Aalto University

CS-C2120 - Ohjelmointistudio 2

Fitting and Visualization of a Regression Model: Project document

Name:	Juho Ylä-Jääski
StudentNro:	590963
Degree program:	Computer Science
Year of studies:	Freshman
Date:	April 16, 2020

Contents

1 Introduction

This project was completed as a part of the course *CS-C2120 - Ohjelmointistudio 2*, and the project concentrated on the visualization of three types of regression models: linear, exponential, and logarithmic. Regression implies the relationship between different variables. If two variables are directly proportional to each other, their relationship can be visualized with linear regression models.

The primary motivation of the project was to create a program where the user of the program can visualize the trends of different data-sets. The user first imports data-sets, consisting of data-points, into the program. These data-sets are two-dimensional, meaning that each of the data-points contains two variables (x, y). The user can then visualize these data-sets with a plotting-tool that enables the user to view the data-points with the regression curve. The plotting-tool also provides some additional features to modify the plot.

The program is easy to use, and plenty of testing was performed to ensure that it functions properly. The program allows the user to analyze multiple data-sets together with the regression models. The structure of the program is a bit messy at times, and the quality of the code is not perfect. However, the program is built in such a manner that ensures modifiability and extensibility of the program. Usability and functionality are the most robust features of the program.

This document is divided into five different sections. The general description and the plan of the project are illustrated in section 2. The program is presented in section 3. This section concentrates on the program structure, data structures, user interface, and files. The most important algorithms are described in section 4. Testing is designated for section 5 and the final evaluation of the project is presented in section 6. Further information, such as the test files provided and instruction for an example plot, are presented in the appendixes A and B, respectively.

2 General description

This section describes the plan and the overview of the final program. The general idea was to complete the advanced version of the given assignment, and the requirements for this are met with the final application. The most significant difference between the plan and the final program is that the user interface was only meant to have one window for all of the operations. The plan is presented in section 2.1. The overview of the final program is described in section 2.2.

2.1 Plan

The general plan of the project was to create a program that enables the user to visualize the trends of the data-sets with regression models. The plan was to implement two different regression models: linear and exponential. The data-set has a linear trend and thus can be modeled with linear regression when the data closely obeys the relationship given in equation 1:

$$y = kx + b, \tag{1}$$

where y together with x denote the variables and k along with b are some constants that characterize the data-set. The other regression model, exponential regression, was in the general plan because of the advanced version requirements. When the data-set describes the exponential relationship, it can be expressed with equation 2:

$$y = \alpha e^{\beta x}, \tag{2}$$

where α and β denote some constants. The linear regression model can be utilized for the exponential regression by taking a natural logarithm from both sides of the equation 2. This was the plan for the regression models, but the user must import the data-sets before utilizing the models.

The advanced version requires to implement two different file supports for data reading. The original plan covered this with both text and excel files. The idea was also to provide the user the opportunity to include some additional information with the use of blocks, such as #Info and #Endinfo.

The general plan for the program structure was to create separate classes for plotting, importing data, calculating regression models, and user interface. This structure is presented in the UML-diagram in figure 1.

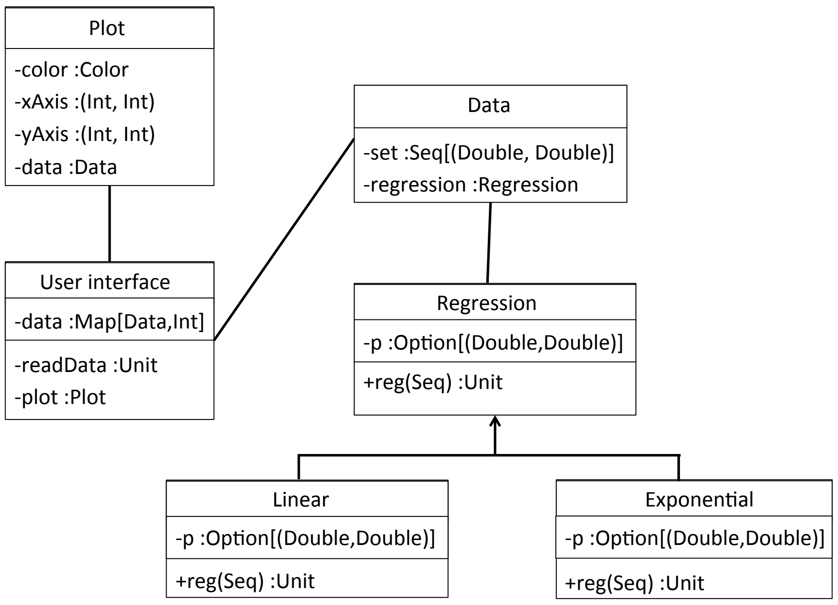


Figure 1: This UML-diagram represents the plan of the program structure.

The visual aspect of the program was also considered during the plan. The idea was to include all of the operations, importing, plotting, and modifying, in the same window. The visual plan is presented in figure 2.

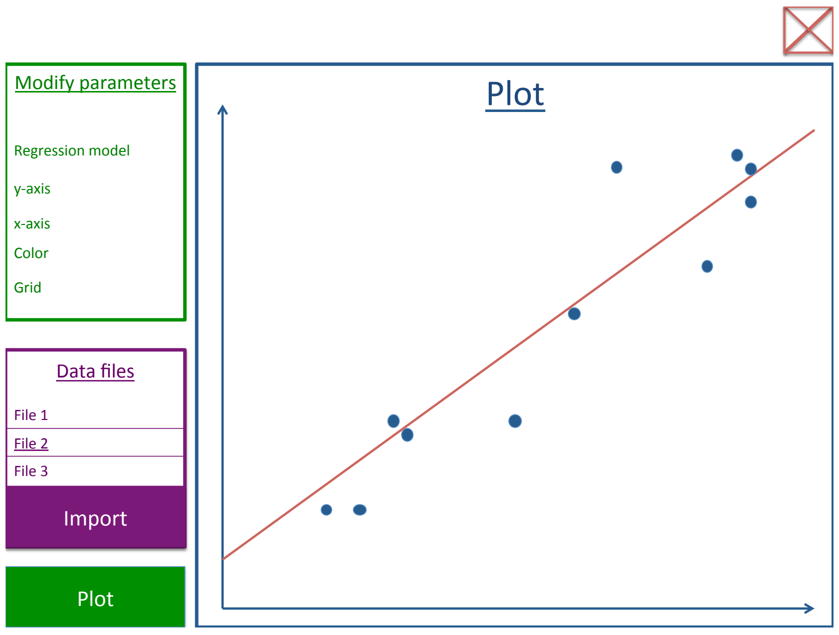


Figure 2: The visual plan for the final program only has one window for all of the operations.

2.2 The final program

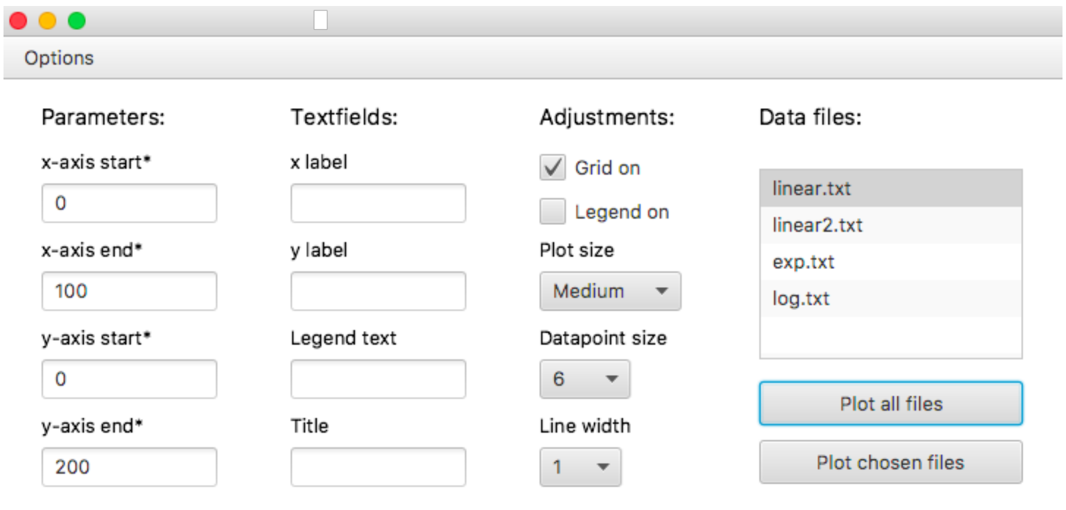
The final program is similar to the project plan version. The advanced version of the assignment asked to implement two different regression models, and at least two different files had to be supported for reading data. In addition to the linear and exponential models presented in the equations 1 and 2 respectively, logarithmic model was also included which is expressed in equation 3:

$$y = a + c \ln x, \tag{3}$$

where a and c denote some constants. The final program also supports three types of data files: txt, xlsx, and csv; thus, the requirements for the advanced version are met.

The structure of the final program closely reassembles the architecture presented in figure 1. Some additional classes, CoordinateSystem and Legend, were added to make the program more advanced. These classes enable the user to visualize the data with a coordinate system, grid, and legend.

The final program was built more advanced than the requirements in other aspects as well. The reason for this was to make the program and the project as useful as possible in the future. I have already used my program to visualize data and regression models because it is much easier and faster than with MATLAB. In addition to the coordinate system, grid and legend, different data-sets may be plotted at the same time for comparison. It is also possible to retrieve the information of the regression models, the constants of the regression model (see equations 1, 2 and 3), to make precise data-analysis. These additional features are also included to compensate for some of the weaknesses of the program. The main page of the program is presented in figure 3 and an example plot is illustrated in figure 4.



x-axis start*: 0
x-axis end*: 100
y-axis start*: 0
y-axis end*: 200
x label:
y label:
Legend text:
Title:
✓ Grid on
☐ Legend on
Plot size: Medium
Datapoint size: 6
Line width: 1
linear.txt
linear2.txt
exp.txt
log.txt

Figure 3: The user can import and plot the data from the main page.

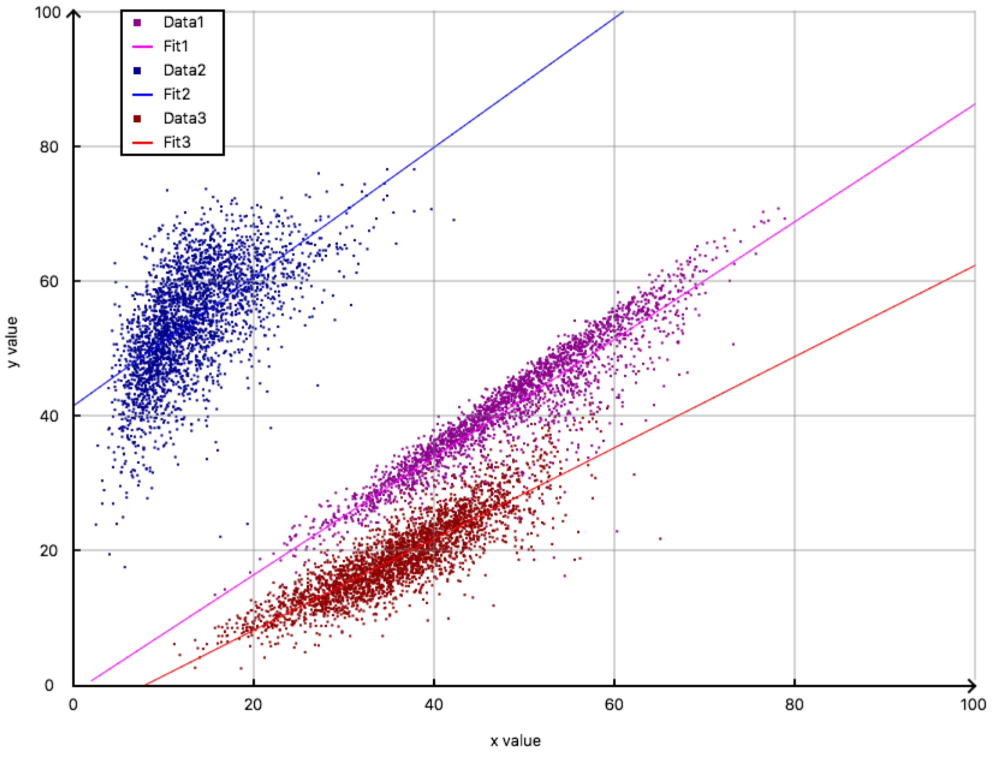


Figure 4: The user can plot many data-sets at the same time for comparison. The instructions for creating this picture are presented in appendix B

The user can also choose whether only to plot the data-points or the regression curve. This is possible if the line width for the regression model is determined to be zero or the data-point size is set to zero.

