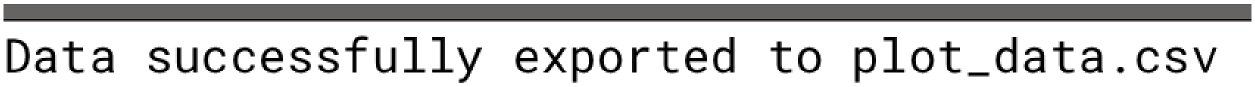
BlueJ:

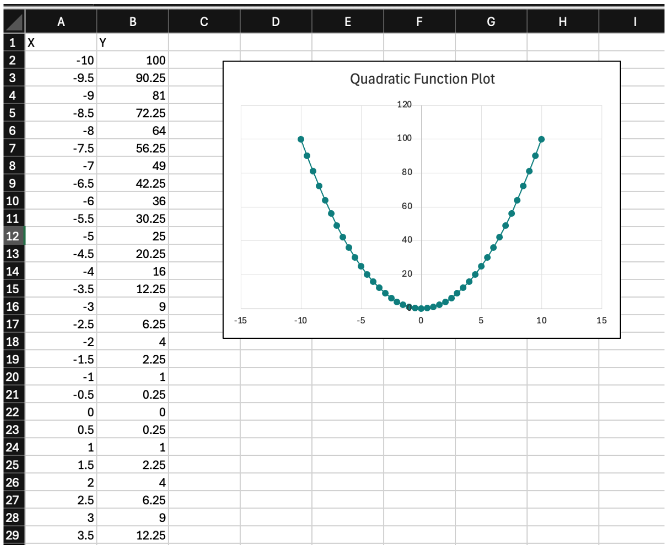
PlotFunction.java



```
Data successfully exported to plot_data.csv
```

- When running the program, this is the output prompted. In the folder that the .java file is in, creates a csv file after running the program.

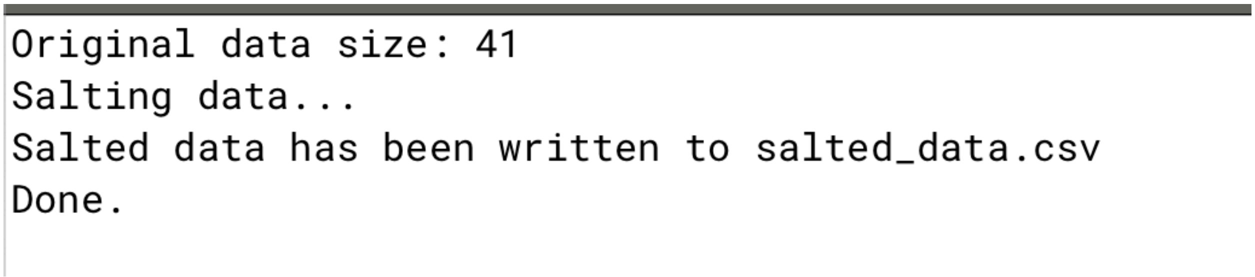
- From here you can view the .csv file that it created for you. I then proceeded to extract this csv file into excel.



	A	B
1	X	Y
2	-10	100
3	-9.5	90.25
4	-9	81
5	-8.5	72.25
6	-8	64
7	-7.5	56.25
8	-7	49
9	-6.5	42.25
10	-6	36
11	-5.5	30.25
12	-5	25
13	-4.5	20.25
14	-4	16
15	-3.5	12.25
16	-3	9
17	-2.5	6.25
18	-2	4
19	-1.5	2.25
20	-1	1
21	-0.5	0.25
22	0	0
23	0.5	0.25
24	1	1
25	1.5	2.25
26	2	4
27	2.5	6.25
28	3	9
29	3.5	12.25

- I was able to make a graph with the points provided in excel after extracting the csv into it.

