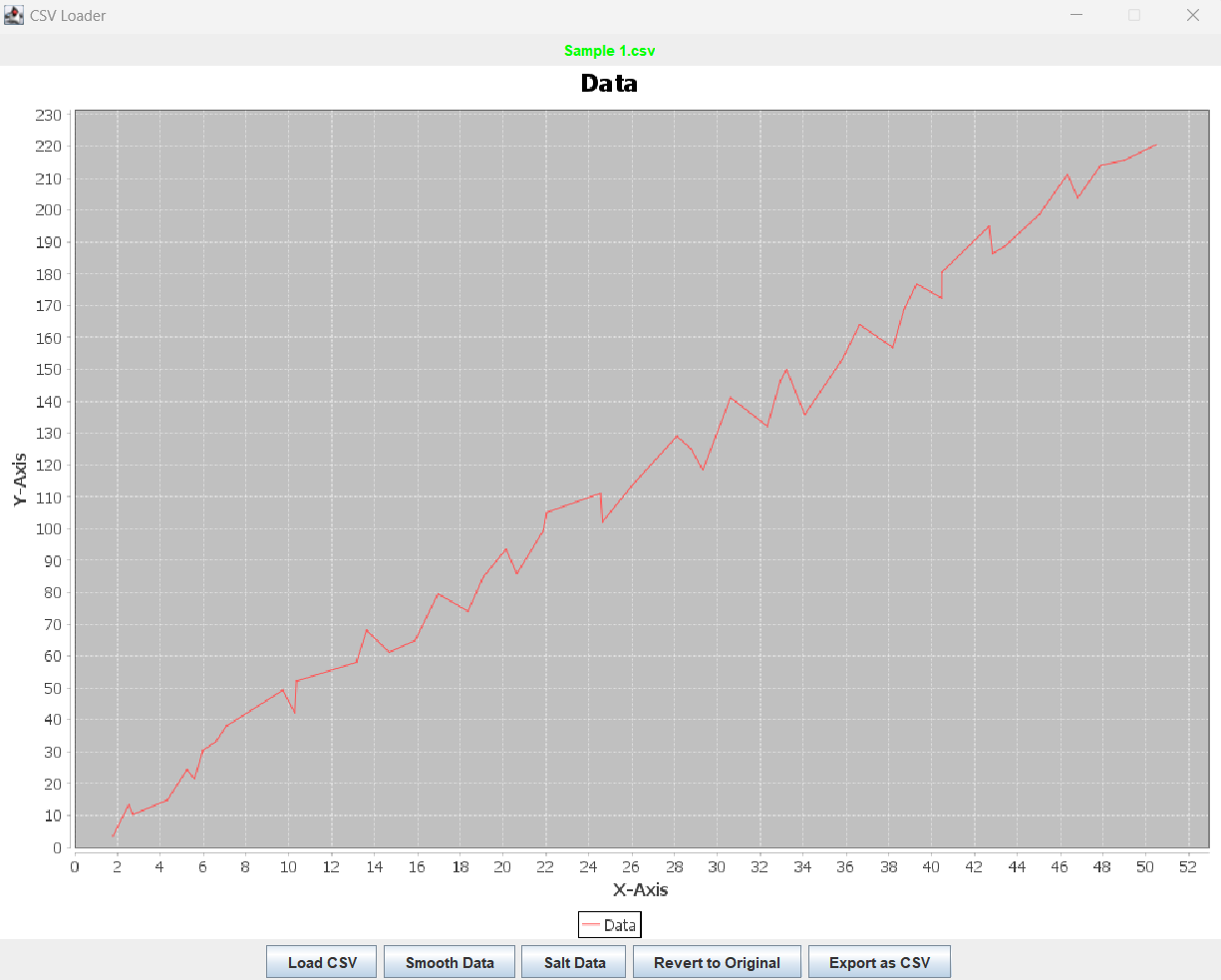
Octave is designed to allow users to easily plot data by calling the plot function, so there was no need for any major programming at all. Because of this, the Octave implementation was definitely the simplest of the three programs to fully implement.