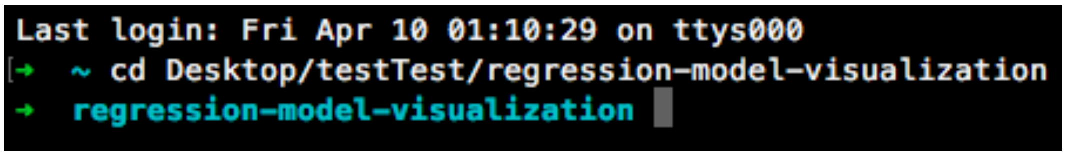
3 The program

This section gives further explanations about the final program. Section 3.1 describes the usage of the program. How is it started? What can you do with it? Program structure, such as the classes, is discussed in section 3.2. Data structures are illustrated in section 3.3 and supported file formats are described in section 3.4. Further information about the test files is presented in appendix A.

3.1 User interface

The usage of this program is divided into four different sub-parts: starting the program, importing data, modifying features, and plotting. Before the user can do anything with the program, the data-set must be imported. Ahead of plotting the data-set, the user can modify numerous plotting parameters, including the width of the axes.

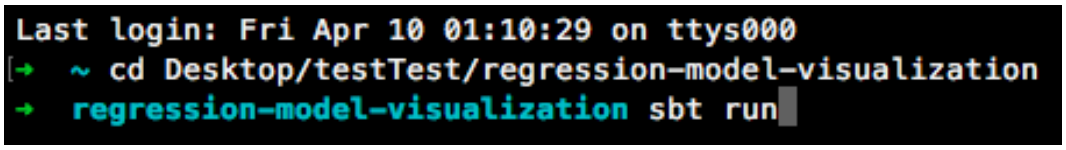
Starting the program is simple, with a command line. The user must first find the directory where the program is located. This can be done with *cd* commands, and in the end, the user should be inside the program directory, which is called *regression-model-visualization*. This is illustrated in figure 5.



```
Last login: Fri Apr 10 01:10:29 on ttys000
[→  ~ cd Desktop/testTest/regression-model-visualization
→  regression-model-visualization
```

Figure 5: The user can start the program from the folder called *regression-model-visualization*.

When the user is inside the folder *regression-model-visualization* (see. figure 5), the program is started with the command *sbt run*. This is expressed in figure 6.



```
Last login: Fri Apr 10 01:10:29 on ttys000
[→  ~ cd Desktop/testTest/regression-model-visualization
→  regression-model-visualization sbt run
```

Figure 6: The program is started with the command *sbt run*.

After the *sbt run* command (see figure 6), it will take around 20 seconds until the main page, presented in figure 3, will open. Now it is actually possible to import and plot data.

Importing the data-sets is simple. Once the main page is opened (see figure 3), the user must click the *Options* button on the upper left corner. Then the user should select *Import data* after which the user can choose from three different options: *Linear*, *Exponential* and *Logarithmic*. Thus, the user must be aware of the trend of the data-set before the data is imported. For example, the final sequence for importing linear data is:
Options→*Import data*→*Linear*. This is illustrated in figure 7.

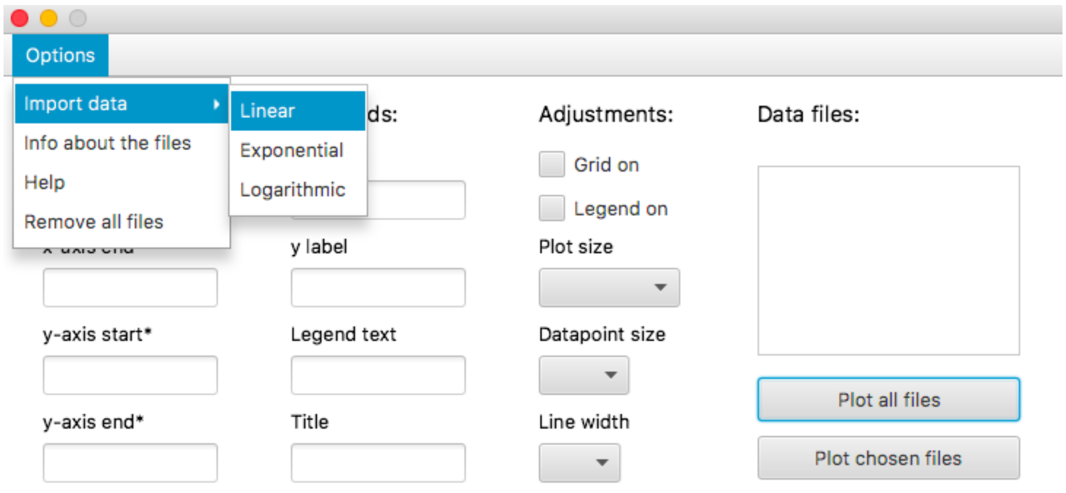
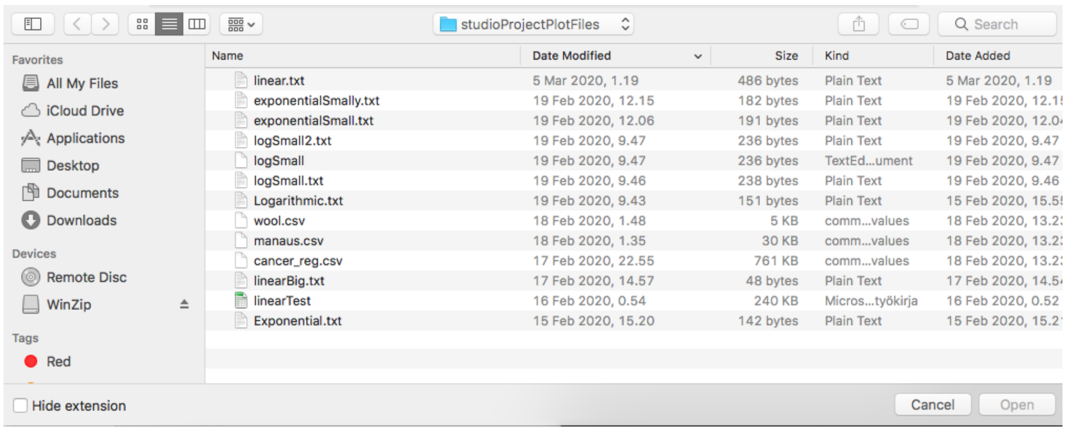


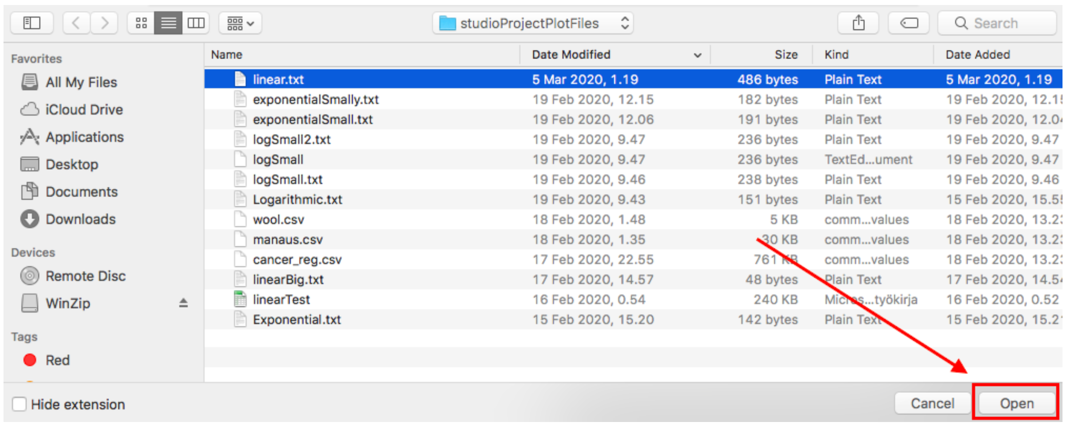
Figure 7: Importing linear data can be done by selecting *Options*→*Import data*→*Linear*.

Importing exponential data can be done by selecting *Options*→*Import data*→*Exponential*, and the same logic applies for the logarithmic data. Once the user has selected the type of data, the program will open a window illustrated in figure 8. Then the user must select the file for importing. This program supports txt, xlsx, and csv files. Further information about the correct file formats is presented in section 3.4 and in appendix A. Once the user has selected the file, the import is finished by clicking the *Open* button at the lower right corner. The selection of the file and the completion of the import are expressed in figure 9.



Name	Date Modified	Size	Kind	Date Added
linear.txt	5 Mar 2020, 1.19	486 bytes	Plain Text	5 Mar 2020, 1.19
exponentialSmally.txt	19 Feb 2020, 12.15	182 bytes	Plain Text	19 Feb 2020, 12.1
exponentialSmall.txt	19 Feb 2020, 12.06	191 bytes	Plain Text	19 Feb 2020, 12.0
logSmall2.txt	19 Feb 2020, 9.47	236 bytes	Plain Text	19 Feb 2020, 9.47
logSmall	19 Feb 2020, 9.47	236 bytes	TextEd...ument	19 Feb 2020, 9.47
logSmall.txt	19 Feb 2020, 9.46	238 bytes	Plain Text	19 Feb 2020, 9.46
Logarithmic.txt	19 Feb 2020, 9.43	151 bytes	Plain Text	15 Feb 2020, 15.5
wool.csv	18 Feb 2020, 1.48	5 KB	comm...values	18 Feb 2020, 13.2
manaus.csv	18 Feb 2020, 1.35	30 KB	comm...values	18 Feb 2020, 13.2
cancer_reg.csv	17 Feb 2020, 22.55	761 KB	comm...values	18 Feb 2020, 13.2
linearBig.txt	17 Feb 2020, 14.57	48 bytes	Plain Text	17 Feb 2020, 14.5
linearTest	16 Feb 2020, 0.54	240 KB	Micros...työkirja	16 Feb 2020, 0.52
Exponential.txt	15 Feb 2020, 15.20	142 bytes	Plain Text	15 Feb 2020, 15.2

Figure 8: The user can select txt, xlsx or csv files for importing.



Name	Date Modified	Size	Kind	Date Added
linear.txt	5 Mar 2020, 1.19	486 bytes	Plain Text	5 Mar 2020, 1.19
exponentialSmally.txt	19 Feb 2020, 12.15	182 bytes	Plain Text	19 Feb 2020, 12.1
exponentialSmall.txt	19 Feb 2020, 12.06	191 bytes	Plain Text	19 Feb 2020, 12.0
logSmall2.txt	19 Feb 2020, 9.47	236 bytes	Plain Text	19 Feb 2020, 9.47
logSmall	19 Feb 2020, 9.47	236 bytes	TextEd...ument	19 Feb 2020, 9.47
logSmall.txt	19 Feb 2020, 9.46	238 bytes	Plain Text	19 Feb 2020, 9.46
Logarithmic.txt	19 Feb 2020, 9.43	151 bytes	Plain Text	15 Feb 2020, 15.5
wool.csv	18 Feb 2020, 1.48	5 KB	comm...values	18 Feb 2020, 13.2
manaus.csv	18 Feb 2020, 1.35	30 KB	comm...values	18 Feb 2020, 13.2
cancer_reg.csv	17 Feb 2020, 22.55	761 KB	comm...values	18 Feb 2020, 13.2
linearBig.txt	17 Feb 2020, 14.57	48 bytes	Plain Text	17 Feb 2020, 14.5
linearTest	16 Feb 2020, 0.54	240 KB	Micros...työkirja	16 Feb 2020, 0.52
Exponential.txt	15 Feb 2020, 15.20	142 bytes	Plain Text	15 Feb 2020, 15.2

Figure 9: Once the user has selected the file for importing, the import is completed by pressing the button *Open* at the bottom right corner.

When the steps described in figures 7, 8 and 9 are completed, the program will return to the main page. If the import was successful (correct file format and correct format for the data-points), the imported file will appear on the main page. This is illustrated in figure 10.