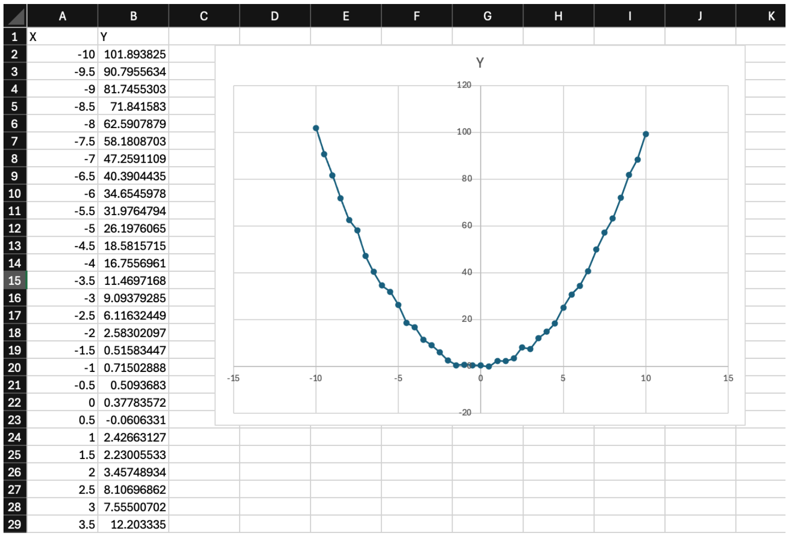
DataSalter.java



```
Original data size: 41
Salting data...
Salted data has been written to salted_data.csv
Done.
```

- When running the DataSalter program, this is the output prompted. In the folder that the .java file is in, creates a csv file after running the program.

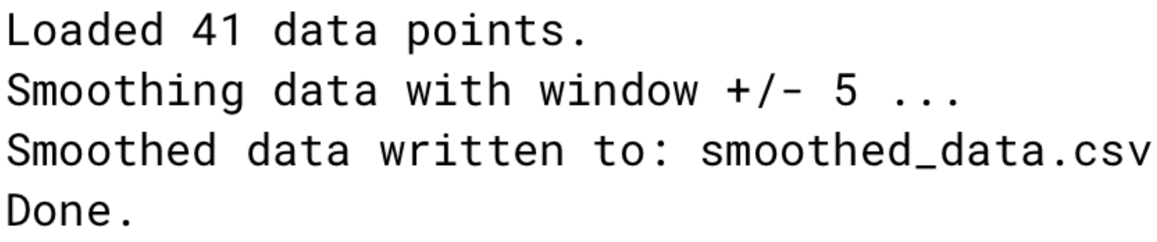
- From here you can view the .csv file that it created for you. I then proceeded to extract this csv file into excel.



	A	B
1	X	Y
2	-10	101.893825
3	-9.5	90.7955634
4	-9	81.7455303
5	-8.5	71.841583
6	-8	62.5907879
7	-7.5	58.1808703
8	-7	47.2591109
9	-6.5	40.3904435
10	-6	34.6545978
11	-5.5	31.9764794
12	-5	26.1976065
13	-4.5	18.5815715
14	-4	16.7556961
15	-3.5	11.4697168
16	-3	9.09379285
17	-2.5	6.11632449
18	-2	2.58302097
19	-1.5	0.51583447
20	-1	0.71502888
21	-0.5	0.5093683
22	0	0.37783572
23	0.5	-0.0606331
24	1	2.42663127
25	1.5	2.23005533
26	2	3.45748934
27	2.5	8.10696862
28	3	7.55500702
29	3.5	12.203335

- I was able to make a graph with the points provided in excel after extracting the csv into it.

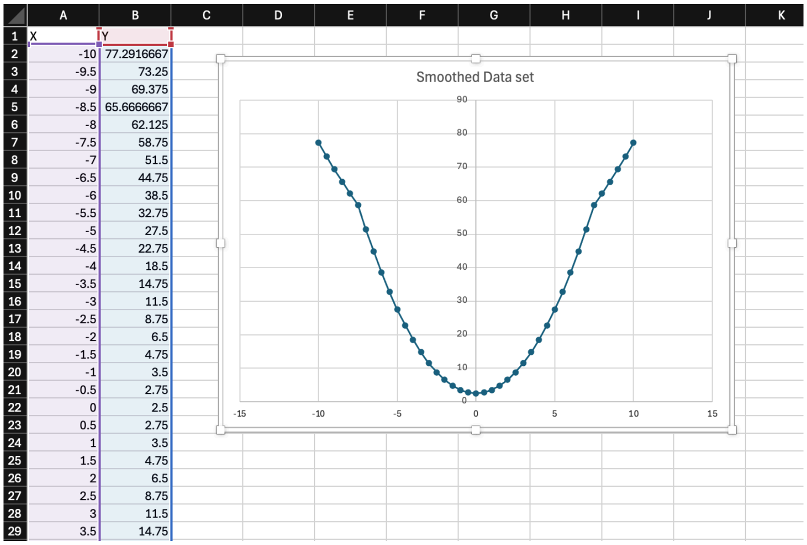
Smoother.java



```
Loaded 41 data points.
Smoothing data with window +/- 5 ...
Smoothed data written to: smoothed_data.csv
Done.
```

