**Java CSV Demo & Report – Project 2**

**Anthony D’Alessandro**

**Table of Contents**

Methods……………………………………………………………………………………………1

Plotter……………………………………………………………………………………...1

Salter………………………………………………………………………………………2

Smoother…………………………………………………………………………………..3

Small Scale Trial…………………………………………………………………………………..4

Plotter……………………………………………………………………………………...4

Salter………………………………………………………………………………………4

Smoother Run 1…………………………………………………………………………...5

Smoother Run 2…………………………………………………………………………...5

Smoother Run 3…………………………………………………………………………...6

Larger Scale Trial………………………………………………………………………………….7

Plotter…………………………………………………………………………………...…7

Salter………………………………………………………………………………………7

Smoother Run 1……………………………………………………………………...……8

Smoother Run 2…………………………………………………………………………...8

Smoother Run 3…………………………………………………………………………...9

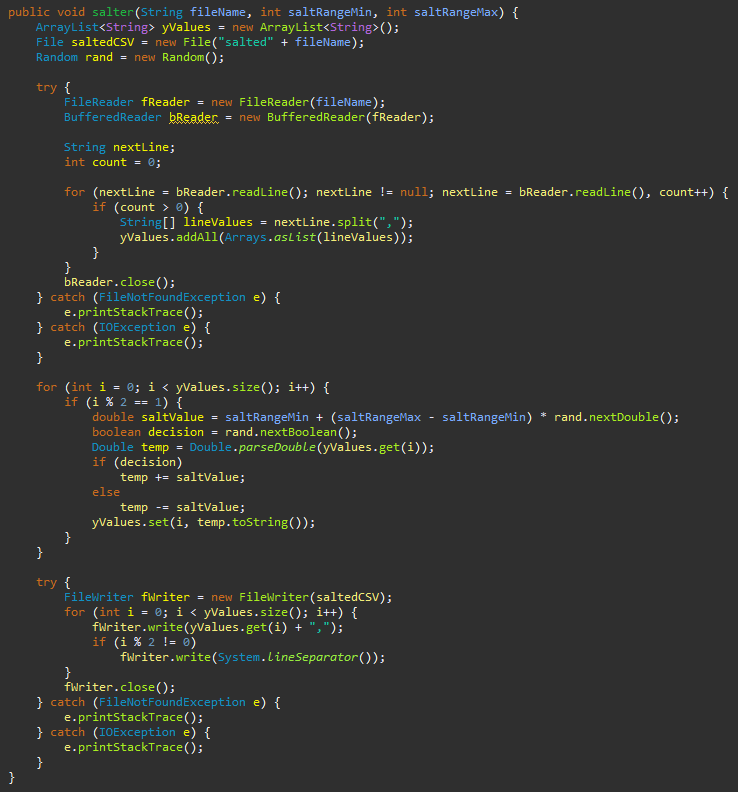
Discussion……………………………………………………………………………………....…9

**Methods**

Plotter



Salter



Smoother



**Small Scale Trial**

Function Used:

Plotter

* Data generated using a population range from [1, 50] with an increment of 0.5



Salter

* Data salted with a salt range from [0, 250]



Smoother

* Data smoothed with a window size of 8

Smoother (Run 2)

* Smoother ran on first set of smoothed data with a window size of 8



Smoother (Run 3)

* Smoother ran on second set of smoothed data with a window size of 8



**Larger Scale Trial**

Function Used:

Plotter

* Data generated using a population range from [1, 500] with an increment of 0.5

Salter

* Data salted with a salt range from [0, 100000]



Smoother

* Data smoothed with a window size of 25



Smoother (Run 2)

* Smoother ran on first set of smoothed data with a window size of 25



Smoother (Run 3)

* Smoother ran on second set of smoothed data with a window size of 25



**Discussion**

The main goals of this program were to use Java generate a collection of data from some function such as , write the data points to a CSV file, read the values from the file and salt the data, write those salted values to another CSV file, and finally read the values from that file and smooth the data. When smoothing the data, the objective was to try and bring the data back as close as possible to its original form. Each CSV file was opened through Excel and a simple graph was generated to visualize the data and make it easier to check if the program was working as intended.