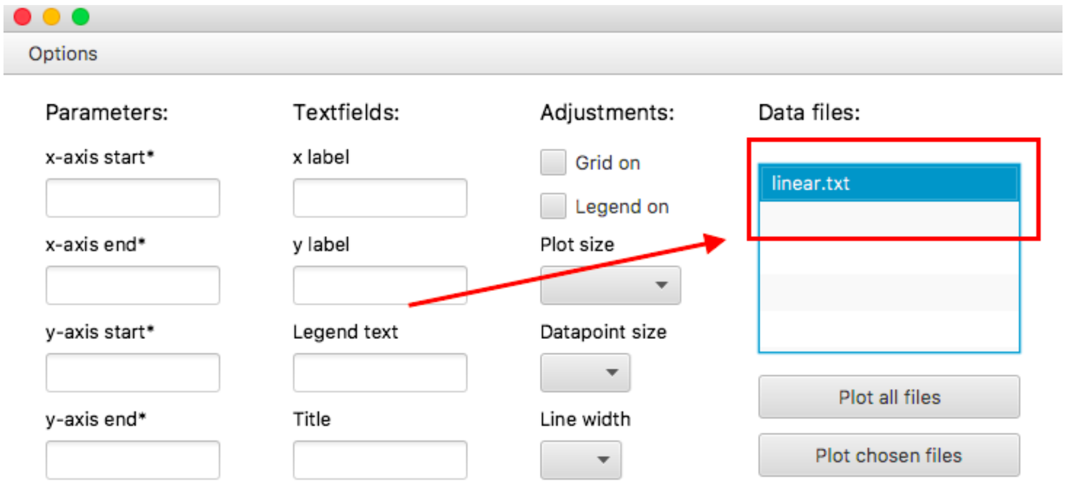


Figure 10: If the import was successful, the file will appear on the main page.

If the import was not successful, the user should follow the instructions given by the alert messages. Further information about the alerts and errors are presented in appendixes.

After successful import, the user can now plot the imported files, but before this, we introduced another file to illustrate the comparison feature of this program. The user can plot the imported data by pressing the button *Plot all files* or the button *Plot chosen files*. This is described in figure 11. If the option *Plot chosen files* is used, the files have to be selected from the list on the main page. The list is located inside the blue square in figure 11.

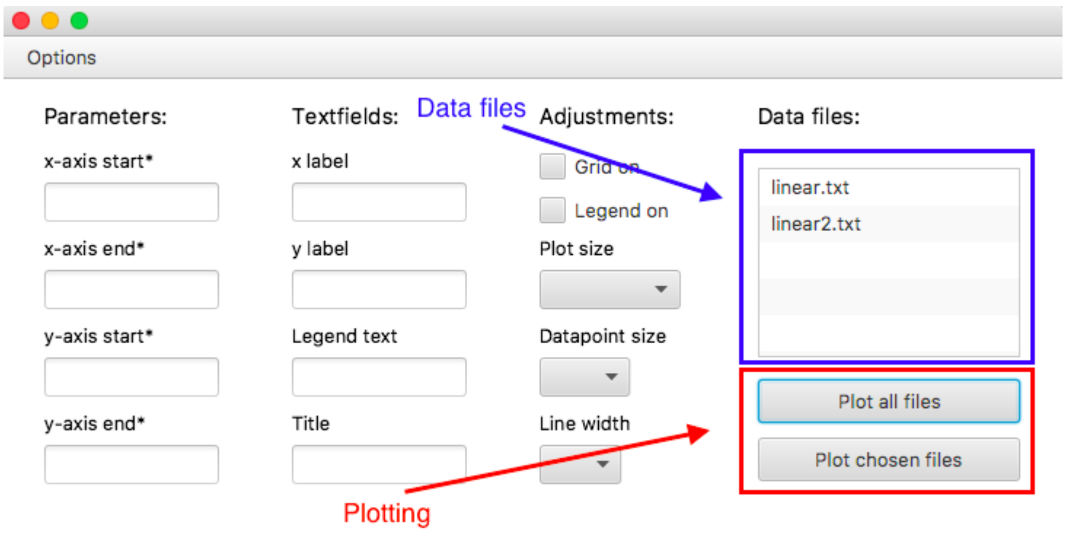


Figure 11: The plot is produced by pressing one of the buttons inside the red square.

If the user does not specify any parameters for the plot, the program will automatically calculate appropriate constraints for the axes. When the button *Plot all files* is pressed, the program will show the window presented in figure 12.

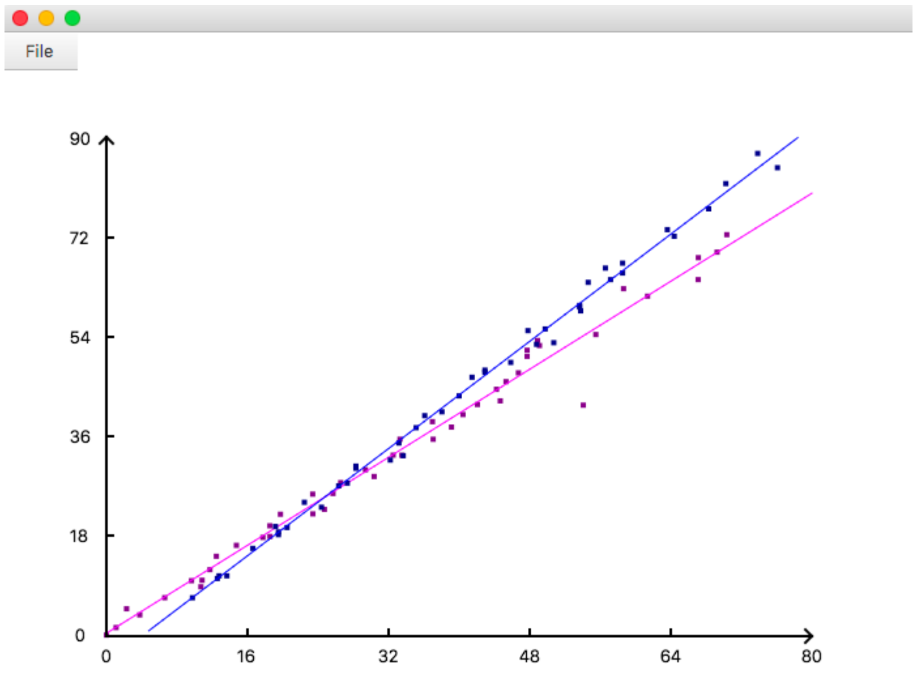


Figure 12: This figure illustrates a simple plot generated by the program.

Returning to the main page is easy. The user must press the button *File* in the upper left corner and then button *Back*. This is described in figure 13.

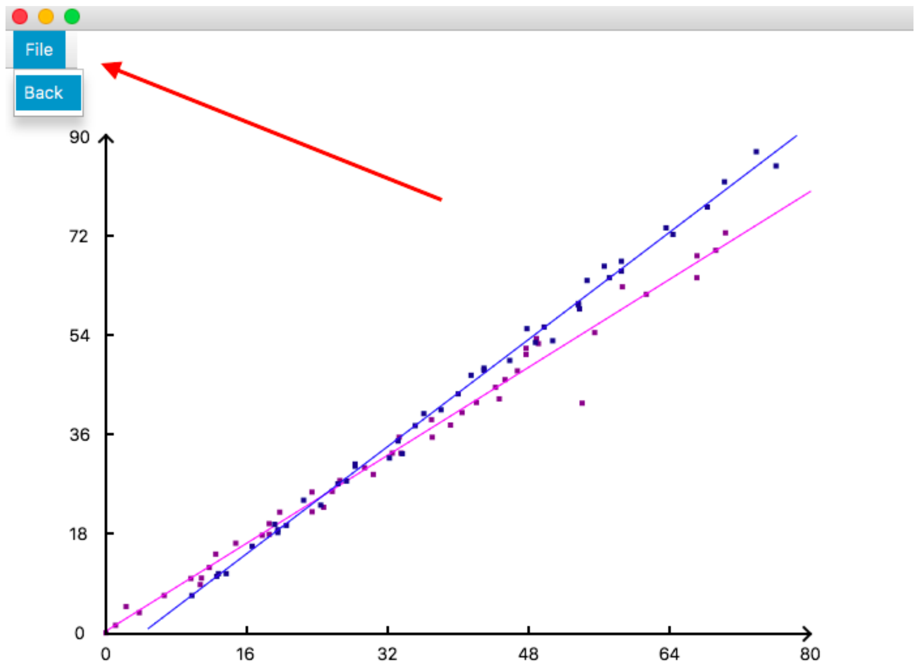
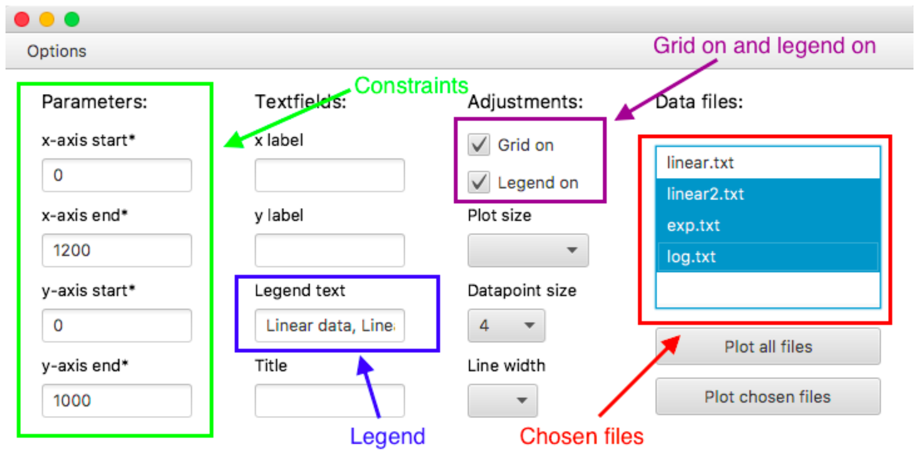


Figure 13: Returning to the main page is easy by selecting *File→Back*.

The final sub-part about the usage is to modify the plotting-parameters. Before the modifications, we imported one exponential model (*Options→Import data→Exponential*) and one logarithmic model (*Options→Import data→Logarithmic*). Then, we select x-axis constraints between 0-1200 and y-axis constraints between 0-1000. We also select grid on and legend on options and write the text, *Linear data, Linear fit, Exp data, Exp fit, Log data, Log fit*, on the legend field. Legend texts are separated with commas.

After these modifications, we choose three files (mac computers. shift + ↓) from the file list and select *Plot chosen files*. All of these steps are described in figure 14.



x-axis start* 0
x-axis end* 1200
y-axis start* 0
y-axis end* 1000
x label
y label
Legend text Linear data, Line
Title
✓ Grid on
✓ Legend on
Datapoint size 4
linear.txt
linear2.txt
exp.txt
log.txt

Figure 14: The modifications for the plot can be done from the main page. After the modifications, we choose *Plot chosen files*.

The result of the modifications in figure 14 is presented in figure 15. This result was produced by choosing *Plot chosen files*.

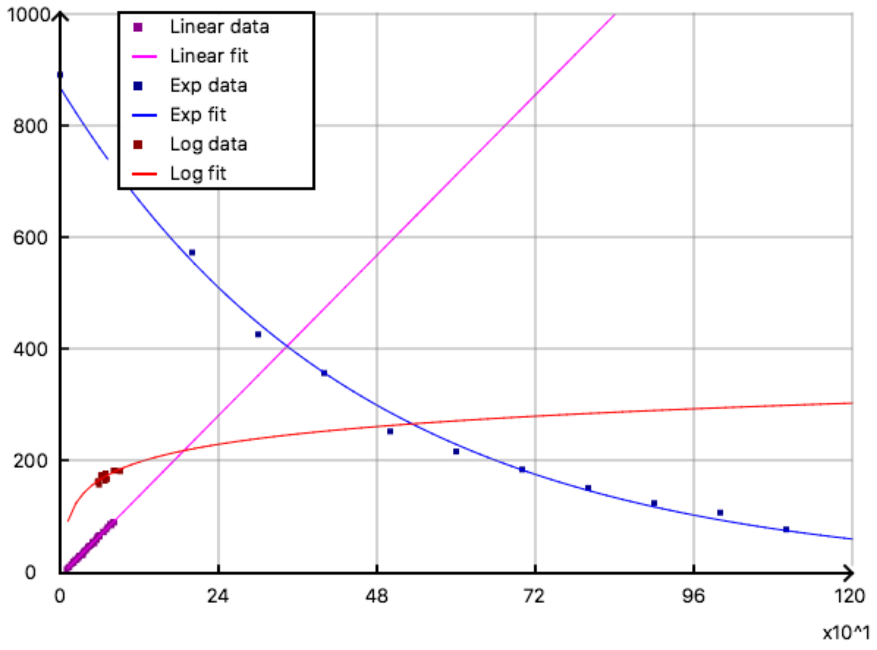
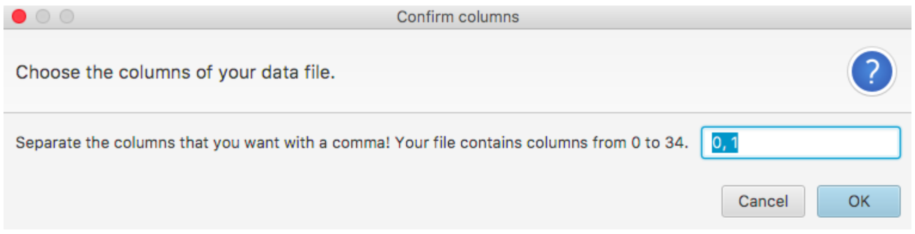


Figure 15: The modifications presented in figure 14 result in the plot illustrated in this figure.