- When running the Smoother program, this is the output prompted. In the folder that the .java file is in, creates a csv file after running the program.

- From here you can view the .csv file that it created for you. I then proceeded to extract this csv file into excel.



X	Y
-10	77.2916667
-9.5	73.25
-9	69.375
-8.5	65.6666667
-8	62.125
-7.5	58.75
-7	51.5
-6.5	44.75
-6	38.5
-5.5	32.75
-5	27.5
-4.5	22.75
-4	18.5
-3.5	14.75
-3	11.5
-2.5	8.75
-2	6.5
-1.5	4.75
-1	3.5
-0.5	2.75
0	2.5
0.5	2.75
1	3.5
1.5	4.75
2	6.5
2.5	8.75
3	11.5
3.5	14.75

- I was able to make a graph with the points provided in excel after extracting the csv into it.

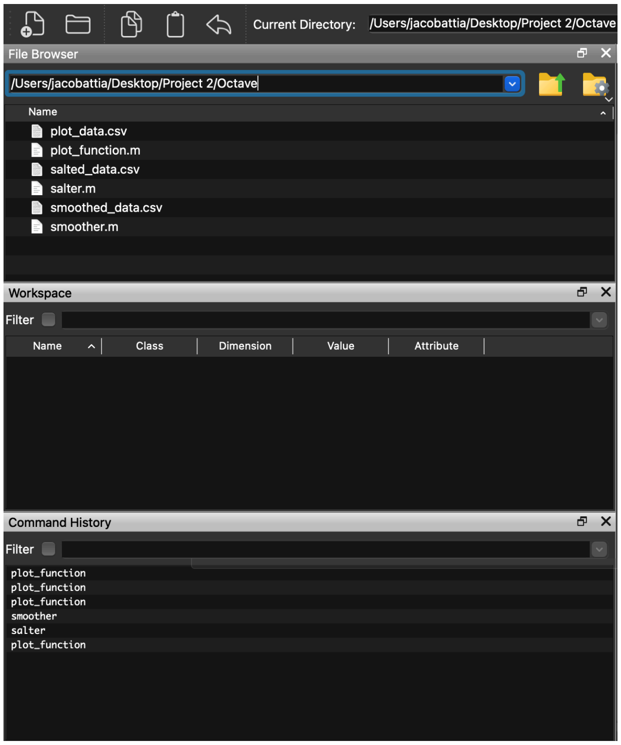
DataHandler.java

```
===== Generating original data (PlotFunction) =====
Data successfully exported to plot_data.csv
===== Salting data (DataSalter) =====
Original data size: 41
Salting data...
Salted data has been written to salted_data.csv
Done.
===== Smoothing data (Smoother) =====
Loaded 41 data points.
Smoothing data with window +/- 5 ...
Smoothed data written to: smoothed_data.csv
Done.

All steps complete!
You should now have plot_data.csv, salted_data.csv, and smoothed_data.csv.
```

- For the DataHandler class, it provides this output which creates all csv files for each of the programs with just one execution !

