Plotting, Salting, and Smoothing using CSV files in Eclipse

CSV files allow for easy organization of data. I used csv files to plot, salt, and smooth an x,y function.

Plotting

For the plotting program, I used a simple y=mx+b function, in this case (1/2)x+9, created a method that would generate and gather all the pairs of x and y values and place them into an ArrayList. The pairs are then written into a csv file line by line. The method has parameters for the lower bound, upper bound, the amount that x will increment, and the number of points generated. In order to make it so that the loop would only iterate for a certain number of points, the executing conditions are as follows. “x <= upperBound and numPoints >=0” after each iteration, the variable for the number of points will be subtracted by the increment value, so if the user wanted the first 50 points, the loop would only iterate 50 times. Below are some results with some of the values of each parameter changed.

Low Bound: 0

Upper Bound: 50

Increment: 2

Number of Points: 50

Low Bound: 0

Upper Bound: 50

Increment: 3

Number of Points: 30

Salting

Figuring out how to salt y-values in a csv file in Java was somewhat challenging and convoluted. This works by generating a random number bound between a positive and negative parameter, then adding it to the y-values and rewriting the new data into a separate file. The parameters filename and saltRange provide the csv file that will be salted and the bounds for which to generate a random number. The salt method works by simply using a scanner to add each line from the input file, as a String of “x,y” into an ArrayList. Then, for each ArrayList entry, split the x and y value into a String[] array, add the random number to array[1] or the y-value, rewrite x,y as a String, return it back into the ArrayList. After all the y-values in the ArrayList were edited, write each entry as a line in the new csv file. Java.util.Random.nextInt() cannot recieve lower bound parameters, so the random number variable is defined as follows.

Double random = -saltRange + random.nextInt(saltRange \*2)

Below are some results with different salt ranges.

Salt Range: 100

Salt Range: 300

Salt Range: 1,000

Smoothing

Smoothing salted data using csv files was also challenging and convoluted. This method of smoothing uses a rolling average of the y-values adjacent to a specific index, within the window value. The smooth method begins similar to the salt method, where each line from the input file is written into an ArrayList. This time, for each index, I needed to get the other y-values within the window value and average them together. Within a nested while loop within a for loop, I set the upper limit to i+windowValue, then proceeded to gather each y-value until i-windowValue, putting each value into a separate ArrayList. After this, I found the average value of all the y-values, reinsert it back into the String of “x,y”, place the String back into the first ArrayList, then write each entry as a line in a new csv file. Below are some results with different window values, and smoothed multiple times.

Window Value: 3

Window Value: 6, First Smooth

Window Value: 6, Second Smooth

Window Value: 6, Third Smooth