Plotting, Salting, and Smoothing Exercise Report

Christian Simpson

CSCI-3327-001 Probability and Applied Statistics

Byron Hoy

April 29th, 2024

The goal of this assignment was to write a set of programs which could plot, salt, and smooth a data set. By displaying the results of the set of values after each program has completed its run, we learned to observe and make inferences about the data in question. We achieved this goal in two ways: An original implementation of the functions, as well as importing external libraries to do the work for us. By learning how to import and implement external libraries, we learned an essential work skill: writing and testing programs with the aid of functions other programmers have already implemented, saving both time and effort on our part. I've explained each section of the programs below, along with any challenges and my results.

Plotting, or taking some number of inputs, and evaluating a function with them is the first step in visually representing a set of data. These inputs and outputs can be mapped to any sort of function or represent a particular data set in many ways. We could, for example, take the average wind speed in different cities, medical information concerning a particular disease in patients of different ages, or the average score of players in a sport over the course of a tournament. Here, we used simple math to display inputs and their associated outputs from a chosen function. The parabola is a simple one, so I chose to use it to have a clear example that could be verified visually. Plotted points were sent as output to a Comma Separated Values file (.csv), where a user can plot the data in a program such as excel using graphs and charts. We also imported the JFreeChart Library, which was used to display the data across all three programs. See an example of the output below (Fig. 1.1).

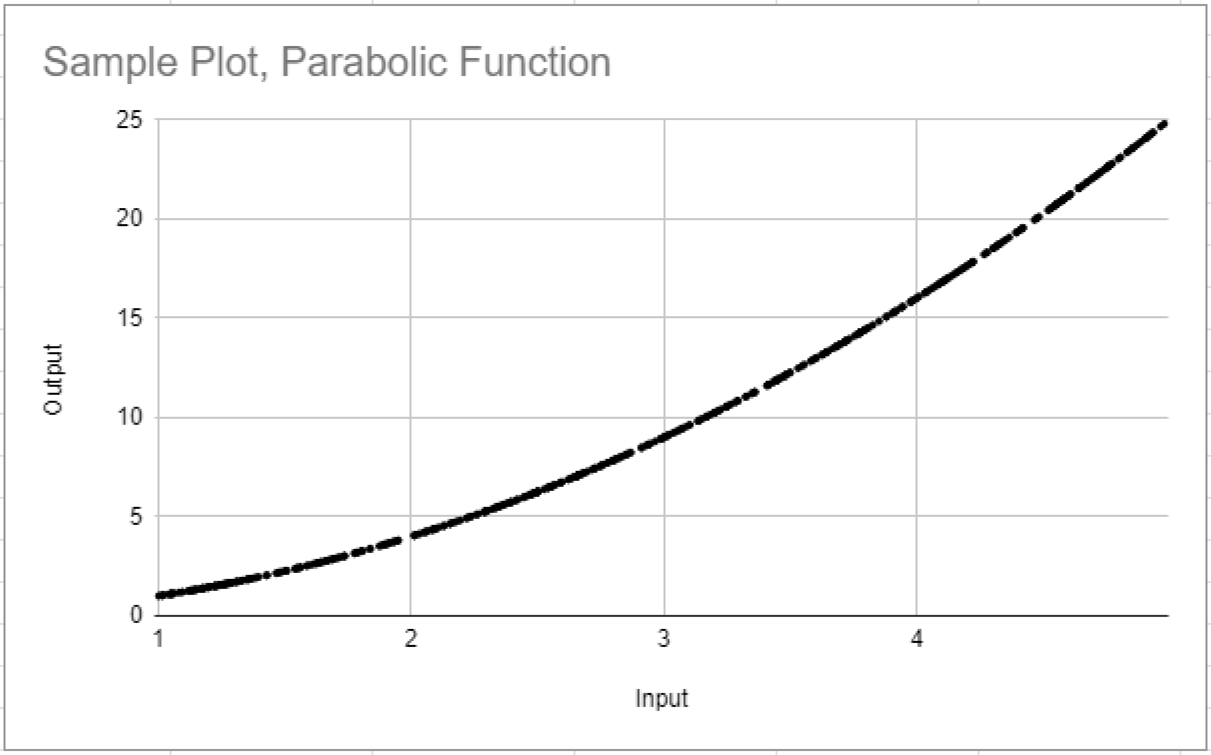


Fig 1.1: Plotted Data in Excel.