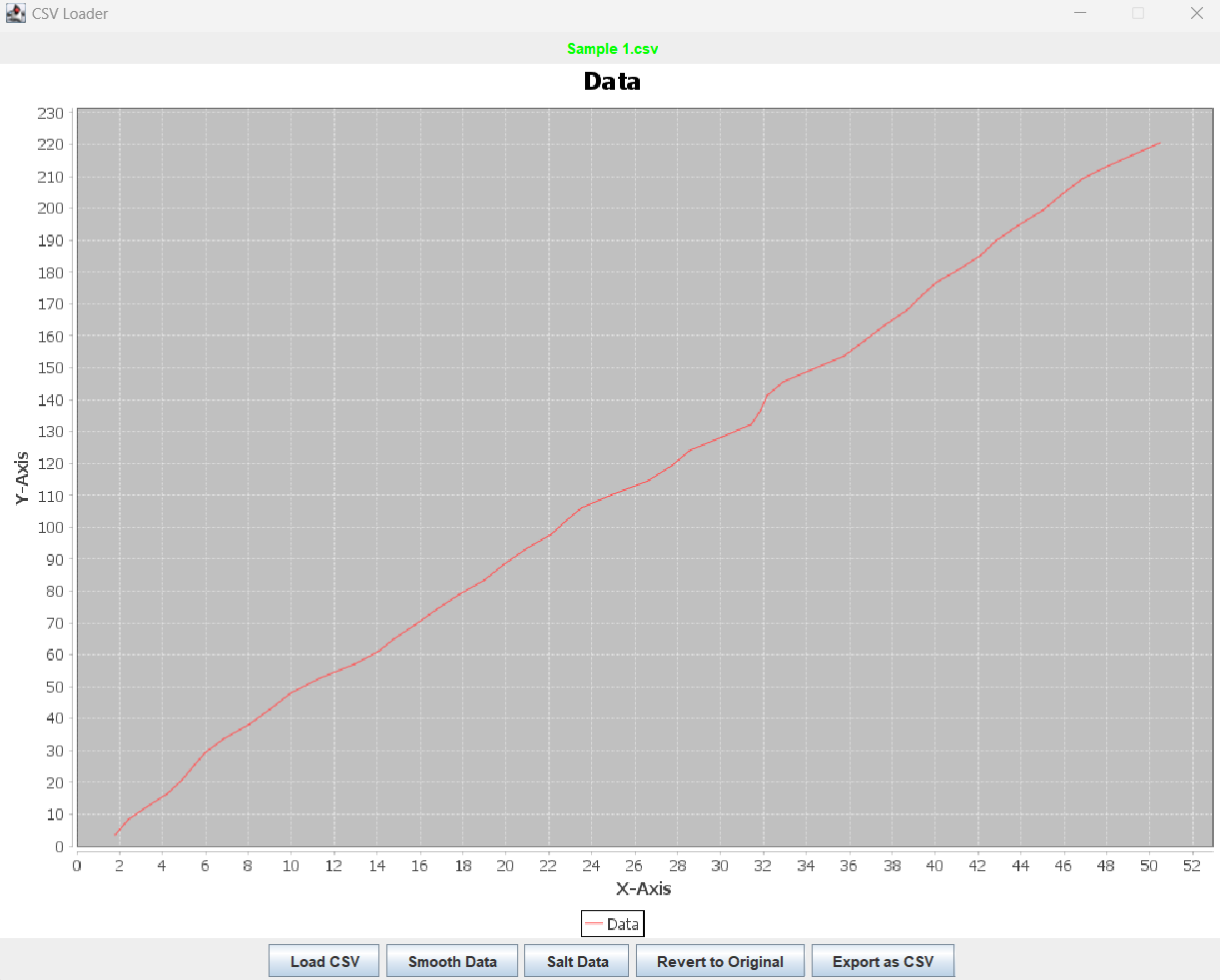
**JFreeChart and Apache Math**

The implementation of the JFreeChart and Apache Commons Math program presented itself as moderately difficult. Because I programmed the third program which salts, smooths, and plots before I programmed this one, I already had most of the main functions created which provided a significant advantage. However, the main hurdles emerged while importing the external libraries, which was surprisingly difficult to tinker with correctly, mainly because a lot of the official documentation was outdated so the imports on their website were incorrect. However, I learned a handy shortcut in Eclipse (CTRL+SHIFT+O) which automatically imports all required libraries that are being used in the program. After I got the libraries fully imported and working as intended, the program’s overall functionality relied mainly on their capabilities, so the process of plotting and data manipulation was rather easy.



