

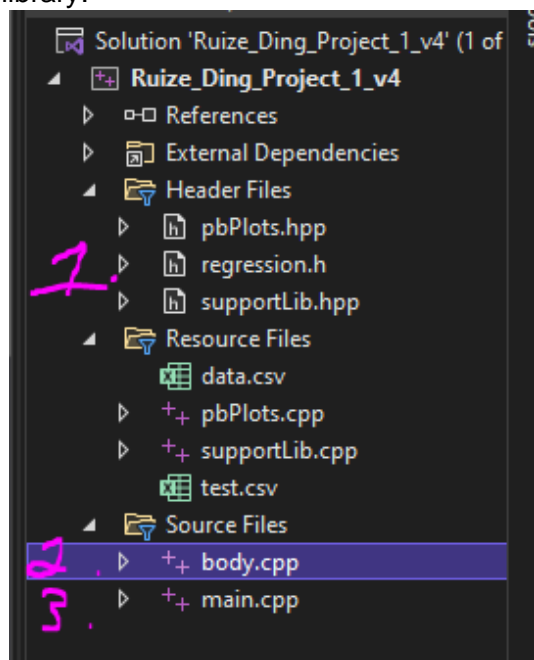
Project 1: Linear Regression

1. Abstract

- My linear regression program calculates the optimal linear equation to model a data set. After generating the values for the linear formula in $y=mx+b$ form, it can calculate predicted y values given a set of x values. Additionally, my program has functionality to plot the expected values in a scatter plot.
- There are two parts to my program. Part one takes the input as a pre-determined vector with values inside. Part two takes two .csv files. The data file has both x and y values and is used to generate the predicted linear regression model. The test file only has x values and is used to calculate the predicted y values.
- Plotting uses the lesser known pbplots library. Although, I have experimented with gnuplot but choose pbplot due to ease of implementation.

2. Introduction

- My code follows the three-file format of header, body and main. The screenshot below shows the files used in this program. In addition to the header, body and main files, there contains a various files containing input data and the pbplot library.



- The header is named "regression.h" according to protocol as it declares all objects in the regression class. Additionally, all the variable names are declared here.
 - The body paragraph contains all the code needed to run the various functions of the regression class.
 - Lastly, the main function just creates two objects and runs through the regression class.
- Below is a screenshot of the "regression.h" header. It shows the structure of the regression class and the variables and functions used. The breakdown of functions and the use of private and public will be discussed further in the main

results section of this report.

```
Ruize_Ding_Project_1_v4 (Global Scope)
1  #ifndef REFERENCE_H
2  #define REFERENCE_H
3
4  #include <vector>
5
6  class Regression {
7  private:
8      double b0, b1;
9      std::vector<double> ynew;
10 public:
11     std::vector<std::vector<double>> data;
12     std::vector<double> xnew;
13     int part;
14     void train();
15     void predict();
16     void cout();
17     void read_data();
18     void read_train();
19     void output();
20
21     int plot1();
22     int plot2();
23 };
24
25 #endif
```

- d. Due to the high complexity of the body file, it will not be discussed in detail in this portion of the report.
- e. The following is the main file of the code. It creates two objects in the regression class. These two objects represent part 1 and part 2 of the project. For each of these parts, it runs the various functions of the regression class depending on the data input style. If data input is from pre-existing vectors inside the main, then different functions are ran to output the results to console. If the data is inputted through data files then it writes the output to a .txt file called "output.txt".

```
Ruize_Ding_Project_1_v4 (Global Scope) main()
1  #include "regression.h"
2
3  int main() {
4      Regression part1;
5      part1.data = { {5, 43.1}, {7.1, 32.1}, {34.5, 40.3}, {13, 39.3}, {1.5, 47.7} };
6      part1.xnew = { 1, 2, 3, 4, 5 };
7      part1.train();
8      part1.predict();
9      part1.cout();
10     part1.plot1();
11
12     Regression part2;
13     part2.read_data();
14     part2.train();
15     part2.read_train();
16     part2.predict();
17     part2.output();
18     part2.plot2();
19 }
```

- f. This concludes the high-level implementation of my code. The next section will dive in deep and provide detailed explanations of the classes and functions.