Octave



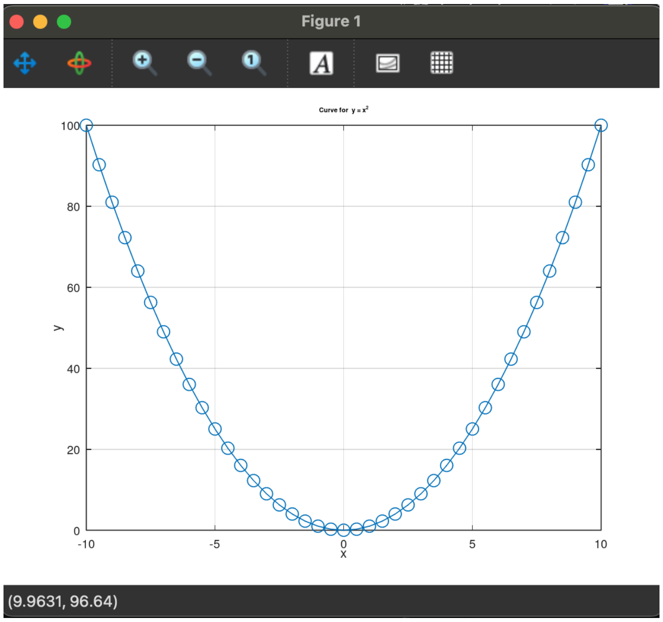
Current Directory: /Users/jacobattia/Desktop/Project 2/Octave

/Users/jacobattia/Desktop/Project 2/Octave

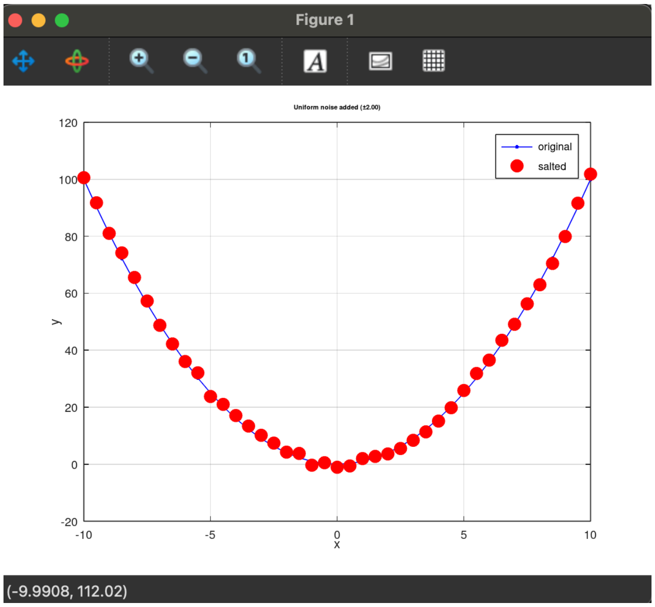
plot_data.csv
plot_function.m
salted_data.csv
salter.m
smoothed_data.csv
smoother.m

```
plot_function
plot_function
plot_function
smoother
salter
plot_function
```

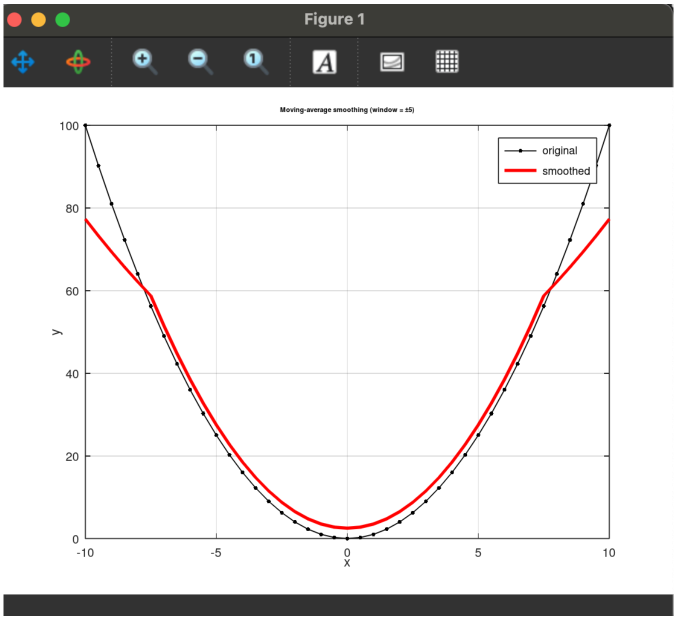
- This is the Octave GUI, once I ran the programs it then created a separate csv file for each.