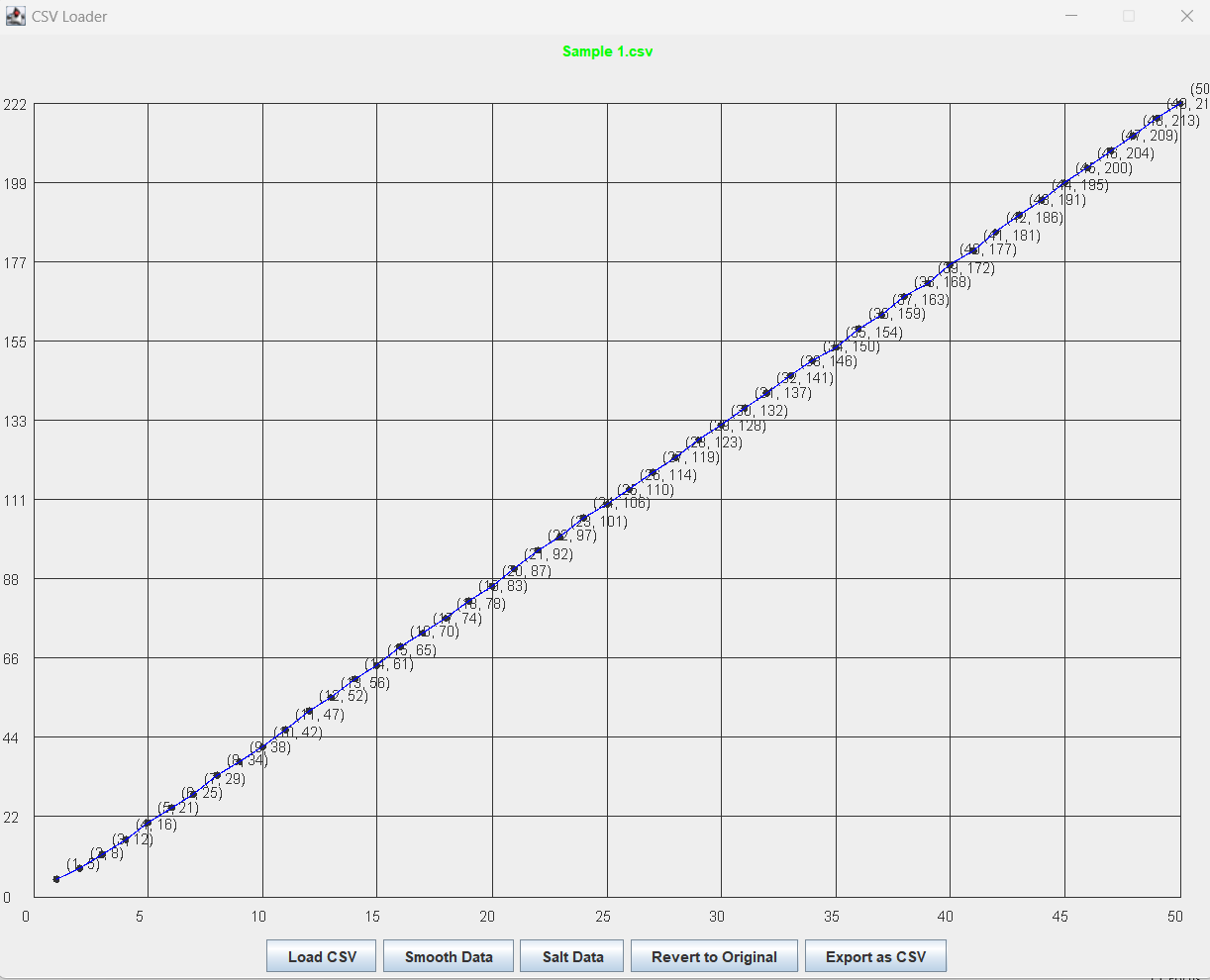
Another intricate aspect of this program was the debugging process. Issues relating to library compatibility and method usage were very difficult to debug because I didn’t have the proper imports established. The official documentation has hundreds of versions of each library, each with their own refined (and oftentimes renamed) methods which meant a lot of times I was using method names that had been changed in a previous version. However, when all issues were fixed, the program was essentially fully completed as I already had the skeleton of the program completed beforehand. In short, the JFreeChart and Apache Math implementation required more knowledge and research for me than the Octave program did specifically due to the library integration complexities. However, the advantage of the pre-existing code and the functionalities of the libraries made this program quite simple once all was figured out.