3. Main Results

- a. This following section will describe the private/public structure of the regression class.
 - i. Private inside a class means that variables and functions can only be accessed within that specific class and cannot be called outside. My private variables are the b0 and b1 variables in addition to ynew. These variables are private because they are calculated using the train and predict functions and only ever used within the regression class.
 - ii. Public inside a class means that variables and functions can be accessed outside the specific class such as in main. In this case, the vectors data and xnew are sent in. This is because in part 1 of the program, they are sent in from the main function. Although if the program only existed of part 2, they could be set as private as they are generated from the read_data() and read_train() functions and have no interaction outside the class. The various functions are all called in main and therefore need to be declared in public. If functions called other functions and my program had a higher level of complexity, then only the input functions would need to be called and the functions will automatically call functions as needed.
- b. This following section shows a labeled version of the regression.h header and will break down in detail the various functions and variables used in the class.

```

1  #ifndef REFERENCE_H
2  #define REFERENCE_H
3
4  #include <vector>
5
6  class Regression {
7  private:
8      double b0, b1;
9      std::vector<double> ynew;
10 public:
11     std::vector<std::vector<double>> data;
12     std::vector<double> xnew;
13     void train();
14     void predict();
15     void cout();
16     void read_data();
17     void read_train();
18     void output();
19
20     int plot1();
21     int plot2();
22 };
23
24 #endif
  
```

- i. My numbering in magenta shows which part of the project each of the functions responds to. A number 1 refers to part one and a number 2 refers to part two.
- ii. The first private calls in two variables—b0 and b1. B0 and b1 represent the b and m values of $y=mx+b$ respectively. This is demonstrated in the following equation:

To calculate the best β_0 , which is denoted by $\hat{\beta}_0$, we use

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X} \quad (3)$$

- iii. The ynew vector is a one dimensional vector that contains all the newly calculated y-values based off the regression formula.
- iv. In public part, the first variable is called is the data vector. It is a two dimensional vector that contains the x and y values of the data used to calculate the linear regression equation. Because it is holding multiple x and y values in the $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\} \dots\}$ format, it needs to be two deminsional.
- v. The next vector is a one dimensional vector that contains the pre-determined x values that will be used to calculated the ynew values.
- vi. The next function void train(); “trains” the data values and calculates the b0 and b1 values. This is done through calculating the total x and total y values, taking the average and then subtracting that average value from each x and y value respectively. Essentially, it calculates the following formula:

To calculate the best β_1 , which is denoted by $\hat{\beta}_1$, we use

$$\hat{\beta}_1 = \frac{\sum_{i=1}^N (x_i - \bar{X})(y_i - \bar{Y})}{\sum_{i=1}^N (x_i - \bar{X})^2} \quad (1)$$

where

$$\bar{X} = \text{avg}(X) = \frac{\sum_{i=1}^N x_i}{N}, \text{ and } \bar{Y} = \text{avg}(Y) = \frac{\sum_{i=1}^N y_i}{N} \quad (2)$$

- vii. Next up is the void predict (); function. This function takes the calculated b0 and b1 values from the train function and calculates the predicted y values based off the given x values.
- viii. The next void cout(); function is labeled “1” because it is only used in part one of the project. This function outputs the results into console in the folowing format:

```

C:\ Microsoft Visual Studio Debug Console
B1 (slope): -0.0845179
B0 (y-intercept): 41.5328
Regression formula: y = 41.5328 + -0.0845179x
(1,41.4483)
(2,41.3638)
(3,41.2793)
(4,41.1947)
(5,41.1102)

```

- ix. Next up is the void read_data(); function. This function takes the “data.csv” file and reads it into the data 2-d vector. This function is only used in part 2 because that is the portion of the project where data is fed in through an external .csv file.
- x. The other read file is the void read_train(); function. Here, it reads the x value from “test.csv” file. This is again inputted into the xnew vector. This function is also only used in part 2 because that is where data is read through a .csv file.
- xi. The output(); function writes to the output.txt file the same output shown in console.