Now we are able to start the program, import data and plot the data with some modifications. These examples, however, only presented cases where txt files were used.

In the next example, we will import one csv file. This is done exactly the same way as shown in figures, 8, and 9. After the user selects a csv file and presses the button *Open* (see figure 9), the window presented in figure 16 will open. Exactly the same type of window will be shown if xlsx file was imported.

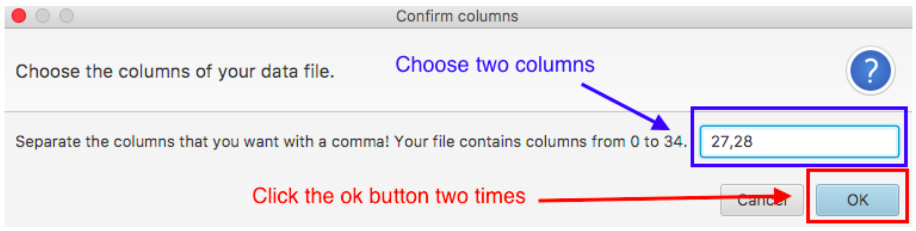

Confirm columns

Choose the columns of your data file.

Separate the columns that you want with a comma! Your file contains columns from 0 to 34. 0, 1

Figure 16: Importing a csv or xlsx file will result in the following window to open.

The window presented in figure 16 will inform about the number of columns in the dataset: *Your file contains columns from 0 to 34.*. The user can choose which two columns to import together. The first column will represent the x values, while the second column will be the y values. The user must separate the two chosen columns with a comma. The procedure to choose the columns is illustrated in figure 17.


Confirm columns

Choose the columns of your data file.

Choose two columns

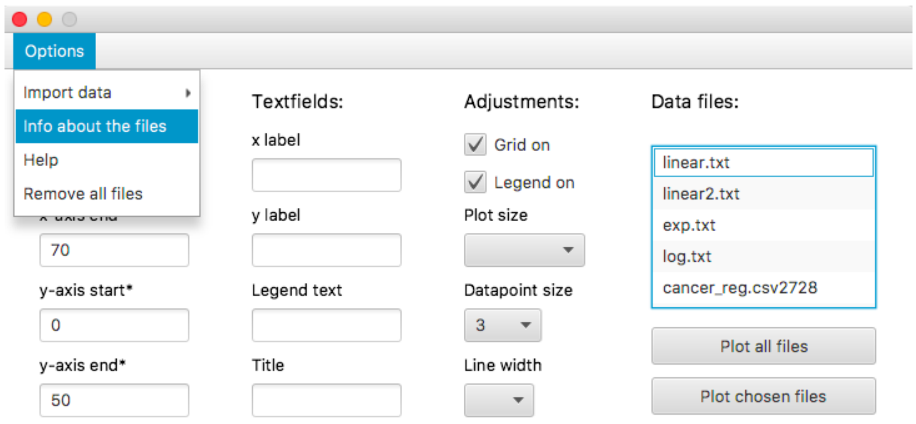
Separate the columns that you want with a comma! Your file contains columns from 0 to 34. 27,28

Click the ok button two times

Figure 17: The user selects two columns from the csv or xlsx file. Then the *OK* button is pressed two times.

After the columns are selected, the user presses the *OK* button two times to complete the import. If the import were successful, the file would be shown on the main page (see figure 10). The file name is not the original file. The program will generate a new data from the chosen two columns. If the original file was names *cancer.csv* and the selected columns were 27 and 28, the program will create a file called *cancer.csv2728*.

This program was not only generated for data visualization but also data analysis. For this reason, there is a possibility to retrieve information about the regression models. This can be done by selecting *Options→Info about the files* from the main page. This is described in figure 18.



x-axis end: 70

y-axis start*: 0

y-axis end*: 50

✓ Grid on

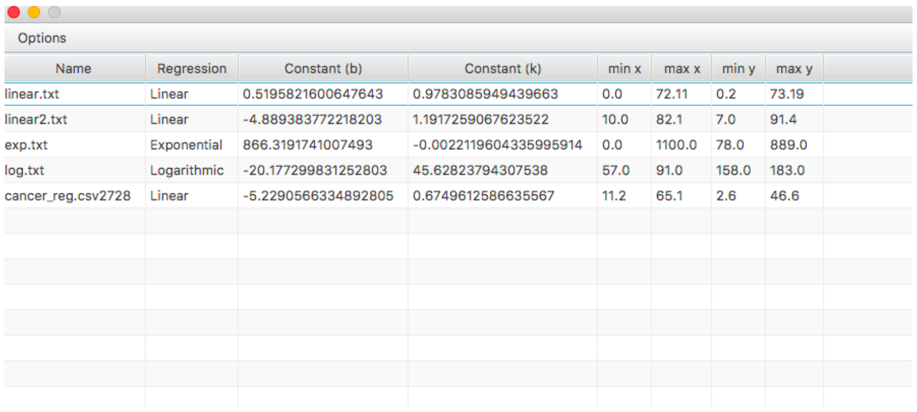
✓ Legend on

Datapoint size: 3

Data files: linear.txt, linear2.txt, exp.txt, log.txt, cancer_reg.csv2728

Figure 18: The user can retrieve the information about the regression models by selecting *Options→Info about the files*. This will open a new window.

If the user follows the instructions presented in figure 18, the window shown in figure 19 will open. This window contains, for example, the information about the constants of the regression models.



Name	Regression	Constant (b)	Constant (k)	min x	max x	min y	max y
linear.txt	Linear	0.5195821600647643	0.9783085949439663	0.0	72.11	0.2	73.19
linear2.txt	Linear	-4.889383772218203	1.1917259067623522	10.0	82.1	7.0	91.4
exp.txt	Exponential	866.3191741007493	-0.0022119604335995914	0.0	1100.0	78.0	889.0
log.txt	Logarithmic	-20.177299831252803	45.62823794307538	57.0	91.0	158.0	183.0
cancer_reg.csv2728	Linear	-5.2290566334892805	0.6749612586635567	11.2	65.1	2.6	46.6

Figure 19: The information about the regression models is presented in the info page.

This section illustrated the usage of the program. More information about the usage is represented in appendix B. This appendix contains a step by step instructions on how to create the figure 4 using the test-files.

3.2 Program structure

The final program structure is similar compared to the plan. The only significant difference is that additional classes were added to make the program more advanced: CoordinateSystem, Legend, and Logarithmic. The program structure closely follows the UML-diagram presented in figure 1. CoordinateSystem and Legend introduce new features for the Plot class, and the Logarithmic class introduces an extra regression model for the visualization. The final UML-diagram of the program is presented in figure 20. The relationships of the classes are more complex than in the diagram shown in figure 20. Because of the little experience related to large programming projects, numerous modifications about the structure had to be done; thus, the structure is quite challenging to understand. The diagram, however, illustrates the basic idea of the program.

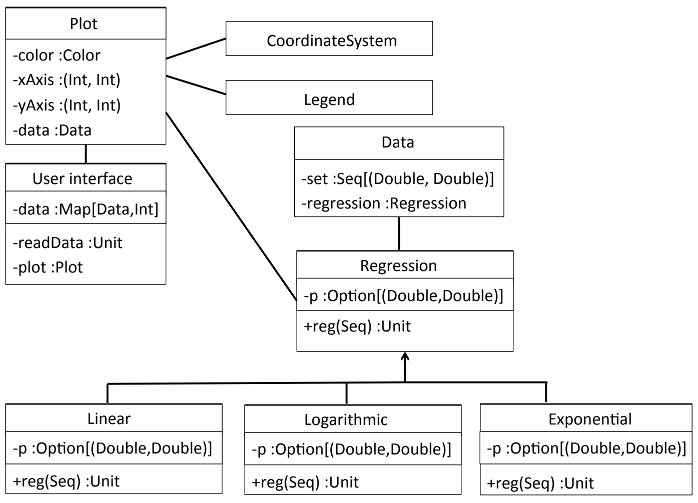


Figure 20: The information about the regression models is presented in the info page.

The class data provides support for reading the data for three different types of files: txt, xlsx, and csv. It also creates a sequence of tuples that the Regression class can use to generate the regression models.

Regression class provides useful methods that the classes Linear, Exponential, and Logarithmic models can utilize when calculating the models. These three classes generate the constants for the regression models. The algorithms behind these classes are discussed in section 4.

Classes Plot, CoordinateSystem, and Legend provide the visualization for the program. They generate the coordinate system, legend, and the class Plot maps the data-set into the coordinate system. The class Plot also utilizes the regression models provided by the regression classes to create the regression curve. Finally, with the help of the user interface, the user can import and visualize the data-sets. The classes and their most important roles behind the program are explained in table 1.

Table 1: This table presents the most important roles of each of the classes.

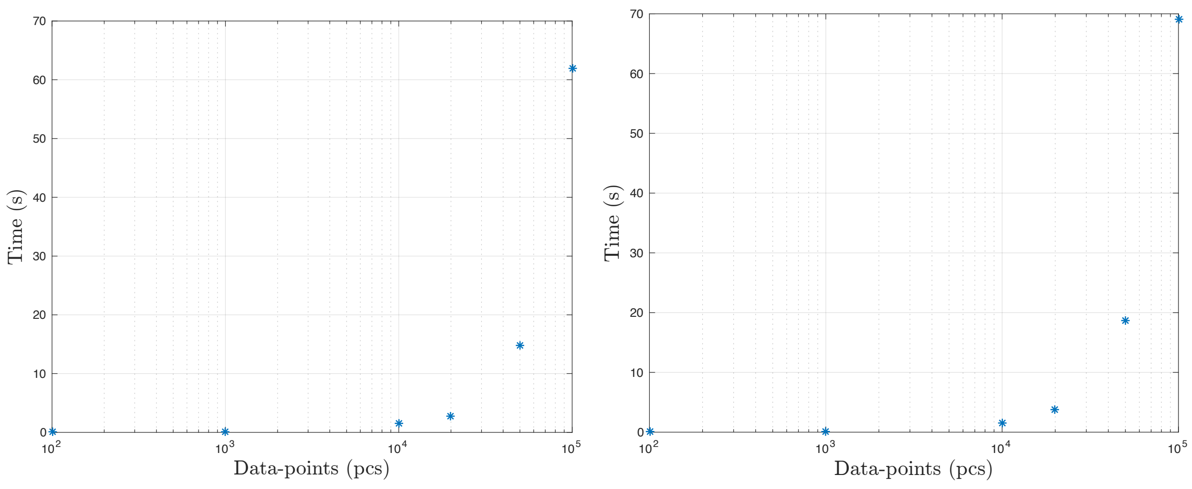
Class	**Role**
Data	Reading the data-sets and creating sequances for the regression classes.
Regression	Provides methods for the classes Linear, Exponential and Logarithmic
Linear	Generates the regression constants for the linear data-set.
Exponential	Creates the regression constants for the exponential data-set.
Logarithmic	Generates the regression constants for the logarithmic data-set.
Plot	Maps the data-points within the constraints of the window and provides a sequences of data points and regression models that can be plotted in the user interface.
CoordinateSystem	Provides the visualization of the coordinate system and grid.
Legend	Provides the visualization of the legend.
UI	Creates an environment for the user to import and visualize data-sets.