Next, we were tasked with Salting the data. Salting is the practice of purposely obscuring data with some other value wholly unassociated with the actual data itself. This could be excess values appended to a function's output, location data not including specific places of interest to a study, or movie listings around a single showing we want to learn more about. By adding a salt to our data, we can both obscure the output as well as simulate real-world behavior in collected data. Often, when studying a population, we gather a lot of data we are not interested in, or that does not serve our interests. Again, see below for an example of the output in Excel (Fig. 1.2).

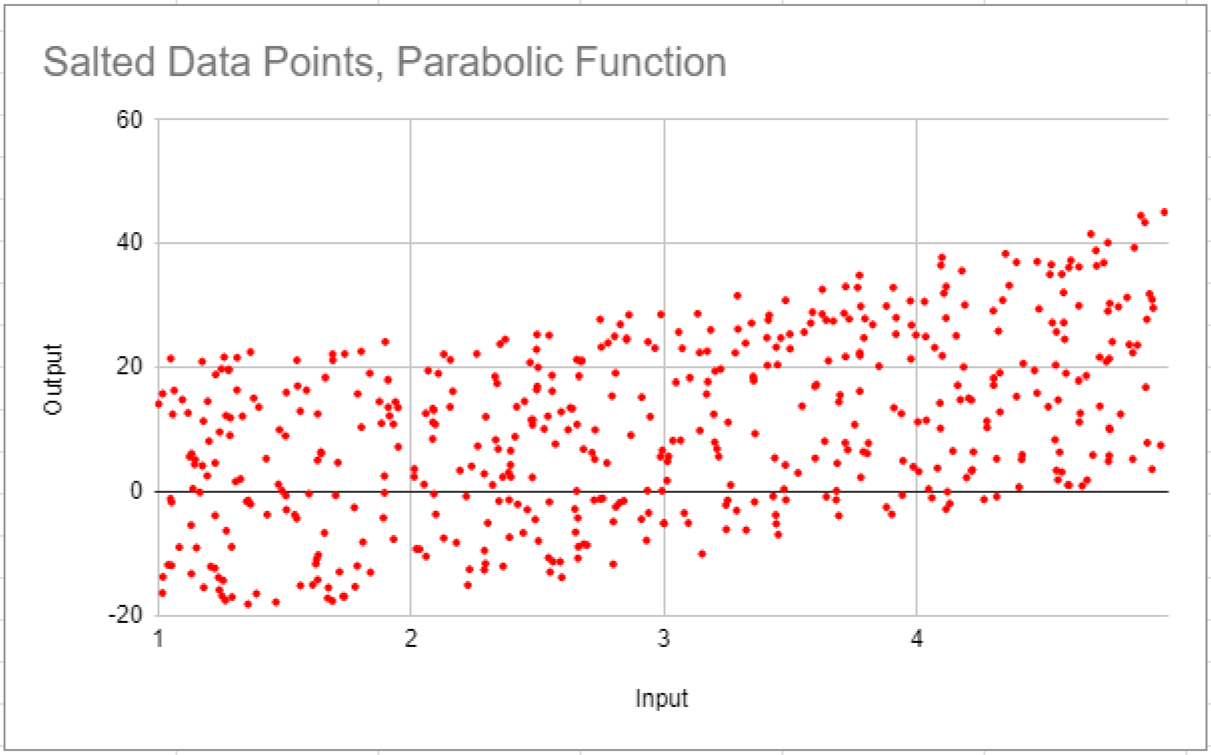


Fig 1.2: Salted Data in Excel.