- xii. The last two functions are both plotting functions. The only difference is that they rename the saved png file as different names for part 1 and part 2.
- c. This next section will break down the body section of the code. This is where all the code to the functions is stored.

```

body.cpp  X  main.cpp  regression.h
Ruize_Ding_Project_1_v4
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <sstream>
5  #include <vector>
6
7  #include "pbPlots.hpp"
8  #include "supportLib.hpp"
9  #include "regression.h"

```

- i.
- ii. These are the defined files needed to run my program. pbPlots.hpp, supportLib.hpp are both pbplot libraries. "regression.h" is my header file.

```

11 void Regression::train() {
12     double xt看 = 0, yt看 = 0, xbar, ybar, numerator = 0, denominator = 0;
13     for (int i = 0; i < data.size(); i++) {
14         xt看 += data[i][0];
15         yt看 += data[i][1];
16     }
17     xbar = xt看 / data.size();
18     ybar = yt看 / data.size();
19     for (int i = 0; i < data.size(); i++) {
20         numerator += (data[i][0] - xbar) * (data[i][1] - ybar);
21         denominator += pow((data[i][0] - xbar), 2);
22     }
23     b1 = numerator / denominator;
24     b0 = ybar - b1 * xbar;
25 }
26
27 void Regression::predict() {
28     for (int i = 0; i < xnew.size(); i++) {
29         ynew.push_back(b0 + b1 * xnew[i]);
30     }
31 }

```

- d.
 - i. This is the train function. The variables xt看 and yt看 represent the sum of all the x values and y values respectively. Xbar and ybar represent the mean of the x and y values respectively. Numerator and denominator variables are used to calculate b1 (slope) of the regression formula.
 1. Solving for the numerator is done through taking each individual x value and subtracting it by the x mean. Then multiplying it by taking each individual y value and subtracting it by the y mean.
 2. The denominator follows a similar process and is done through squaring each individual x value minus the mean.
 3. Dividing the numerator by the denominator produces the end result of b1.

```

27 void Regression::predict() {
28     for (int i = 0; i < xnew.size(); i++) {
29         ynew.push_back(b0 + b1 * xnew[i]);
30     }
31 }

```

e.

- i. This is the predict function calculates the new ynew value. It uses a for loop the size of the xnew vector and for each xnew vector uses the regression formula to calculate the ynew value. These ynew vectors are put into the ynew vector.

```

33 void Regression::cout() {
34     std::cout << "B1 (slope): " << b1 << "\n";
35     std::cout << "B0 (y-intercept): " << b0 << "\n";
36     std::cout << "Regression formula: y = " << b0 << " + " << b1 << "x \n";
37     for (int i = 0; i < xnew.size(); i++) {
38         std::cout << "(" << xnew[i] << ", " << ynew[i] << ") \n";
39     }
40 }

```

f.

- i. The cout function produces the output into the console for part 1. It outputs the b1, b0 values in addition to the regression formula. After this, it cycles through the xnew vector and prints out the x and y values in the (x,y) format.

```

42 void Regression::read_data() {
43     std::ifstream read("data.csv");
44     if (read.is_open()) {
45         std::string line;
46         std::vector<double> tempVector;
47         while (getline(read, line)) {
48             double x, y;
49             std::string tempString;
50             std::stringstream inputString(line);
51             getline(inputString, tempString, ',');
52             x = std::stod(tempString.c_str());
53             getline(inputString, tempString);
54             y = std::stod(tempString.c_str());
55             tempVector.push_back(x);
56             tempVector.push_back(y);
57             data.push_back(tempVector);
58             tempVector.clear();
59         }
60     }
61     read.close();
62 }

```

g.

- i. This next function is the read_data function. It reads through the data.csv file used for part 2. First it uses ifstream to open the file, an if statement checks to see if the read is open. This is an error checking mechanism and makes sure the function will not run unless the read function is indeed working as intended.

- ii. Two variables are created inside this if statement. The string named line will be used to read each individual line of the data.csv file. The tempVector is necessary for storing the 1d array before it is formatted into a 2d array.
- iii. The next part of the code is a while(getline()) loop. This while loop will run through the code until there is no longer any more code to run through. A string called tempString is created to assist in converting a string into a double. A double x and y are created to be used for each of the x and y values. However, in order to get the data in the form of x and y doubles, it first must be read in as a string and converted.
 - 1. The process of reading in a value is as follows. First, getline is used to read a value until the “,”. This is the x value. Next, we convert that tempString into a double using the stod command. The stod command is a higher level version of the atoi command, which works on chars.
 - 2. The process is now running to find y. Where we now read from the “,” to the end of the line. The same process of using a temporary string and then converting it into a double is also used.
- iv. After finding the x and y values, they are stored into the tempVector which is a 1d vector. This temp vector is then pushed into the data vector and then cleared. This allows it to be reused for the next x and y values down the line and so on.

```

64 void Regression::read_train() {
65     std::ifstream read("test.csv");
66     if (read.is_open()) {
67         std::string line;
68         while (getline(read, line)) {
69             double x;
70             x = std::stod(line.c_str());
71             xnew.push_back(x);
72         }
73     }
74     read.close();
75 }