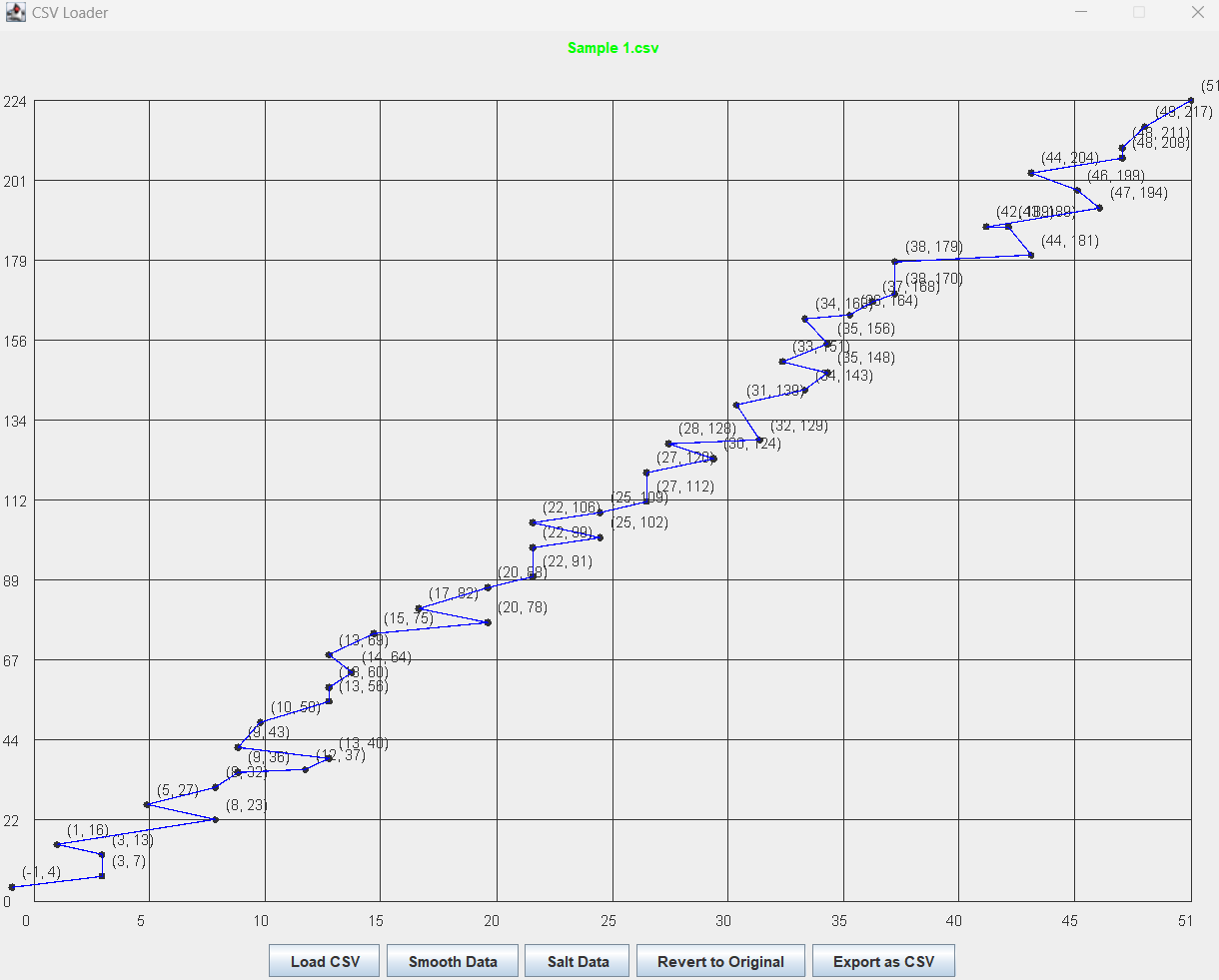
**Java From Scratch**

It came as no surprise that the development of the Java program from scratch was easily the most challenging to accomplish. It required substantial effort and time investment. One of the primary issues that arose during implementation was displaying the plot to graph the points. More specifically, the task of dynamically scaling the graph to accommodate any dataset of varying size. For example, a dataset with the largest number of X being 500 and the largest number of Y being 500, would need to have a much larger graph as opposed to a dataset with the largest numbers of X and Y being 5 and 10. Determining the appropriate scaling factor for both axes, especially when dealing with datasets featuring widely varying values, proved to be the most challenging portion of the program. The dynamic scaling algorithm first determines the two maximum X and Y values, and then scales the graph accordingly. The algorithm calculates the scaling factor to map the data points to appropriate positions to prevent the graph from distorting.