Finally, we were tasked to Smooth the data. Smoothing is the process of eliminating errors in the data by using averages. By taking a number of values around a data point and averaging them, we can begin to see the original trend in a data set before a salt was applied. With this practice, we can more closely observe trends in a data set holding many values apart from our needed observations. This area of the project, admittedly, gave me the most trouble. Even when testing several different window values, averaging using both the point in question and just the area around it, and in smaller and larger data sets (n < 1000, n > 5000), I could not get my salted data to replicate the original curve of the data set. See below for examples of the outputs (Fig. 1.3). Lastly, an example of the output generated by implementing Apache Math and JFreeChart is displayed (Fig 1.4).

In the future, I would like to attempt to correct my smoothing algorithms to display an accurate output.

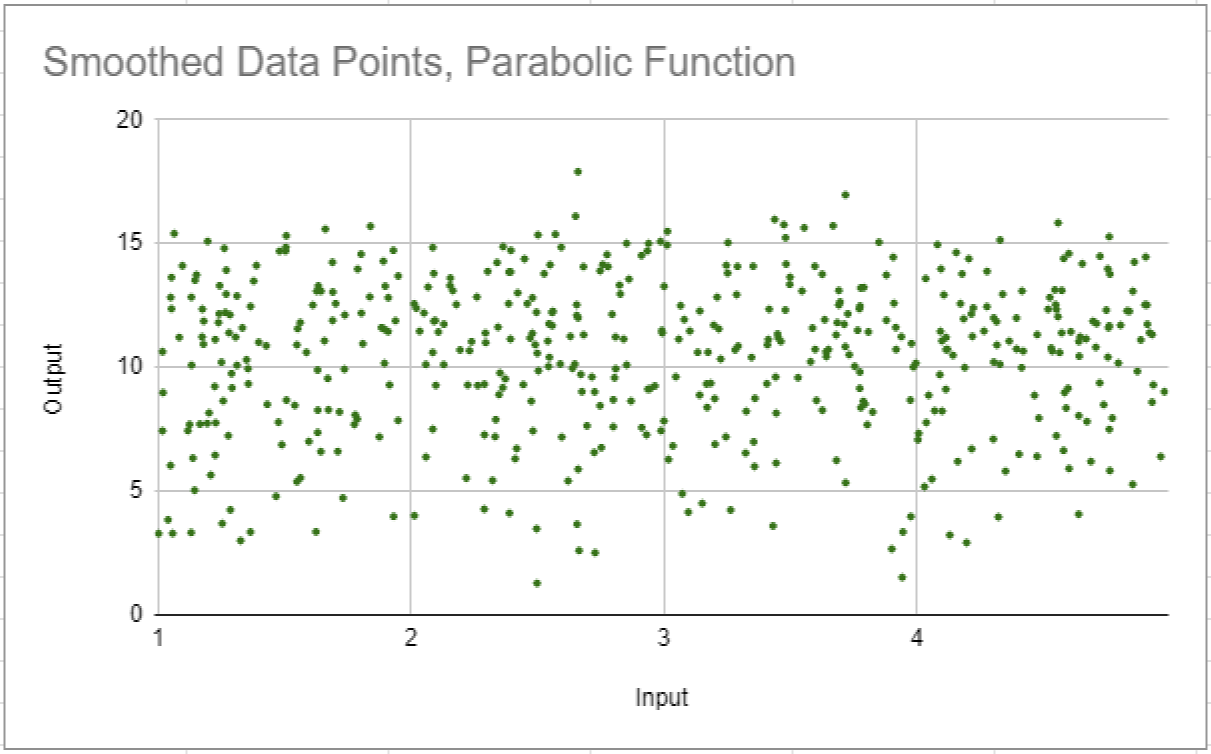
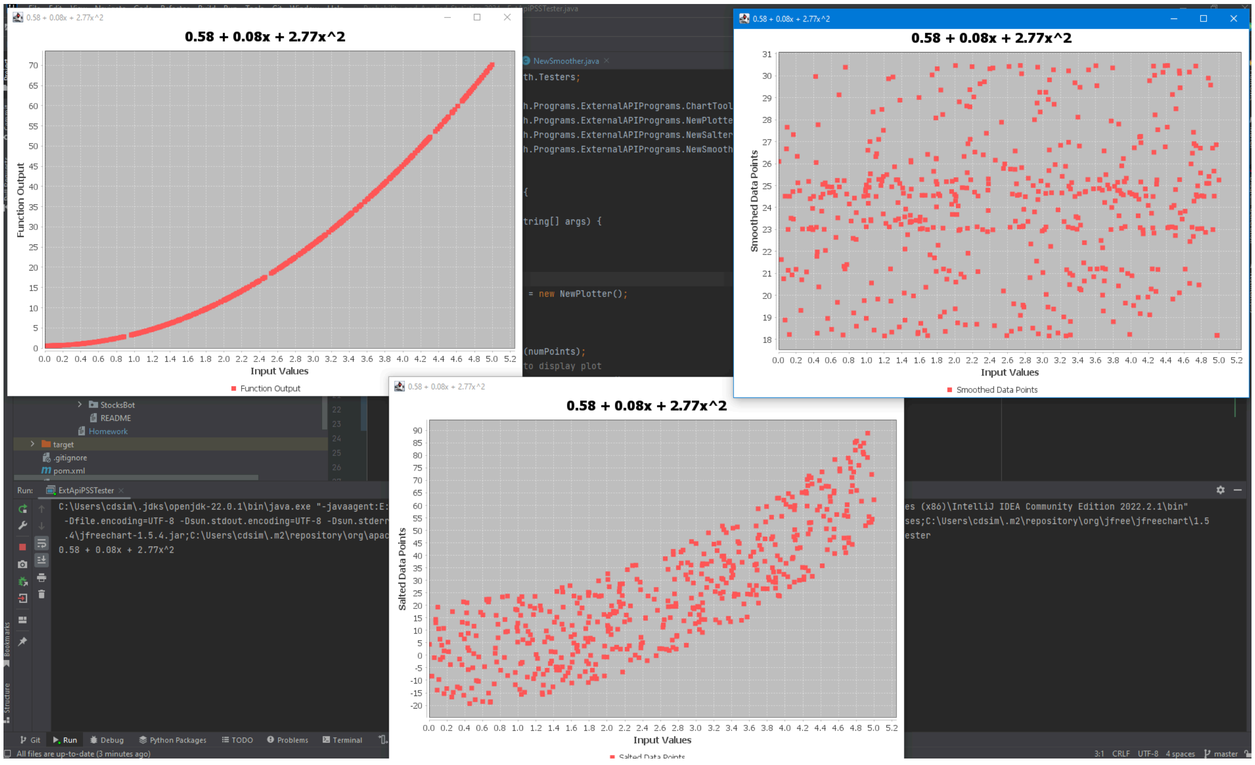


Fig 1.3 Smoothed Data Points in Excel.



```
th.Testers;

h.Programs.ExternalAPIPrograms.ChartTool
h.Programs.ExternalAPIPrograms.NewPlotte
h.Programs.ExternalAPIPrograms.NewSalter
h.Programs.ExternalAPIPrograms.NewSmooth


{

tring[] args) {



 = new NewPlotter();



(numPoints);
to display plot
```

```
C:\Users\cdsim\.jdks\openjdk-22.0.1\bin\java.exe "-javaagent:E:     es (x86)\IntelliJ IDEA Community Edition 2022.2.1\bin"
 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr     ses;C:\Users\cdsim\.m2\repository\org\jfree\jfreechart\1.5
 .4\jfreechart-1.5.4.jar;C:\Users\cdsim\.m2\repository\org\apac     ester
0.58 + 0.08x + 2.77x^2
```

All files are up-to-date (3 minutes ago)

Fig 1.4: Example Outputs generated using Apache Math3 and displayed using JFreeChart.