```

h.

- i. The read_train() function has a similar purpose as the read_data function, but instead it reads in the test.csv file. This function also uses the same if loop and while(getline()) structure. However, because it is only reading in a single x value at a time, there is no need for a temp vector. Values are converted into a string and then stored into the xnew array.


```

77 void Regression::output() {
78     std::ofstream output("Output.txt");
79     if (output.is_open()) {
80         output << "B1 (slope): " << b1 << "\n";
81         output << "B0 (y-intercept): " << b0 << "\n";
82         output << "Regression formula: y = " << b0 << " + " << b1 << "x \n";
83         for (int i = 0; i < xnew.size(); i++) {
84             output << "(" << xnew[i] << ", " << ynew[i] << ") \n";
85         }
86     }
87     else std::cout << "Output file does not open";
88     output.close();
89 }

```

i.

- i. This output() function has a similar function as the cout() function. However, here it is outputting the results into a output.txt file. There is error checking in this function where if the file is not created a message informing the user there was an error will present itself.

```

91 int Regression::plot1() {
92     bool success;
93     StringReference* errorMessage = new StringReference();
94     RGBABitmapImageReference* imageReference = CreateRGBABitmapImageReference();
95
96     success = DrawScatterPlot(imageReference, 600, 400, &xnew, &ynew, errorMessage);
97
98     if (success) {
99         std::vector<double>* pngdata = ConvertToPNG(imageReference->image);
100         WriteToFile(pngdata, "part_1.png");
101         DeleteImage(imageReference->image);
102     }
103     else {
104         std::cerr << "Error: ";
105         for (wchar_t c : *errorMessage->string) {
106             std::wcerr << c;
107         }
108         std::cerr << std::endl;
109     }
110
111     return success ? 0 : 1;
112 }
113
114 int Regression::plot2() {
115     bool success;
116     StringReference* errorMessage = new StringReference();
117     RGBABitmapImageReference* imageReference = CreateRGBABitmapImageReference();
118
119     success = DrawScatterPlot(imageReference, 600, 400, &xnew, &ynew, errorMessage);
120
121     if (success) {
122         std::vector<double>* pngdata = ConvertToPNG(imageReference->image);
123         WriteToFile(pngdata, "part_2.png");
124         DeleteImage(imageReference->image);
125     }
126     else {
127         std::cerr << "Error: ";
128         for (wchar_t c : *errorMessage->string) {
129             std::wcerr << c;
130         }
131         std::cerr << std::endl;
132     }
133
134     return success ? 0 : 1;
135 }

```

j.

- i. These two functions are the plot functions. They plot the xnew and ynew values. The only difference in these functions is that they name the files different names. Unfortunately, I was unable to figure out how to properly pass a value into the function to change the naming convention. This should be done with just one plotting function for higher efficiency.
- k. Lastly, I will describe the main function and its layout/function.

```

Ruize_Ding_Project_1_v4 (Global Scope) main()
1  #include "regression.h"
2
3  int main() {
4      Regression part1;
5      part1.data = { {5, 43.1}, {7.1, 32.1}, {34.5, 40.3}, {13, 39.3}, {1.5, 47.7} };
6      part1.xnew = { 1, 2, 3, 4, 5 };
7      part1.train();
8      part1.predict();
9      part1.cout();
10     part1.plot1();
11
12     Regression part2;
13     part2.read_data();
14     part2.train();
15     part2.read_train();
16     part2.predict();
17     part2.output();
18     part2.plot2();
19 }

```

- i. The main function is very basic and exists to create two objects using the regression class. The object called part1 represents the first part of the project and has the pre-determined input of data and xnew.
- ii. The second object is called part2 and also includes the read file functions to get its input data.