- When running the plot_function.m program, it shows a visual graph and also produces a csv file.



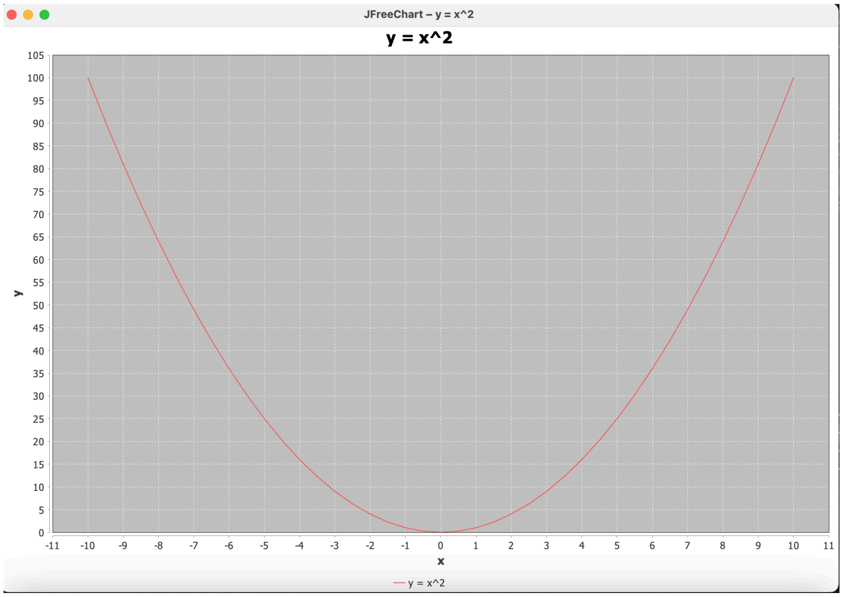
- When running the salter.m program, it shows a visual graph and also produces a csv file.



- When running the smoother.m program, it shows a visual graph and also produces a csv file.

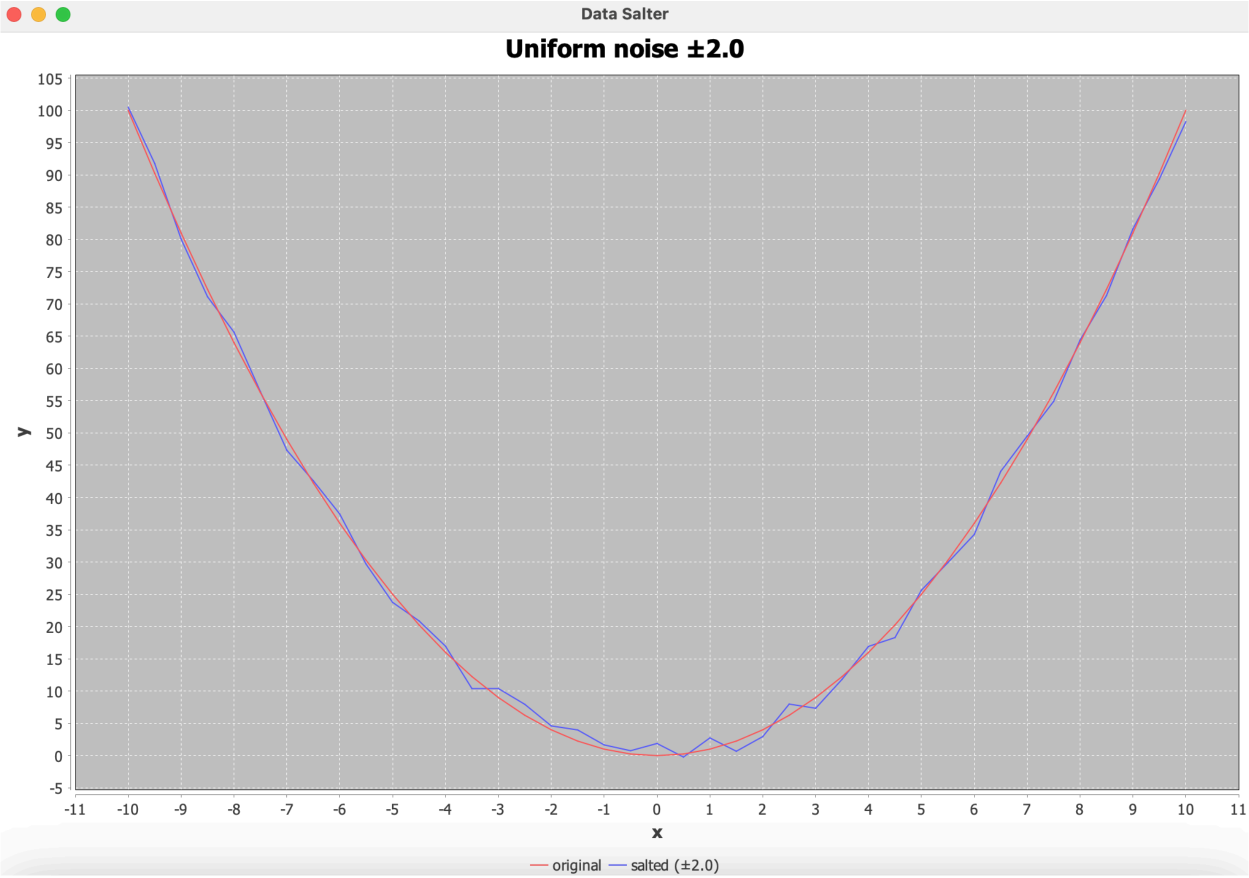
VS Code (JfreeChart + Apache stats library)