3.3 Data structures

This program only uses basic data structures provided by Scala. The class Data (see figure 20) processes the data-sets. These data-sets are saved into immutable sequences (Seq[(Double,Double)]). The data-points could have been saved separately, but storing them together makes it easier to track which data-points belong with each other.

These data-points and the generated lines (see secttion 4.3) were visualized with *ScalaFX* lines and squares. These lines and squares were first mapped with the equations 13 and 14 to fit them inside the program window. They were then stored in a sequence (Seq[Shape]) which was the only possible option as I used *ScalaFX Scene* and *Pane* to visualize the lines and squares, and *Pane* only supports immutable Lists.

However, better methods and data-structures could have been used, but within the time limits of this project, everything could not have been analyzed perfectly. The reading and plotting times of the data-sets is illustrated in figure 21. The performance of the program starts being slower when more than 50 000 data-points are read or plotted.



(a) Reading

(b) Plotting

Figure 21: The reading and plotting of the data-sets starts getting slower when the size of the data-set is over 50 000.

The file that was used for the testing is called 50000.csv. It is included in test-files, and the size of it was modified during testing.

3.4 Files

This program supports three types of file formats: txt, xlsx, and csv. The files have to be constructed precisely correctly since otherwise the program will be unable to read the data, and errors occur. These errors are handled with alerts.

The correct form of the text files is represented below. Every line of the text file must contain two doubles, and they are separated with spaces.

```
0.0 889.0
100.0 692.0
200.0 572.0
300.0 426.0
```

```
400.0 357.0
500.0 253.0
```

If the numbers are separated with commas, an error will occur, and the erroneous line is changed to 0.00000 0.00000. The user has the responsibility to remove erroneous data-points from the files.

If csv or xlsx files are used, the user can choose which of the columns to import. The selected columns must be the same length, and they must only contain doubles. In other cases, an error will occur, and the user should follow the instructions given by the program.

4 Algorithms

This project was mathematical, and numerous algorithms were used. The main algorithms can be assigned into three different categories: creating the regression models, mapping the data-sets into the window, and generating the curves of the regression models. The creation of the regression models is discussed in section 4.1. The mapping of the data-sets is described in section 4.2. The generation of the curves is presented in section 4.3.

4.1 Regression models

This program supports three types of regression models: linear, exponential, and logarithmic. This program creates the regression models with the least-squares method. The Least-squares method minimizes the distances of the data-points to the regression curve [1]. This is illustrated in figure 22.

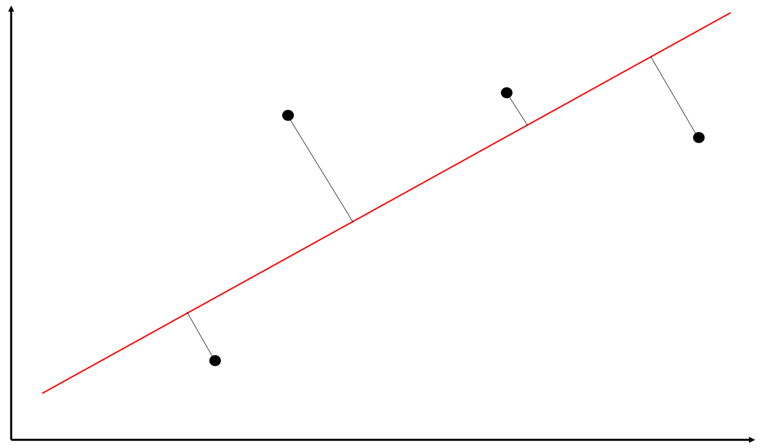


Figure 22: Least squares method minimizes the distances from the data-points to the fitted curve.

The linear regression model generates the best-suited constants k and b for the linear model (see equation 1). The best-suited constants are calculated with the equations 4 and 5:

$$k = \frac{n(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2} \tag{4}$$

$$b = \frac{n(\sum x_i^2)(\sum y_i) - (\sum x_i)(\sum x_i y_i)}{n \sum x_i^2 - (\sum x_i)^2}, \tag{5}$$

where n denotes the size of the data-set. For the exponential regression model, a similar idea is used. We are trying to fit an exponential curve (see equation 2) and generate the most suited constants α and β. The constants are best suited if the equation 6 is minimized [2]:

$$L = \sum_{i=1}^{n} y_i (\ln y_i - \alpha - \beta x_i)^2, \tag{6}$$

where n denotes the size of the data-set. The optimal values for the constants α and β are obtained from the equations 7 and 8:

$$\alpha = \frac{\sum(x_i^2 y_i) \sum(y_i \ln y_i) - \sum(x_i y_i) \sum(x_i y_i \ln y_i)}{\sum y_i \sum(x_i^2 y_i) - (\sum x_i y_i)^2} \tag{7}$$

$$\beta = \frac{\sum y_i \sum(x_i y_i \ln y_i) - \sum(x_i y_i) \sum(y_i \ln y_i)}{\sum y_i \sum(x_i^2 y_i) - (\sum x_i y_i)^2}. \tag{8}$$

This project also supports a third regression model, logarithmic regression, which is presented in equation 3. Fitting the best-suited constants a and b can be done with the equations 9 and 10 [3]:

$$c = \frac{n \sum(y_i \ln x_i) - \sum y_i \sum \ln x_i}{n \sum (\ln x_i)^2 - (\sum \ln x_i)^2} \tag{9}$$

$$a = \frac{\sum y_i - c \sum(\ln x_i)}{n}, \tag{10}$$

where n denotes the size of the data-set. As all of the regression models contain similar sums, the class Regression provides some summing methods to support all three regression models.

4.2 Mapping data-sets

Mapping the data-sets is a crucial part of the plotting-tool. The window that is used for plotting has a defined height and width, where the horizontal values lie between 0 and x_w, and the vertical values are between y_w and 0. The variables x_w and y_w denote the width and height of the window, respectively.

The data-set has x-values between x_s and x_b where x_s denotes the smallest and x_b the biggest x-value of the data-set while the y-values lie between y_s and y_b. To be able to plot the data-set, we have to map the data-points between the constraints set by the window. The user might also want to set constraints for the data-set, but here we do not take that into account to simplify the problem. The mapping functions for the x - and y-values are presented in the equations 11 and 12:

$$[x_s, x_b] \to [0, x_w] \tag{11}$$
$$[y_s, y_b] \to [y_w, 0], \tag{12}$$

where the y-values are mapped the other way around as the upper left corner of the window denotes the y-value zero. The window is presented in figure 23.

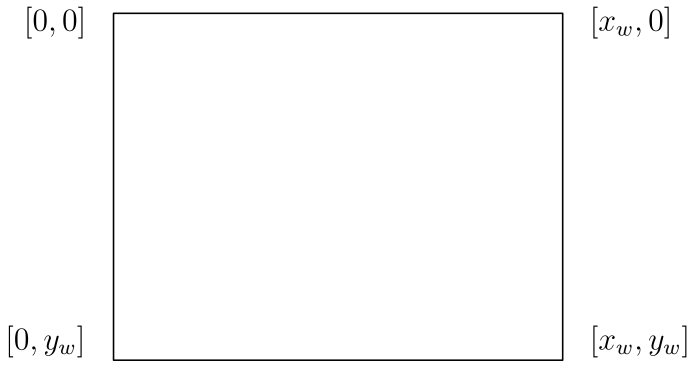


Figure 23: This figure shows the coordinates of the corners of the window.

Each of the data-points (x_i, y_i) are mapped with the equations 13 and 14:

$$x_i = \frac{x_w(x_i - x_s)}{x_b - x_s} \tag{13}$$
$$y_i = \frac{y_w(y_s - y_i)}{y_b - y_s} + y_w, \tag{14}$$

if the user sets the constraints for the x - and y-values, we would only change the values x_s, x_b, y_s and y_b for the constraints provided.

4.3 Generating curves

The perfect mathematical world is not part of the Scala; hence, creating perfect curves is impossible, meaning that discrete methods must be applied. The regression curve is divided into discrete sections for all of the regression types: linear, exponential, and logarithmic. The division is mandatory for the exponential and logarithmic models, but for the linear model, it is only applied to avoid plotting the linear model over the legend box. This section only describes the methods to generate exponential curves for the regression model as linear and logarithmic curves are made similarly.

The exponential regression model constants are calculated with the equations 7 and 8. Thus, we get the exponential trend for the data-set given by the equation 2. As it is only possible to plot lines with Scala, we divide the model into discrete parts. This is shown in figure 24. The more discrete parts, the more accurate the plotted "curve" is going to appear.

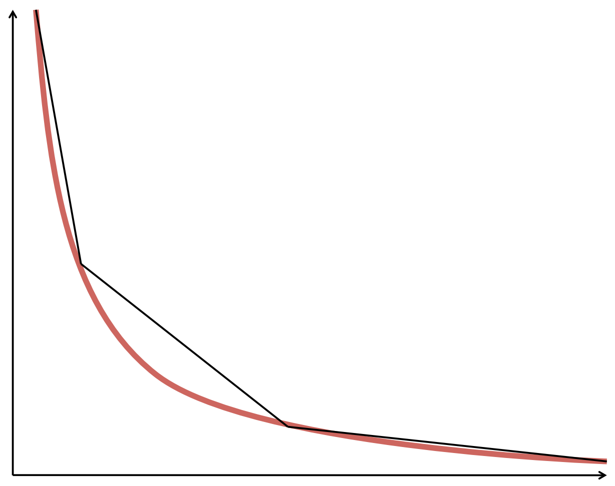


Figure 24: The mathematically perfect exponential function (red curve) is divided into discrete parts (black lines).

The exponential function $y = \alpha e^{\beta x}$ describes the data. The x-values of the data-set lie between $[x_s, x_b]$; thus, we divide these constraints into discrete parts. The amount of the discrete parts is fixed as the program divides the constraints always into 100 equal lengths. This amount was chosen because it gave visually great results, and the program calculations were efficient. The division is completed with the equation 15:

$$D_n = \sum_{n=0}^{100} (x_s + n\frac{x_b - x_s}{100}), \tag{15}$$

where D_n is the list, containing 101 data-points, and each of the data-points has a fixed length between the adjacent data-points. These data-points are used to calculate the discrete lines that are plotted. How are these lines formed? The formation of lines is illustrated in figure 25, where one of the parts is studied.

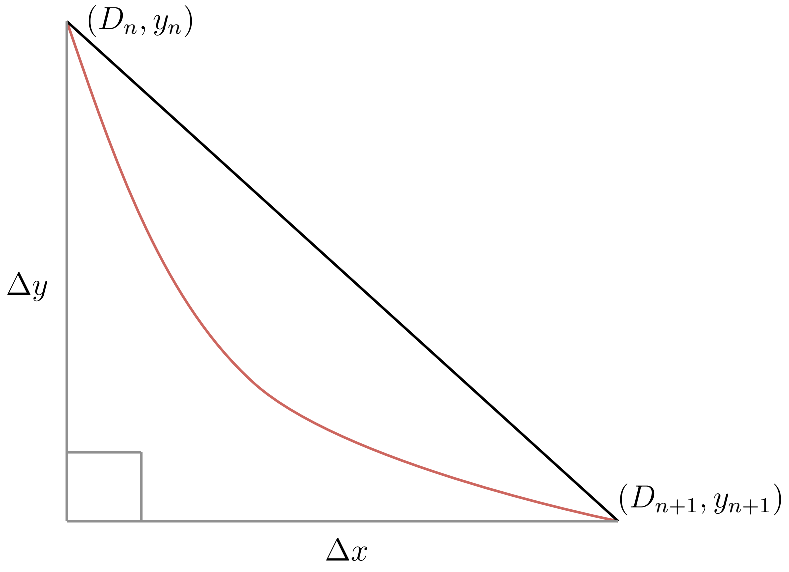


Figure 25: This figure views one of the discrete parts.

The starting point of each of the lines is (D_n, y_n) and the ending point is (D_{n+1}, y_{n+1}). Hod do we calculate y_n and y_{n+1}? For this we use the equation 2, thus, we get the equations 16 and 17:

$$y_n = \alpha e^{\beta D_n} \tag{16}$$

$$y_{n+1} = \alpha e^{\beta D_{n+1}}. \tag{17}$$

Once we have calculated the starting points and ending points for all of the values in the list D_n, we map these points with the equations 13 and 14 to fit the lines inside the window. The exact same type of procedure applies to the linear and logarithmic models.