4. Numerical Results

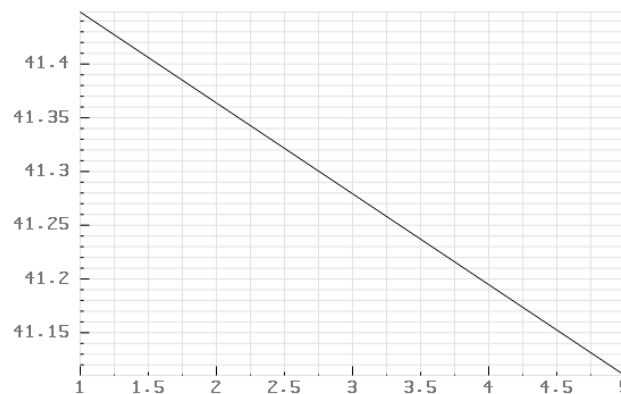
- a. The following is the output of part1 of the project.

```

Microsoft Visual Studio Debug Console
B1 (slope): -0.0845179
B0 (y-intercept): 41.5328
Regression formula: y = 41.5328 + -0.0845179x
(1,41.4483)
(2,41.3638)
(3,41.2793)
(4,41.1947)
(5,41.1102)

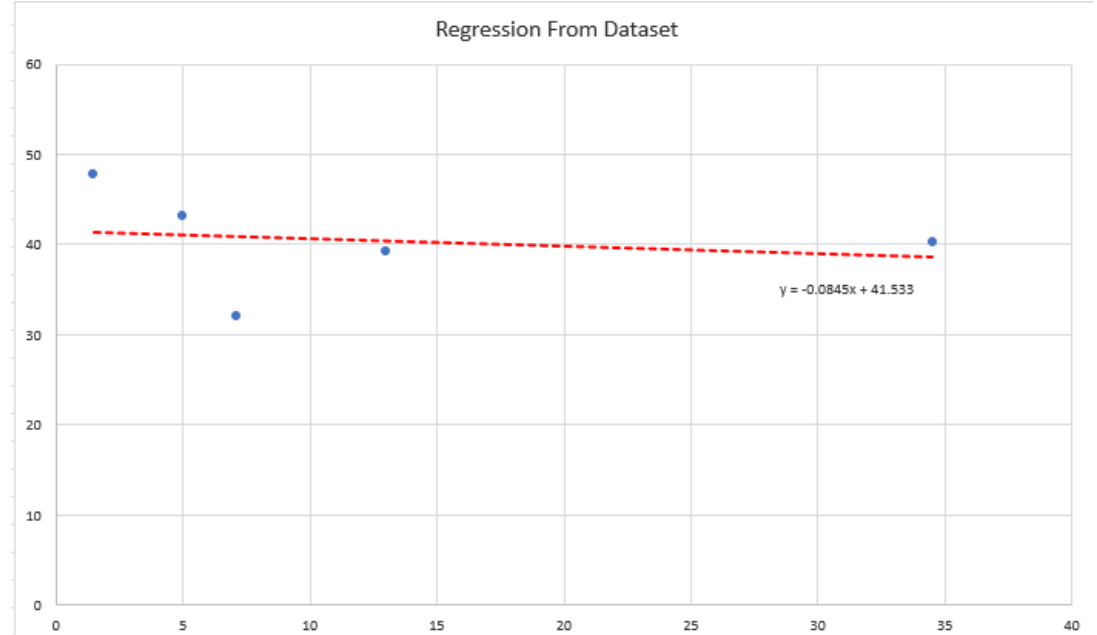
```

i.



ii.

- iii. Below is the results ran through an excel spreadsheet. The numbers match and are consistent.



iv.

X	Y
Dataset	
5	43.1
7.1	32.1
34.5	40.3
13	39.3
1.5	47.7

v.

- b. The following is the output of part2 of the project. As shown here, the output is done through a text file.