DataPlotter.java



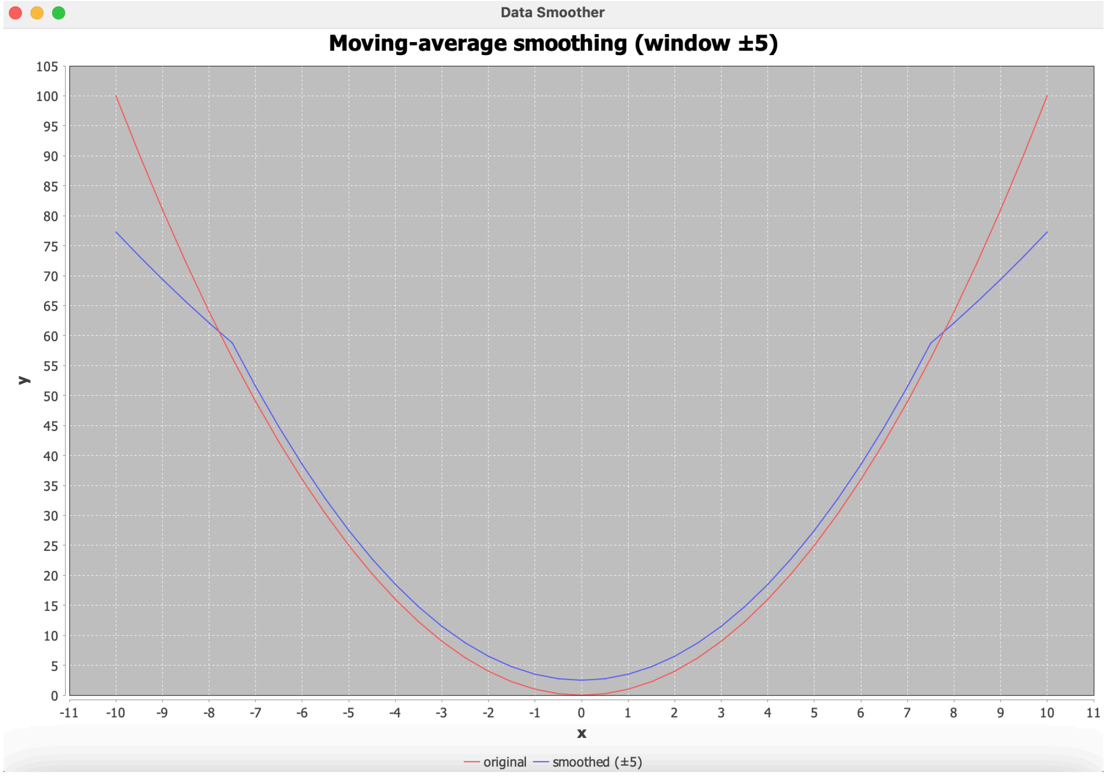
- VS code output with the Jfreechart plugin. When running the program it automatically opens this graph.