5 Testing

This section discusses the testing process of the program; the methods that were used to test the program during the building process together with the testing of the final result. The testing is divided into four different categories: regression models, reading data, plotting, and user interface. The testing of the regression models is described in section 5.1. The testing process of reading the data sets is expressed in section 5.2. The explanation about the testing process of the plots is described in section 5.3, and the testing of the user interface is discussed in section 5.4.

5.1 Regression models

The testing process of the regression models was simple, with the help of a known program that works properly. The same data set was analyzed with this program and MATLAB.

Figure 26 shows the results provided by this program, and the results from MATLAB are illustrated in figure 27. The regression models of these programs are compared in table 2.

Table 2: This program and MATLAB provide exactly the same results for the regression models provided in figures 26 and 27.

Linear model	Matlab (b)	This program (b)	Matlab (k)	This program (k)
1 (red)	41.480746	41.480746	0.956832	0.956832
2 (blue)	-0.960107	-0.960107	0.871267	0.871267
3 (violet)	-5.229056	-5.229056	0.674961	0.674961

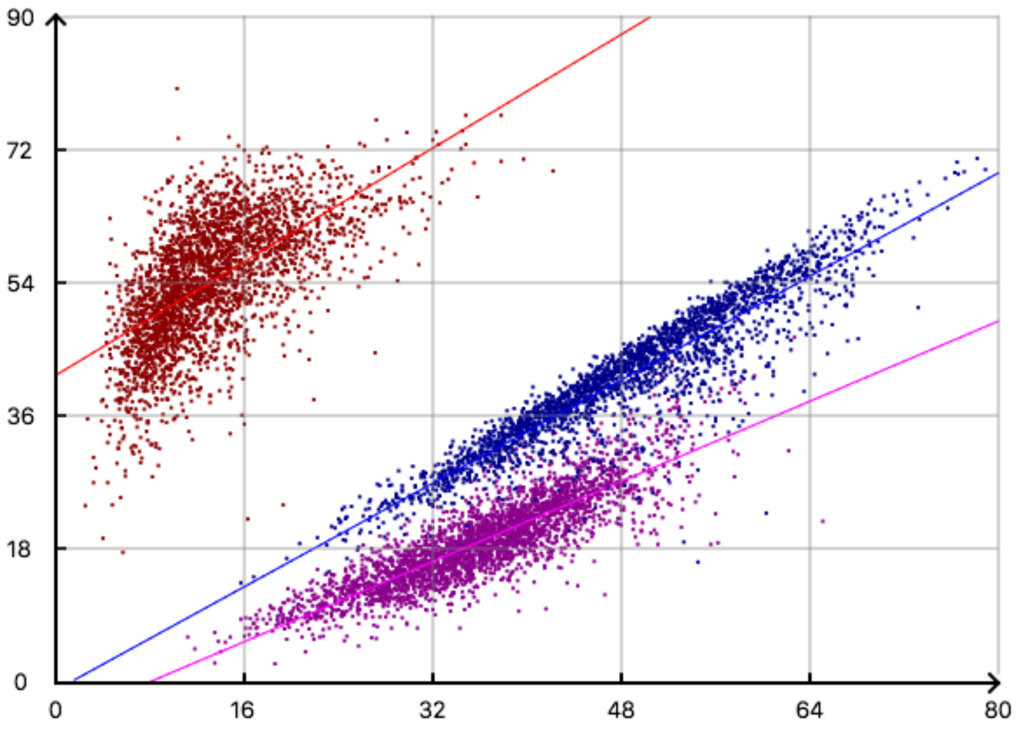


Figure 26: This figure represents the result provided by this program.

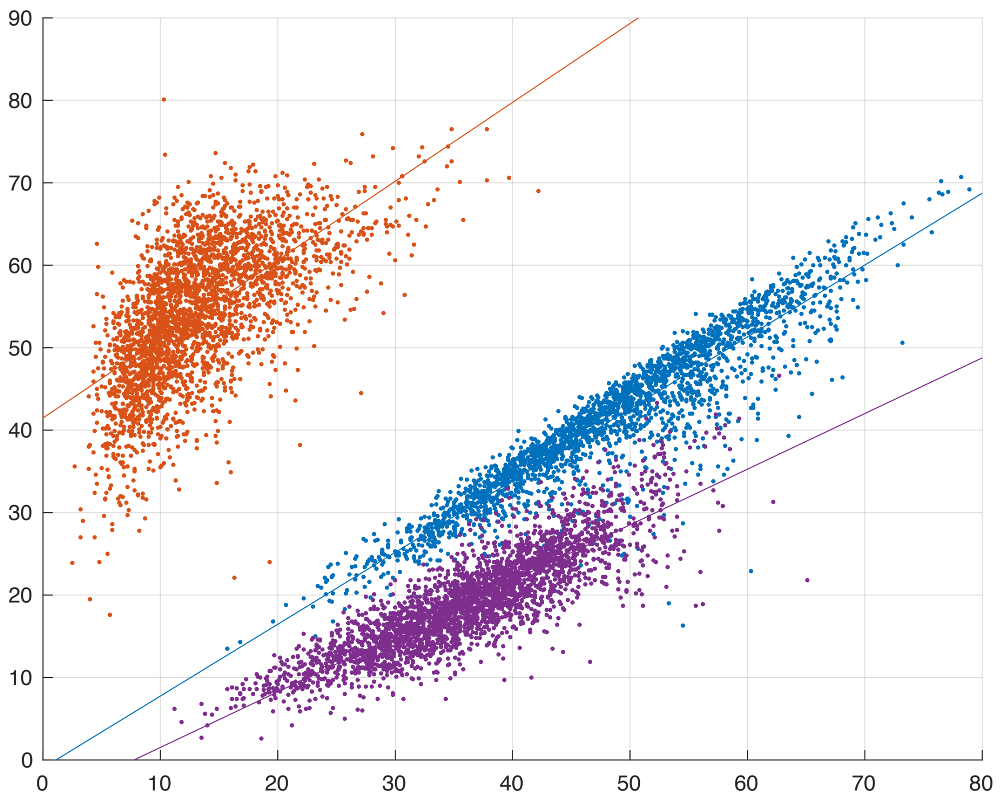


Figure 27: This figure is constructed with MATLAB.

As we can observe from the table 2, this program, and MATLAB gave precisely the same result for linear regression which concludes that the program generates correct models (this was just an example test). The figures 26 and 27 are also extremely similar, which strengthens the belief that this program works correctly. To conclude that exponential and logarithmic models also operate correctly, tables 3 and 4 are provided, where exponential and logarithmic models of this program are compared to MATLAB.

Table 3: This program and MATLAB provide similar results for the regression models provided in figures 28 and 29.

Model	Matlab (α)	This program (α)	Matlab (β)	This program (β)
Exponential	880.7	866.3	-0.002286	-0.002211

Table 4: This program and MATLAB provide similar results for the regression models provided in figures 28 and 29.

Model	Matlab (a)	This program (a)	Matlab (c)	This program (c)
Logarithmic	-20.18	-20.17	45.63	45.62

This program and MATLAB provide similar results for the exponential and logarithmic regression models as it can be obtained from the tables 3 and 4. The little differences could be caused by different methods used for calculating the regression models. The methods used by this program are expressed in equations 7, 8, 9 and 10. The models that are used in tables 3 and 4 are presented in figures 3 and 4.

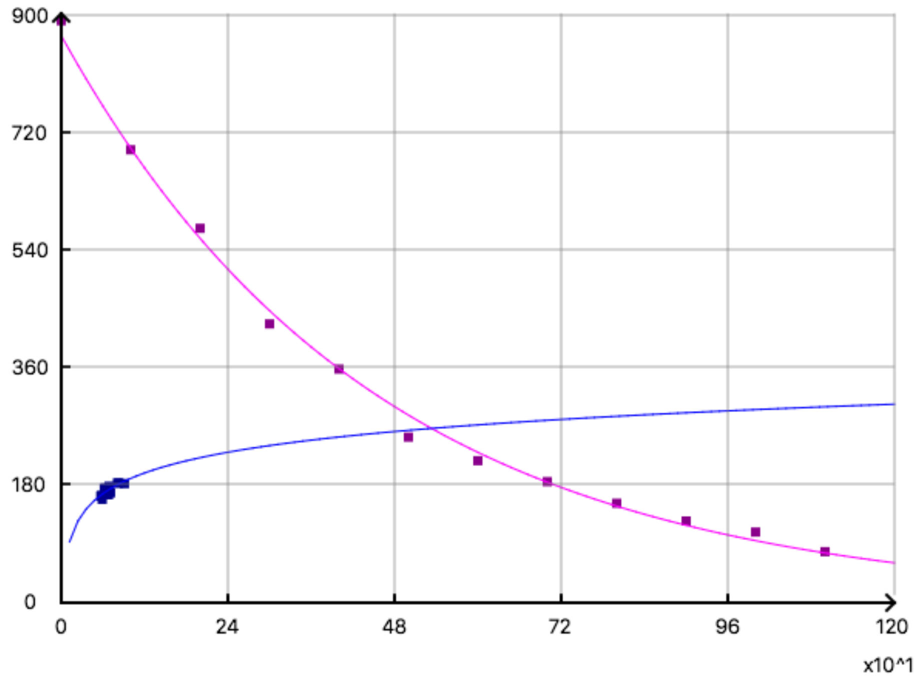


Figure 28: This figure represents the result provided by this program.

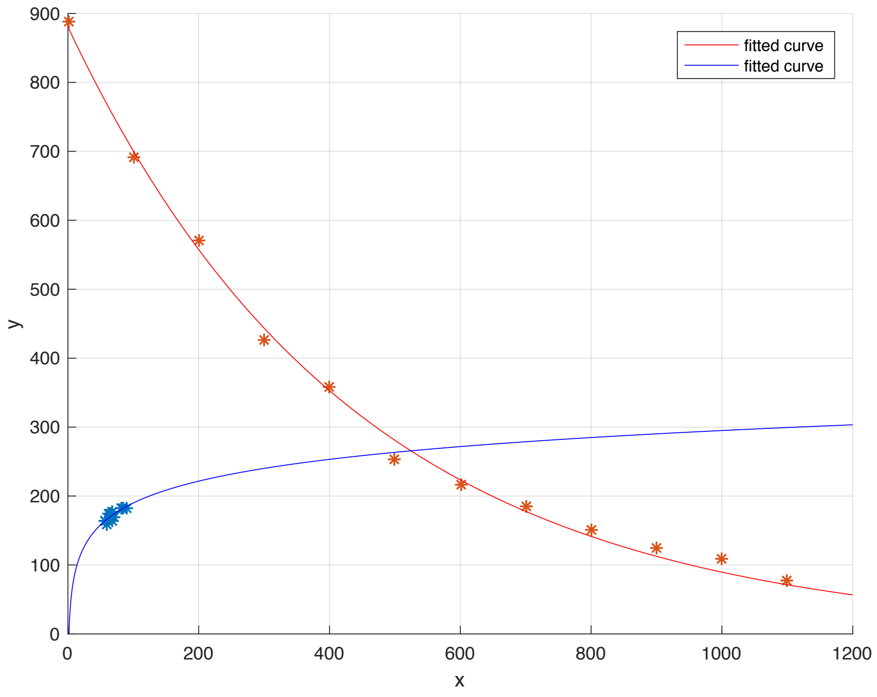


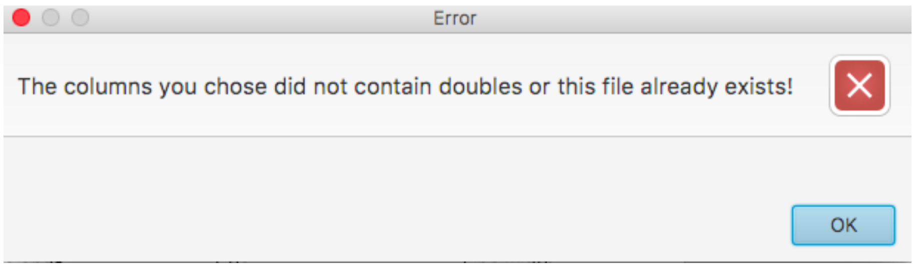
Figure 29: This figure is constructed with MATLAB.

The type of testing presented in this section was the primary method for testing the regression models. Obviously, more tests were performed, and these were just some examples from the tests. The visual aspect was also studied, and it is easy to observe that the figures provided by this program and MATLAB are extremely similar.