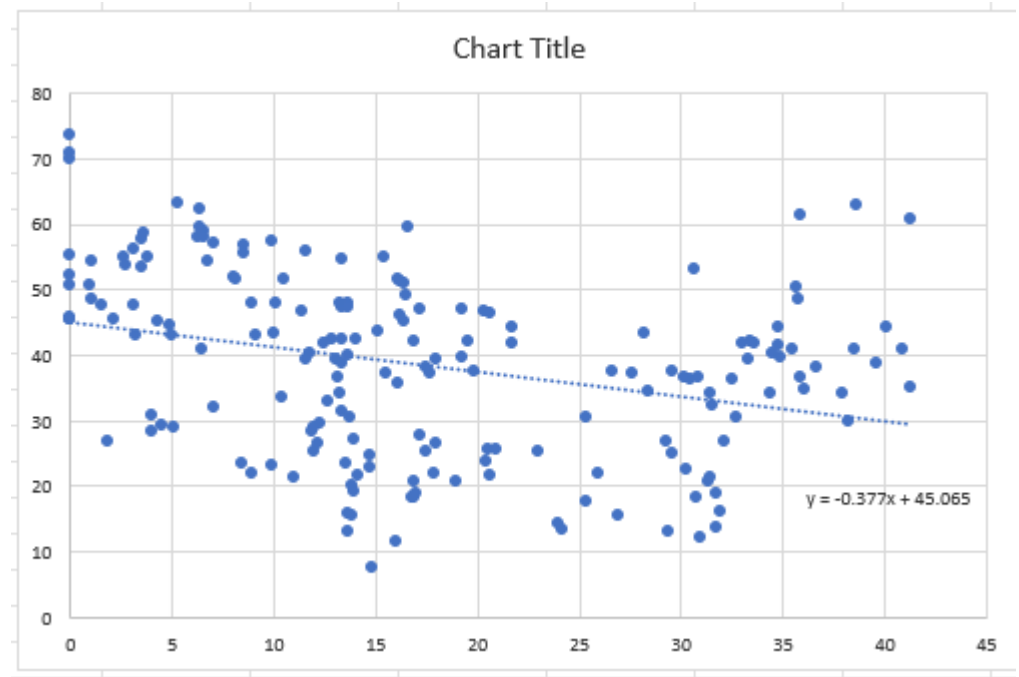
```
Output.txt - Notepad
File Edit Format View Help
B1 (slope): -0.376957
B0 (y-intercept): 45.0653
Regression formula:  $y = 45.0653 + -0.376957x$ 
(3.9,43.5952)
(37.3,31.0048)
(0,45.0653)
(14.1,39.7502)
(8,42.0497)
(16.3,38.9209)
(29.1,34.0959)
(16.1,38.9963)
(18.3,38.167)
(0,45.0653)
(16.2,38.9586)
(10.4,41.145)
(40.9,29.6478)
(32.8,32.7011)
(6.2,42.7282)
(42.7,28.9693)
(16.9,38.6948)
(32.6,32.7765)
(21.2,37.0738)
(37.1,31.0802)
(13.1,40.1272)
(14.7,39.5241)
(12.7,40.278)
(26.8,34.9629)
(7.6,42.2005)
(12.7,40.278)
(30.9,33.4174)
(16.4,38.8832)
(23,36.3953)
(1.9,44.3491)
(5.2,43.1051)
(18.5,38.0916)
(13.7,39.901)
(5.6,42.9544)
(18.8,37.9785)
(8.1,42.012)
(6.5,42.6151)
```

i.



ii.

- iii. I also calculated the regression in excel to guarantee the data I am outputting is correct. Below is the excel output. All numbers match and are correct.



iv.