DataSalter.java



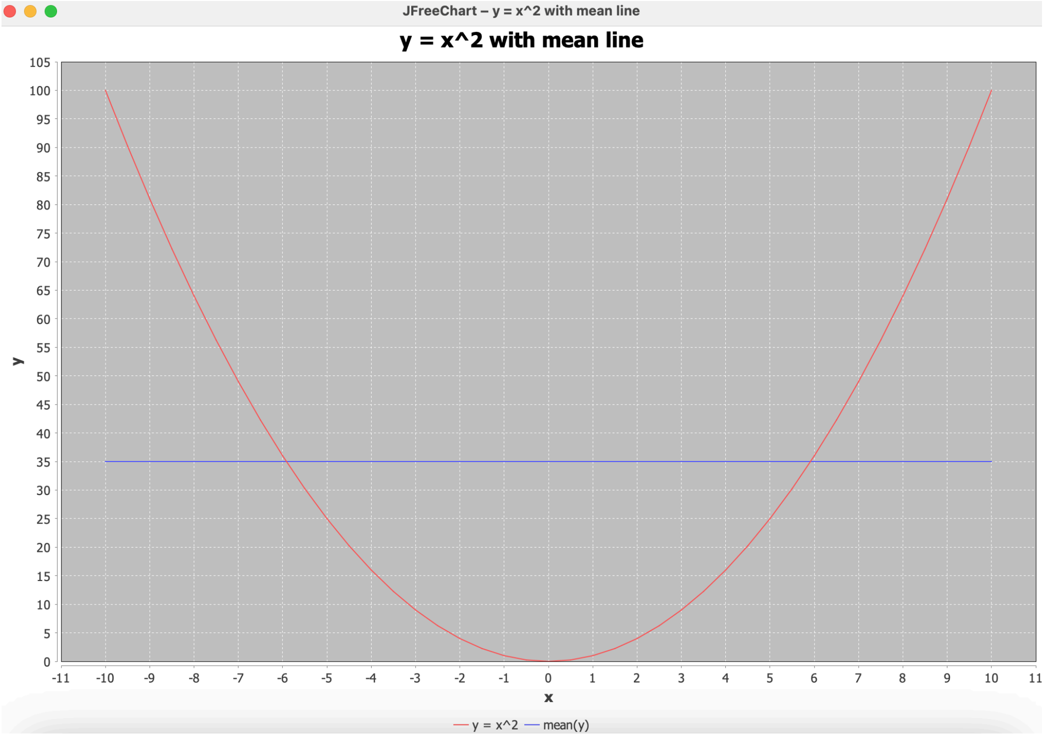
- VS code output with the Jfreechart plugin. When running the program it automatically opens this graph.

DataSmoother.java

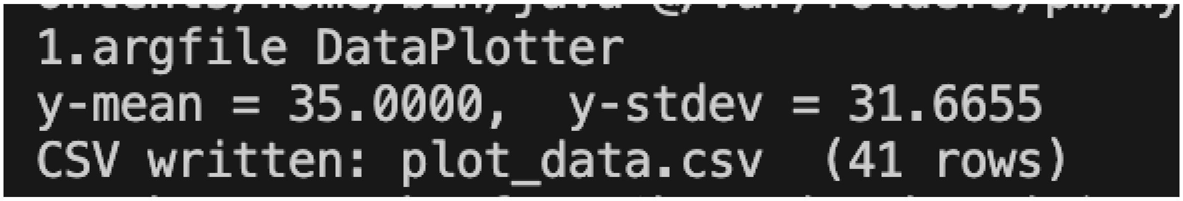


- VS code output with the Jfreechart plugin. When running the program it automatically opens this graph.