5.2 Data reading

There are two cases where data must be read: importing data and reading the text field on the main page. Importing the data was tested with the correct inputs and with incorrect ones. The program is strict with the xlsx and csv files as they have to be formed precisely correctly. If the program notices an error with csv or xlsx files, the importing process will be stopped and the import will fail. However, many csv files were found on the internet, and all of them worked properly if the columns of the data were double types. Numerous programs, including Comsol, create csv or xlsx files from the simulations they perform. These files were also tested, and the program read the data generated by Comsol successfully.

This program will most probably work with data-sets generated by programs, but errors might appear if the data-set was created by a human. If the xlsx or csv file contains a wrong type of data the error message presented in figure 30 will appear.


Error

The columns you chose did not contain doubles or this file already exists!

Figure 30: This error message will appear if the csv or xlsx file was formed incorrectly.

Many incorrect and correct files were tested to conclude that the reading process worked properly, and the error message appears if the user tries to import incorrect data. Still, this program will not recognize empty xlsx files as incorrect.

If the user tries to import txt files, and the txt file contains erroneous lines, the program will change the erroneous line to 0.00000 0.00000. More information about correct type of text files is presented in section 3.4 and appendix A. Different types of incorrect data was tested to conclude that the program realises the incorrect data.

The text fields are also sensitive to human errors. The errors related to text fields were also tested by creating many different incorrect and correct inputs. This was done bu myself, but this program was also given to some other people for testing purposes. They tried to come up with ridiculous data, and many errors were realized during this process.

5.3 Plotting

The testing related to plotting was mostly done visually by comparing the results to other programs, such as MATLAB. Figure 26 and 27 represent the same data visualized by this program and MATLAB, respectively, and it is easy to observe that the results provided by this program are similar. The same analysis applies for figures 28 and 29.

The most important feature to modify the plots is the selection of constraints. With this feature, it is possible to zoom-in and zoom-out for more accurate data analysis. Zooming-out is more critical because that way, no information is lost (data-points) and the user can extrapolate the trend of the data with the regression model. Zooming in is also possible to some extent, but this is not important for data-analysis purposes because data-points are lost (from sight).

Testing the feature of changing the constraints is easy by comparing the results with MATLAB. The zooming out is tested by changing the constraints in figures 26 and 27. The results are presented in figures 31 and 32.

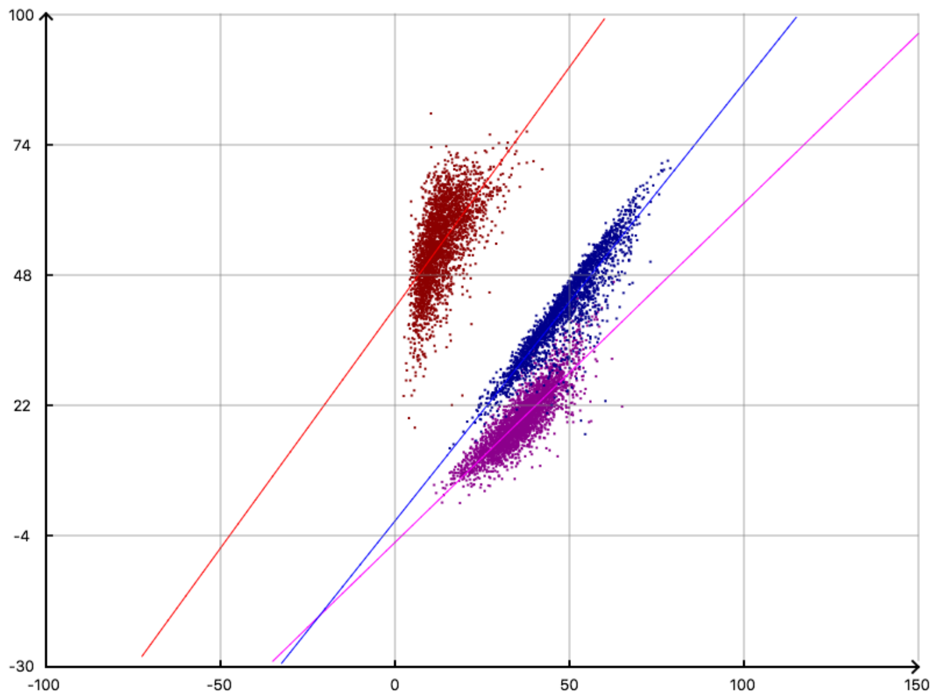


Figure 31: This figure represents the result provided by this program when the constraints were changed.

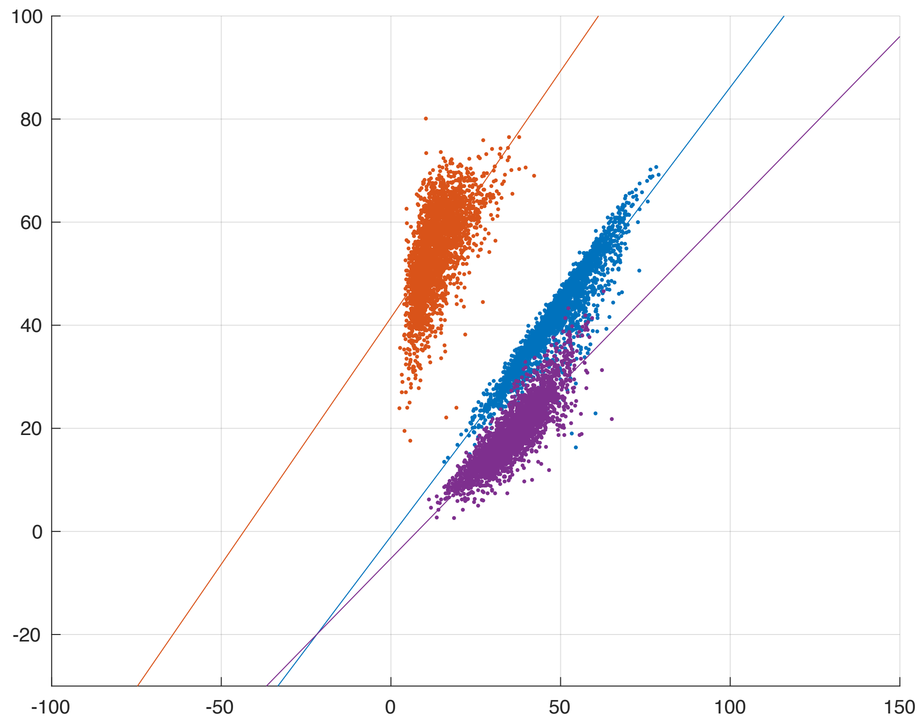


Figure 32: This figure is constructed with MATLAB.

Figures 31 and 32 represent the same data set and same constraints. Figure 31 was created with this program and 32 with MATLAB. They both look very similar, so we can conclude that the changing constraints property works properly. This kind of analysis was covered with many data-sets and many different constraint values.

Other aspects of plotting were analyzed besides the constraints. During the building process, especially the grid and legend had to be tested. For example, the regression models and data-points should not be plotted above the legend box. This is illustrated in figure 14, where the blue curve goes under the legend box. Different data-points sizes had to be tested too, which was also done visually. Other aspects, such as legend texts and label texts, were tested by other users, and many problems were discovered. Other users also tested to change the constraints to identify any issues with the program.

5.4 User interface

The tests performed in sections 5.1, 5.2 and 5.3 were mostly done with the user interface. However, testing the user interface is crucial since it is the only method to experience the whole program. Testing the user interface was done by many different users trying to visualize and analyze the data. At the same time, trying to make the program crash is important in exploring the possible mistakes. The performance of the program was tested with different files (txt, xlsx and csv), many of them containing large amounts of data points. Information about the test files provided with this project is presented in appendix A. The visualization of the data was tested with numerous different inputs by many people. Most of the problems were found, but the program still suffers from some known bugs and weaknesses that are discussed in sections 6.1 and 6.2.

6 Evaluation

This section provides the overall evaluation of the program and the process of building the application. This program met all of the requirements given by the assignment for the advanced version. However, some weaknesses are part of any project, but some decisions have also influenced the structure and the motivation for building specific parts in a particular manner. Known bugs and missing features are described in section 6.1. Three of the strengths and weaknesses are discussed in section 6.2. Deviations from the plan and the final realized schedule of the project is expressed in section 6.3. The final evaluation of the program and building the application is assessed in section 6.4.

6.1 Known bugs and missing features

Even though strict testing was performed, the final program has some bugs and weak features. The table 5 expresses the known bugs, and the table 6 describes the missing features.

Table 5: This table describes some of the known bugs in the program.

Importing empty txt or xlsx files cause errors.
Incorrectly formed txt files are not handled ideally by the program.
The user is able to produce weird looking plots.
While importing a csv or xlsx file, the user must click the OK button twice.
The program only plots the coordinate system if the file contains one data-point.
If the file contains y-values, where $y \leq 0$, no exponential model is formed.
If the file contains x-values, where $x \leq 0$, no logarithmic model is formed.
If too large data-sets are imported, the program will just run for a long time.

Table 6: This table describes some of the missing features of the program.

Zooming in is only possible to a certain degree.
The range of the constraints has to be divisible by five.

The most problematic errors are related to the import of the data-sets. Text- and excel-files cause failure if they are empty, meaning that the user interface will not inform the user about the problem. The program also does not handle the incorrect lines in the text file ideally as it replaces every wrong line with 0.00000 0.00000, for example er45..34,45→0.00000 0.00000. However, this does not cause the program to crash, and erroneous data-points should be filtered out by the user before creating regression models. If any of the data-points were changed, the program would inform the user about it.

The user is also able to produce weird looking plots by choosing absurd constraints, long label texts, or long legend texts; these are just examples. However, most of the time, to produce weird plots that has to be the motivation, and an example of such a bizarre plot is presented in figure 33.

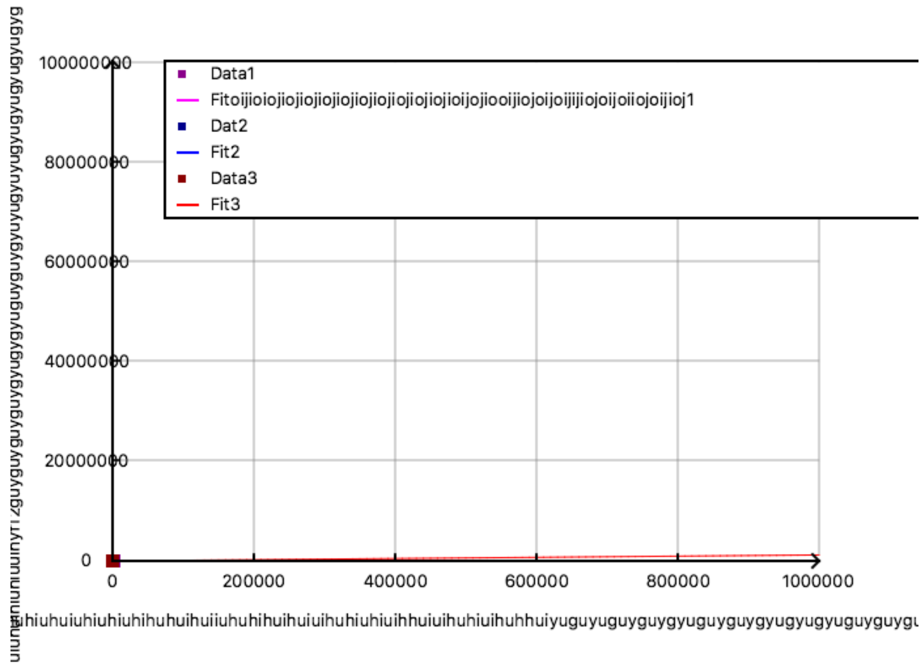


Figure 33: The user is able to produce weird looking plots.

Table 5 also mentions about a bug where the user must click the OK button two times when importing csv or xlsx files. However, this does not reduce the usability of the program or cause any errors. The program is also unable to produce exponential models if the data-set contains y-values, where $y \leq 0$, and logarithmic models if the data-set contains x-values, where $x \leq 0$. However, this is logical because such values cannot be achieved by these models. The program will not produce any error messages for the user to inform about this problem.

The program also experiences problems if the user imports data-sets that are too large. In this case, the program would just run for a long time, and it would not inform the user about it. The times to read and plot data are illustrated in figure 21. In addition to some bugs, this program also has some missing features.