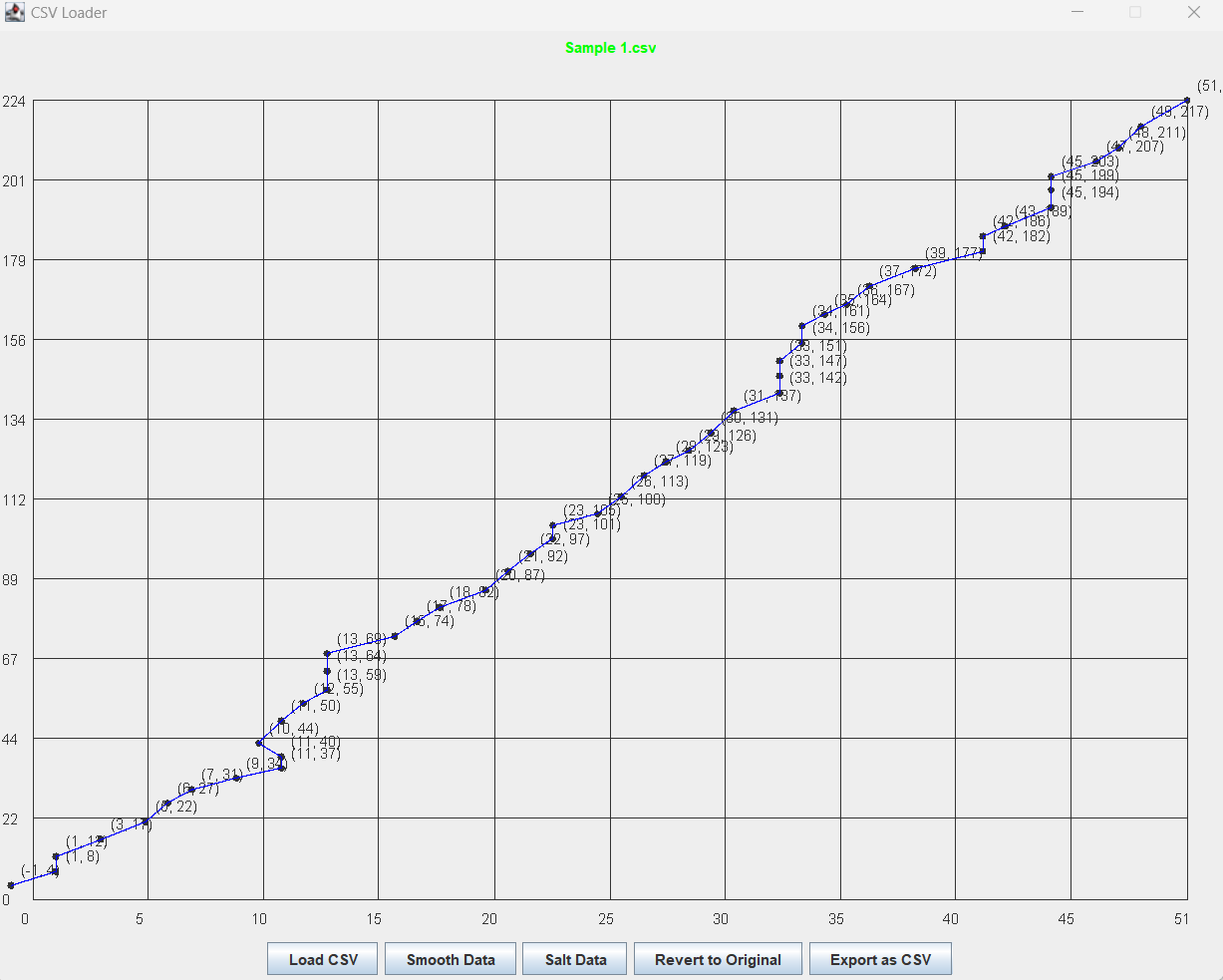


Likewise, creating the salting and smoothing methods from scratch added an additional layer of difficulty. Ensuring the algorithms chose random variations and implementing a simple moving average algorithm for smoothing required more of a mathematical approach as opposed to a programmatic approach.

The salting algorithm uses random variation to simulate noise in the dataset. For each data point, a random value within a specified range is added to both the X and Y values. The salt range can be manually adjusted, allowing control over the variation as well.



The smoothing algorithm, on the other hand, uses a simple moving average algorithm to reduce noise in the dataset. The algorithm begins by iterating through data points, replacing each point with the average of itself and its neighboring points. This moving average creates a simple smoothness to the graph.



In summary, the program from scratch required a significant more thought and focused more on a mathematical approach. Data reading and dynamic scaling proved to be the most challenging part of the program. While the salting and smoothing algorithms presented their own difficulties, they were more straightforward as they are simply functions being called on a dataset of numbers. Dynamically scaling the graph to accompany any sized dataset was more challenging to implement, and was also difficult to debug properly.

**Conclusion**

Exploring the three distinct approaches to the same problem in different environments was genuinely an eye-opening experience. It demonstrated to me how a vast knowledge of different programming environments can assist you in creating a solution. If this problem was presented to me in a real-life scenario, I’d know to immediately choose Octave to solve it as it is an environment that is based around mathematics and visualizing data. Each program had its unique set of challenges; crafting the Java program from scratch proved to be the most difficult; it demanded extensive efforts and obstacles that took many hours to work through. As difficult as it was, completing it before the JFreeChart and Apache program proved to be worth my time, as transitioning to that program was much more straightforward after having the main works already completed. Nevertheless, even that program proved to be difficult at times by having to learn how to properly import and utilize multiple external libraries. Octave, on the other hand, offered a more streamlined path to success by using an environment that’s already catered to solving this type of problem. All three programs varied heavily in difficulty and experience – Java from scratch being the most challenging, JFreeChart moderately so, and Octave offering the smoothest ride. As stated previously, these experiences underscore the importance of choosing the right tool for the task as it can cut down on time spent significantly. Ultimately, this programming experience honed many technical skills and also emphasized the significance of pragmatic choices in software development.