DataPlotter.java (Apache common math implementation)



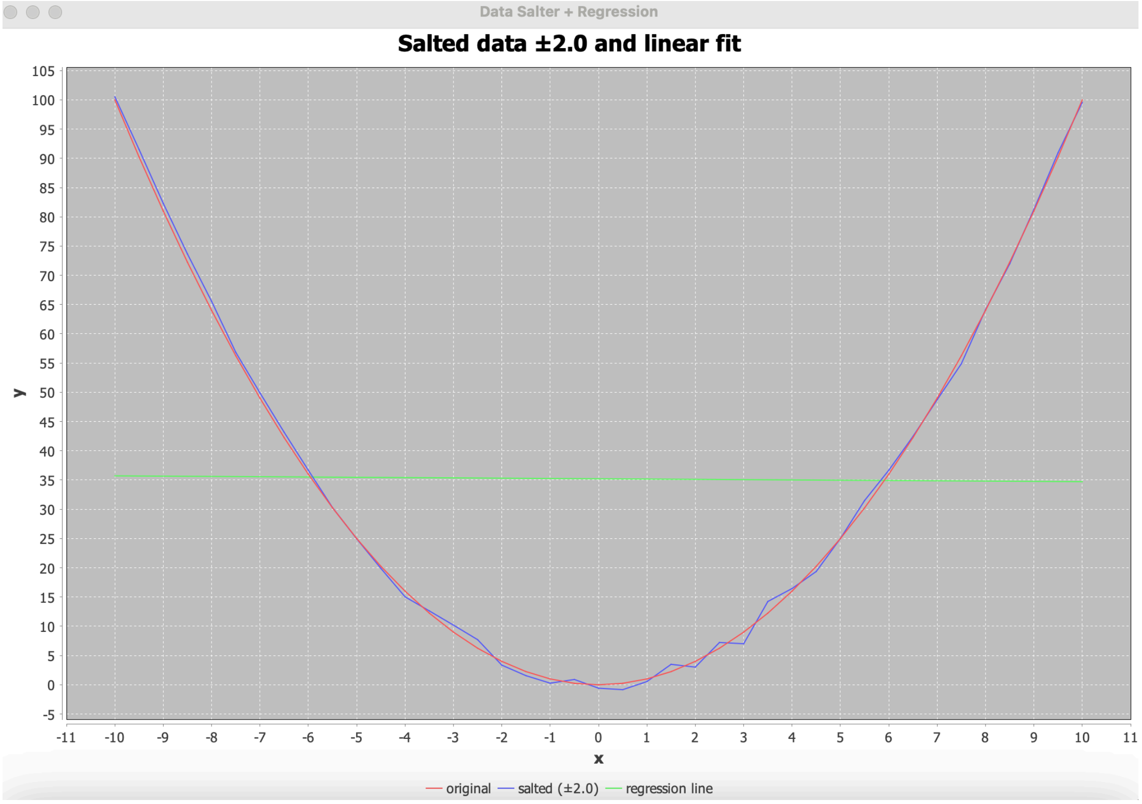
- When apache common math is implemented into the program, it shares a mean line and can show any regression lines when prompted as well. It works in unison with Jfreechart to show a more advanced graphical interface.



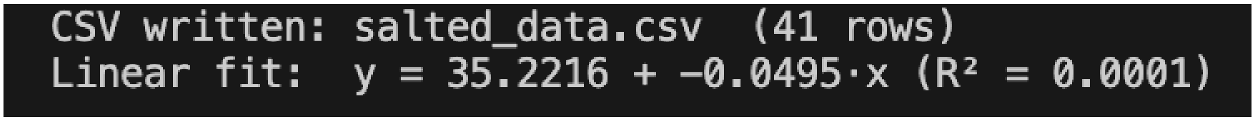
```
1.argfile DataPlotter
y-mean = 35.0000,   y-stdev = 31.6655
CSV written: plot_data.csv   (41 rows)
```

- Apache common math also shares the mean and standard deviation in the console, to these problems when prompted to do so.

DataSalter.java (Apache common math implementation)



- When adding the apache common math integration, a best-fit straight line through the noisy points demonstrating a practical use of Apache Commons Math.



```
CSV written: salted_data.csv  (41 rows)
Linear fit:  y = 35.2216 + −0.0495·x (R² = 0.0001)
```

- Results of the datasalter + apache common math integration in the console.