The missing feature is related to changing the constraints of the axes. The user cannot choose any range as the range must be divisible by five. This is due to the grid, and illustrative numbers next to the axes since the number of these indicative numbers is fixed to five per axis. There are two reasons why this problem arises: the fixed number of the illustrative numbers is five, and the illustrative numbers only support integers.

This problem could be solved by removing the illustrative numbers and only placing numbers at the end of the axes. However, for data-analysis and good-looking plots, this is not a great solution. The numbers could also support double types, but that way, some ugly looking numbers like 15.3456237 might appear, and the numbers might not even be exact.

The best solution would be to give support for different amounts of illustrative numbers (such as 2,3,4,5...). This would have made the building process a lot more time consuming, and the issue behind the reason to spend that much time on something would not have been wise.

The other missing feature relates to constraints as well. The zooming is restricted to some extent as the final range of the constraints has to be divisible by five. If the x-values of the data-set lie between [0,10], the smallest range the user can plot is [0,5] or any other range that has a length of five. The data set is always mapped to have a range of at least ten, if the original range was smaller. For example, if the x-values lie between [0,1], they are mapped into the range [0,10], and a correction term 10^{-1} will appear in the x-axis, meaning that the user can, for example, plot values between [0,0.5].

The zooming in feature could be fixed by removing the other illustrative numbers or by changing the indicative numbers from Int to Doubles. Zooming into the data-set is not common when analyzing regression and data since information is lost by zooming into the figure. The lost data-points are relevant data, and if serious data-analysis were done, such actions would not be reasonable; thus, the zooming in option is not that important. Extrapolating data to see, for example, the future trends is an essential part of data-analysis, making zooming out more critical, and zooming out is not restricted by this program. The missing features of this program could be fixed, but I find it more important to have illustrative numbers next to the axes to help with data analysis.

6.2 3 best sides and 3 weaknesses

This section discusses the three weakest and three best sides of the program. The weakest sides might not be bad, and the best sides might not be excellent, but this section gives a general overview of these weakest and best sides of the program. Three of the best and three of the weakest sides are listed in table 7.

Table 7: This table presents the best and weakest sides of the program.

Best	**Weakest**
Functionality	Program structure at some points
User interface	The quality of code at some places
Testing	Some error situations

The program functions correctly. Importing different data-sets is possible, and most of the time, clear alerts are presented for the user if something was done incorrectly. The plotting is easy, and before plotting, numerous features of the plot can be modified, including constraints, grid, legend, and data-point size. The plotted data-sets, together with the regression curves, look beautiful and correct. Some additional functionalities were added to the program, which makes the program better. Different data-sets can be plotted at the same time, which is excellent for data comparison. The user can also retrieve the information about the regression models, which is useful for precise data analysis.

The user interface is simple and easy to use, and it is logically constructed. Importing data-sets is fast, and only couple of clicks are needed. Modifying plotting features is trivial and creating beautiful looking plots is effortless. For example, creating the plot in figure 26 took under a minute to construct while the figure 27 took around 15 minutes to assemble with MATLAB. This program creates regression plots explicitly while MATLAB is more complex, thus, creating specific plots with MATLAB is not necessarily easy.

The testing of this program was done professionally by comparing the results with MATLAB. Plotting, parameter modifications, and regression models of this program were easy to compare with the results of MATLAB. Many different data-sets, parameters, and regression models were tested to conclude that the program worked properly.

Different data-sets created by humans and programs were also tested. The data-sets created by simulation tools, such as Comsol, always seemed to work correctly. The incorrect type of data-sets usually has a human kind of error in it. Human error was tested by giving the program to multiple users to check correct inputs together with incorrect and even absurd ones. Examining the overall usability of the program was tested by providing the program to various users, and all of the sub-parts of using the program were tested: starting the program, importing files, plotting, and modifying features. More information about the testing is presented in section 5.

The weakest sides of the program are the overall structure, quality of the code, and handling some erroneous situations. The whole structure of the code is presented in figure 20. The structure of the regression classes is logical, and the structure is justified. However, the overall structure of the program is messy, and the connections between different classes are not clear. The individual classes usually have a logical structure, but the connections are messy. The use of public variables is not justified in some cases, which could lead to errors if individual classes or the overall program was edited.

The overall quality of the code is not perfect. Classes Legend and CoordinateSystem look ugly, and the code is repetitive. These classes, however, introduce some extra features that make the usability of the program more enjoyable. These classes should be constructed in a more logical and easier manner. The classes Regression, Data, Plot, and UI also have some repetitive code in them, but not as much. The visibility of the variables and methods within these classes is not well justified. These classes also include some useless variables and methods. The code is, however, mostly understandable and well commented. The overall structure of the program should be modified, and different approaches to decrease the repetition of code should be considered.

In addition, the program has problems handling the erroneous data correctly. This gives the user additional responsibilities to check that the data that is imported is formed correctly. The application should, however, operate properly if data-sets produced by programs were used, and if the user is aware of the correct inputs. Creating correct data is not difficult. The program is, in most cases user friendly, meaning that it will provide error messages, but in some cases, for example, importing an empty excel file, no error messages are presented.

The weakest features about the code and the program are not horrible, but some of them should be improved considerably to make, for example, the modifications of the application easier. In addition to the three best sides of the program, the chosen data structures, algorithms, and the justifications about the features have been carefully studied, and the preferred methods, together with solutions, are valid and justified.

6.3 Deviations from the plan, realized process and schedule

The final program closely reassembles the plan presented in section 2.1. The idea was to build the advanced version of the program given in the assignment, and the final result provides all of the required features, in addition, to some extra features to make the program more enjoyable, practical, and easier to use.

The planning of the original schedule is presented in table 8, but this schedule was not followed closely as the deadlines in other courses influenced the order and the timing of the process.

Table 8: The working hours and the progress order are presented in this table.

Week	Theme	Time estimate
8	Linear regression	5h
9	Reading data	10h
10	Plotting and testing	5h
11	User interface	10h
12	User interface and testing	10h
13	Exponential reg and plotting	5h
14	User interface, finishing, testing	10h
15	Finishing, testing	10h
16	Job done	sum=65h

Since I had so many other courses, including writing the Bachelor's thesis, it was almost impossible to plan the schedule for this course. For this reason, this project was promoted at all times; there was time for it. The final realized schedule is represented in table 9.

Table 9: The working hours and the progress order are presented in this table.

Week	Theme	Time estimate
8	Regression models	2h
8	Reading data	2h
8	Plotting and testing	5h
8	Simple user interface	10h
8	Testing plotting	6h
9	Testing everything	20h
9	Creating better user interface	15h
10	Fixing bugs and testing	5h
11	Fixing bugs and testing	5h
14	Fixing bugs and testing	2h
15	Fixing bugs and testing	2h
15	Writing the report	40h
15	Errors with different computers	16h
16	Job done	sum=120h

The realized schedule differs from the original plan massively as the working load of some of the tasks was overestimated, and some of them were underestimated. Building the actual program was more straightforward and faster than expected, but the real testing and fixing the program was hard and took a lot of time. The program was probably built too fast due to the other courses, and the overall structure of the individual classes and the connection between the classes and variables were not thought throughout and carefully enough, thus, making the modifying process difficult. I also forgot to include the construction of the final report, which ended up being laborious as the course homepage stated that the final report should reassemble *"professional high quality"*.

6.4 Final evaluation

This report explained the program together with the process of building the application. The overall structure of the program closely followed the plan presented in section 2.1. Still, some extra features were added during the process of building the program to make the program more advanced, enjoyable, and usable.

The advanced version of the program was built, and numerous additional features, including plotting multiple data-sets at the same time for comparison, were added to make the program a useful tool for precise data analysis. Some properties, such as zooming in to the figure and choosing exact constraints, are not entirely supported by the program, but the justifications behind these decisions are addressed, for example, in sections 6.1 and 6.2.

The motivation of this project was to create a visualization tool for data-sets and regression models. This program supports three types of regression models: linear, exponential and logarithmic. Data-sets, consisting of two-dimensional data-points, can be easily imported, and analyzing the data-sets together with the constructed regression models is enjoyable with the plotting tool where numerous modifications for the plot are supported. All of these features together form a program that corresponds to the motivation of the project. The final program is an easy tool to create plots together with regression models, and precise data-analysis is also possible. All of the advanced requirements are also met.

However, the program has some weaknesses and missing features. The overall structure of the program is messy ad the connections between classes should have been considered more carefully. The style of the code is repetitive at times, and the use of variables and methods, together with the visibility, is not justified every time. The program does not always handle erroneous data correctly, which gives some additional responsibilities for the user of the program.

The main missing features about the program are related to the constraints of the axes. The user is not able to choose precise constraints for the axes as the range of the axes has to be divisible by five. The user cannot zoom into the figure too much as the axes only support whole numbers. These missing features could have been added by removing the illustrative numbers from the axes and adding support for decimal numbers.

These fixes, however, would not have made the program better, but they would have reduced the usability of the program. The illustrative numbers help the user to visualize the data, and the decimal numbers are not precise, they can also look unsightly. The zooming in option is not essential in data-analysis as information is lost by removing data-points from sight. Extrapolating data is more important to visualize the future trends of the data. This program is capable of showing the data-set together with the regression model, and slight modifications for the plot can be performed by the user. Multiple data-sets can be analyzed simultaneously, and the information behind the regression models is provided by the program, making this program perfect for precise data analysis. All of the justifications of the missing features are described in section 6.1 and 6.2.

The usability of this program is simple, and the functionality is fast and correct. The user interface is easy to use, and generating beautiful plots and analyzing data is quick and straightforward. The program calculates the regression models correctly, and the plots provided are wonderful. There are a lot off possible modifications and additional features for the plot, which are all easy to use.

The program will probably be modified and extended in the future. Changing the visual aspect of the plot, such as the legend and the coordinate system, should be easy. Adding new regression models is trivial if the algorithm behind the model is known. The overall structure of the program is messy, making small adjustments behind the whole process a bit laborious, however, adding new extensions should be easy: new features to the plot, new regression models, new pages to the interface and new file supports. The overall result of this program is excellent, and I have already used it in any of my own projects.

A Test files

Table 10: This table shows the information about the test files provided.

File	**Regression**	**Columns**	**Rows**
50000.csv	Linear	2	50000
cancer_reg.csv	Linear	35	3048
exp.txt	Exponential	2	12
exponentialSmall.txt	Exponential	2	12
exponentialSmally.txt	Exponential	2	12
italy.txt	Exponential	2	35
linear.txt	Linear	2	46
linear2.txt	Linear	2	46
linearBig.txt	Linear	2	4
linearTest.xlsx	Linear	2	10005
log.txt	Logarithmic	2	14
logSmall2.txt	Logarithmic	2	14
manaus.csv	Linear	3	1081
usa.txt	Exponential	2	35
wool.csv	Linear	3	310

The correct format of the txt-files is provided below (exp.txt). Each of the data-points have to be on a separate line and the values of the data-points are separated with spaces.

```
0.0 889.0
100.0 692.0
200.0 572.0
300.0 426.0
400.0 357.0
500.0 253.0
600.0 217.0
700.0 185.0
800.0 152.0
900.0 125.0
1000.0 108.0
1100.0 78.0
```

B Plot example

To produce the plot presented in figure 4 the user has to import the files represented in table 11.

Table 11: This table shows the files to be imported to recreate the figure 4.

File	**Regression**	**Columns**
cancer_reg.csv	Linear	21,22
cancer_reg.csv	Linear	25,26
cancer_reg.csv	Linear	27,28

After importing the files described in table 11, the user must select the following modifications presented in tables 12 and 13.

Table 12: This table shows some of the modifications to produce the plot in figure 4.

x-axis start	**x-axis end**	**y-axis start**	**y-axis end**	**Grid on**	**Legend on**
0	100	0	100	Yes	Yes

Table 13: This table shows some of the modifications to produce the plot in figure 4..

Legend text	**Plot size**	**Datapoint size**	**Line width**
Data1, Fit1, Data2, Fit2, Data3, Fit3	Medium	2	1

References

[1] Linear Regression. Brilliant.org. https://brilliant.org/wiki/linear-regression/ Retrieved: February 10, 2020

[2] Weisstein, Eric W. "Least Squares Fitting–Exponential." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/LeastSquaresFittingExponential.html Retrieved: February 10, 2020

[3] Weisstein, Eric W. "Least Squares Fitting–Logarithmic." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/LeastSquaresFittingLogarithmic.html Retrieved: February 10, 2020