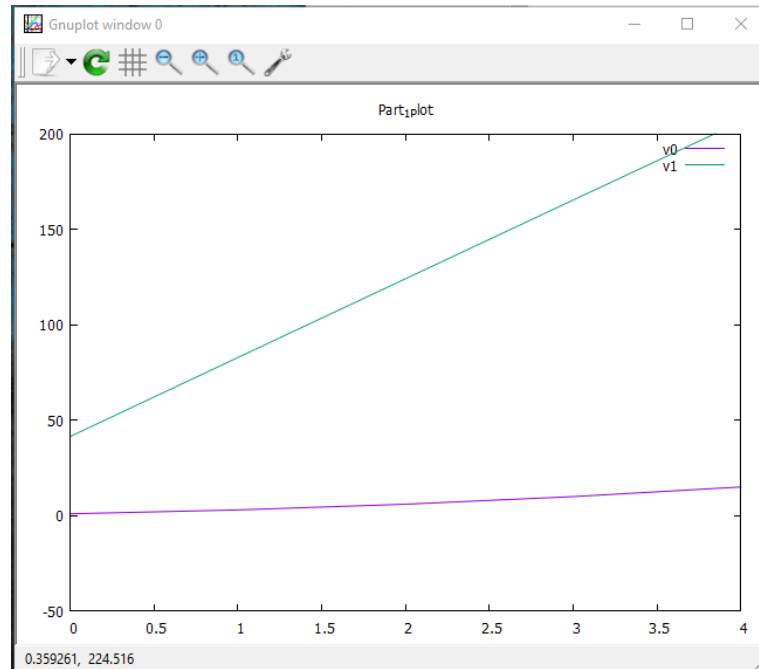
R	S	T
Num	Val	Result
1	3.9	43.5947
2	37.3	31.0029
3	0	45.065
4	14.1	39.7493
5	8	42.049
6	16.3	38.9199
7	29.1	34.0943
8	16.1	38.9953
9	18.3	38.1659
10	0	45.065
11	16.2	38.9576
12	10.4	41.1442
13	40.9	29.6457
14	32.8	32.6994
15	6.2	42.7276
16	42.7	28.9671
17	16.9	38.6937
18	32.6	32.7748
19	21.2	37.0726
20	37.1	31.0783
21	13.1	40.1263
22	14.7	39.5231
23	12.7	40.2771
24	26.8	34.9614
25	7.6	42.1998
26	12.7	40.2771
27	30.9	33.4157
28	16.4	38.8822
29	23	36.394
30	1.9	44.3487
31	5.2	43.1046
32	18.5	38.0905
33	13.7	39.9001
34	5.6	42.9538
35	18.8	37.9774
36	8.1	42.0113
37	6.5	42.6145

v.

5. Reflections

- a. This second iteration of the program offer significant improvements from the first version. These improvements are listed below:
 - i. The program nolonger has the previous data load error and now the entirely of part 2 is displayed correctly.
 - ii. The train and predict functions are entirely unchanged as mentioned in the project directions.
 - iii. This program makes use of classes, allowing for better organization.



- iv. Lastly, this program has a plot function and can graph the predicted plot.
- b. My program is very well organized and very efficient. There is minimal repeated code. Looking back at it, the major changes I would make would be to 1) use gnuplot or matplotlib. I did not know matplotlib was allowed as a library on this project because the project description says no matlab. This would have been the easiest implementation solution and resulted in a much better-looking graph. Gnuplot is also a good solution, although the implementation is a bit more complex.



- i.
- ii. Above is a screenshot of a gnuplot plot. I attempted to graph the part 1 data, but ran into problems due to the difficult implementation of gnuplot.
- iii. Gnuplot was very difficult to install, requiring the boost library and the vcpkg. Additionally, the current version of gnuplot I was using only works in c++ 17, limiting compatibility.
- c. Another change is I would have only used a single plotting function instead of two. I was unable to figure out how-to pass-through variables into the function to change the naming convention and had to hard code two functions into the program.
- d. In addition to pbplot and gnuplot, I learned how to use classes, passing by reference/pointer and reading files.
- e. More importantly however, I learned a more effective method of problem solving in programming. This strategy has me isolating the issue to the bare minimum amount of code and attempting the trouble shooting there in a separate program. Another method I used was to rewrite code to attempt to create the error, when I was unsure where the exact problem was. This problem-solving method I learned my first two days of working on this new project. My .csv files were corrupted and not in the correct format, causing them not to be read in the program. I kept on thinking it was a code issue until I created a new program with the same code and different files, and I realized it was a .csv file issue.

- f. Other skills I learned were what online resources to use. I learned how to properly ask questions on StackOverflow and how to use github to find libraries. I also found out quickly just how rude some people on StackOverflow can be and why it's important to attempt independent problem solving first.

Rude much?

1  "I am new to C++" so beware of the keyword `new` - MatG 22 hours ago 

6. Conclusions

- a. In conclusion, I was able to complete the project and follow all guidelines. In addition to just calculating the regression formulas, my program is also able to plot the regression line using the test plots. I was also able to learn many new skills during this project.