Program Structure

The program starts with the plotter function that takes parameters such as a file name, a minimum and maximum value for the range of x-values, as well as an interval for how much each x-value should be incremented by. A new file object is created in which a FileWriter is used to write each line to the CSV file. A for loop runs from the minimum value up to the maximum value using the interval to increment each time where the function mentioned above is evaluated at every x-value to generate a corresponding y-value. For each pass of the loop, the x-value and its y-value are simply written to the CSV file separated by a comma.

The next part of the program is the salter method that takes parameters including a file name and a minimum and maximum value for the range of the “salt” value. It starts by using a FileReader and BufferedReader to parse through the first CSV file and extract the x and y-values. Each line is split by a comma, stored into a string array, and then added into an ArrayList. The salting functionality follows where the program loops through the newly created ArrayList of x and y-values, generates a random “salt” value and a random Boolean, and then uses the Boolean to decide whether to add or subtract the salt value to or from the current y-value in the ArrayList. The program then loops through the ArrayList of x-values and salted y-values and uses a FileWriter to write the data to a new CSV file.

The final part of the program is the smoother method that takes a file name and a “window” value as parameters. It starts just like the salter method in which it uses a FileReader and BufferedReader to read from the CSV file of salted data and store the x and y-values into an ArrayList. The smoothing section of this method begins by looping through the extracted data, calculating a rolling average of the y-values, and then storing the rolling average that corresponds with each y-value in the ArrayList of x and y-values. This is done by simply creating a temporary ArrayList to hold each y-value and its surrounding y-values using the “window” to decide the surrounding limits. The rolling average is then computed by taking the sum of the y-values in that temporary ArrayList and dividing that by the size of the list or the number of elements. Finally, the program loops through the ArrayList of x-values and smoothed y-values and, similar to the first two methods, uses a FileWriter to write the data to a new CSV file.

Trials and Observations

The first trial was completed using a population with a range of 1 to 50 with an increment of 0.5 for a total of 100 x-value data points. The plotter method worked as expected and simply evaluated the function for each x-value and wrote the x-y pairs to a CSV file. The graph that was generated in Excel visualized the results as a gradually upward sloping curve. The data was then salted with a salt range of 0 to 250 and saved to a new CSV file. The Excel graph showed that the data still followed a general upward slope but the data points were sporadically shifted up or down resulting in a vertical “zig-zag” pattern. The smoother method was then run on the salted data with a window value of 8 and the results were saved to a new CSV file. The graph of the smoothed data displayed a much more uniform curve that was relatively “bumpy” but resembled something similar to the original curve from the plotter function. While the smoother method seemed to work as expected, it was run two more times using the smoothed data from each previous test. The graphs that were generated from the second and third smoother runs were very similar in which they both displayed curves that were much smoother than the first smoothing test but seemed to tail off towards the end and created a flat segment. A potential solution to this might be to increase the window value to include more surrounding y-values in the rolling average since towards the end of the curve, there are no successive values.

The second trial was completed using a population with a range of 1 to 500 with an increment of 0.5 for a total of 1000 x-value data points. Similar to the first trial, the plotter used the function to generate the data and write the x-y pairs to a CSV file. The Excel graph is essentially identical to the first graph in the smaller scale trial but scaled up substantially. The data was salted with a salt range of 0 to 100000 and saved to a new CSV file. The data points in the graph generated in Excel were dispersed much more during this trial but still retained a general upward slope. The smoother method was then run on the salted data with a window value of 25 and the results were saved to a new CSV file. The graph of the first set of smoothed data was noticeably closer to the original curve from the plotter function than the first smoothing run in the first trial. To remain consistent with the small-scale trial, the smoother method was run two more times on the data. The resulting graphs from the second and third were also similar to each other but were not much smoother than the first run since the first smoothing test was very effective. Similar to the first trial, the end of the curve seemed to flatten out but it was less pronounced in this trial with the larger data set and wider window size. Overall, the larger scale trial appeared to be more effective in the smoothing process and produced a curve that was very similar to the original.

Conclusions

The programming involved with this project was mostly straightforward and the only slightly tricky aspect was accounting for the lower and upper bounds of the data in the smoothing method to avoid any out of bounds errors. Since this program was written from scratch, there were some very valuable takeaways despite the existence of easier ways to achieve the same goals of the program. For example, the ability to read from and write to CSV files is especially useful in any profession or general application that involves raw data. The salting and smoothing concepts of this program are also particularly valuable in areas such as cryptography where one may need to scramble or unscramble data to obtain understandable information.