

University of Wales Trinity Saint David

School of Engineering



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

Engineering Practice and Study Skills – Technical Report

Analysis of Student Grades using Statistical Methods in

Matlab ACMA4007

Meuryn Nelson

2308099

20/05/24

Table of Contents

1	Introduction	1
1.1	Aims	1
1.2	Objectives	1
2	Main Body	2
3	Conclusions	14
4	References.....	15
5	Appendix	16

Figure 1 Data of students and their exam results in Microsoft Excel.....	2
Figure 2 Code in Matlab showing how to import data from a current folder.....	2
Figure 3 Code in Matlab showing how to work out the overall exam results for each student	3
Figure 4 Code in Matlab showing how to separate numerical data into a matrix	3
Figure 5 Code in Matlab showing how to separate numerical data into different variables	3
Figure 6 Workspace in Matlab showing all the different tables of data created from the code inputted	3
Figure 7 Code in Matlab showing how to create a table from the variables inputted	4
Figure 8 Table results from the code entered for the variables	4
Figure 9 Code in Matlab for corrplot function	4
Figure 10 A correlation matrix created in Matlab using corrplot code	5
Figure 11 Code in Matlab showing how to group the data into the two different genders	6
Figure 12 Code in Matlab for the ttest2 function	6
Figure 13 Hypothesis results from the ttest2 shown in the command window in Matlab.....	6
Figure 14 Code in Matlab using the histcounts function dictating how the histograms will be presented	7
Figure 15 Code used to create a histogram for the results in Exam 1	8
Figure 16 Histogram visualising numerical data for the results in exam 1	8
Figure 17 Code used to create a histogram for the results in Exam 2	9
Figure 18 Histograms showing exam 1 and 2 results next to each other, great for comparisons	9
Figure 19 Code to create the histogram showing overall exam results	10
Figure 20 Histograms showing the exam results from exams 1 and 2 next to the overall exam results	10
Figure 21 Code in Matlab to describe how the ogive graphs will appear.....	11
Figure 22 Code in Matlab to create an ogive for exam 1 results	11
Figure 23 Code in Matlab to create an ogive for exam 2 results	11
Figure 24 Code in Matlab to create an ogive for the overall exam results	12

1 Introduction

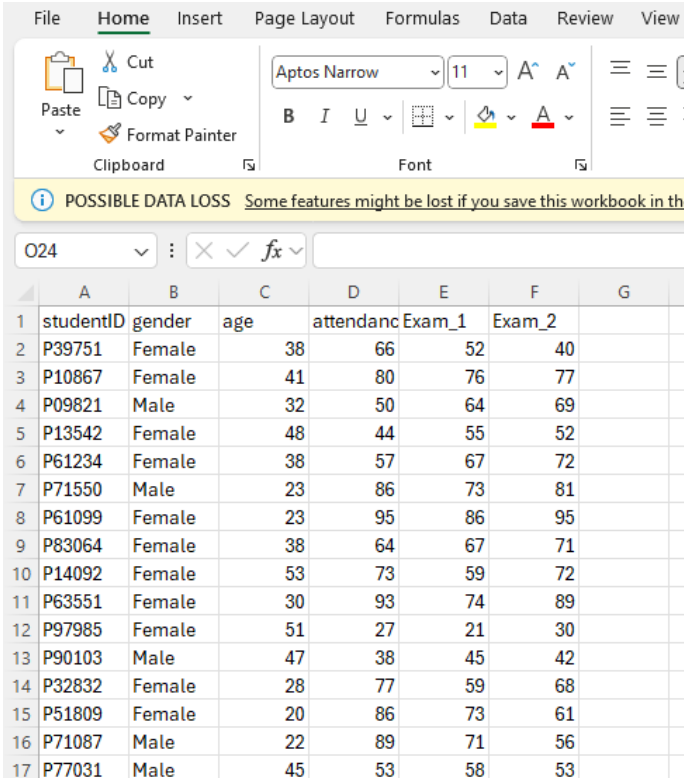
1.1 Aims

The aim of this report is to show the reader how the coding software Matlab can be a powerful tool for analysing statistical data, specifically in this case, data from Microsoft Excel on exam results from students.

1.2 Objectives

The objective is to import student exam results data from Microsoft Excel into Matlab and write various lines of code to gain summary statistics, perform a correlation analysis and visualise data. This report will show the reader how Matlab can give an in-depth analysis from data using basic functions within the software.

2 Main Body



The screenshot shows the Microsoft Excel interface with the 'Home' tab selected. A yellow warning bar at the top indicates 'POSSIBLE DATA LOSS: Some features might be lost if you save this workbook in the current format'. The active cell is O24. The data is organized in a table with the following structure:

	A	B	C	D	E	F	G
1	studentID	gender	age	attendance	Exam_1	Exam_2	
2	P39751	Female	38	66	52	40	
3	P10867	Female	41	80	76	77	
4	P09821	Male	32	50	64	69	
5	P13542	Female	48	44	55	52	
6	P61234	Female	38	57	67	72	
7	P71550	Male	23	86	73	81	
8	P61099	Female	23	95	86	95	
9	P83064	Female	38	64	67	71	
10	P14092	Female	53	73	59	72	
11	P63551	Female	30	93	74	89	
12	P97985	Female	51	27	21	30	
13	P90103	Male	47	38	45	42	
14	P32832	Female	28	77	59	68	
15	P51809	Female	20	86	73	61	
16	P71087	Male	22	89	71	56	
17	P77031	Male	45	53	58	53	

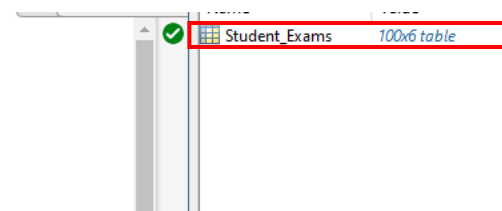
Figure 1 Data of students and their exam results in Microsoft Excel

The above Figure1 shows part of the data from the Microsoft Excel spreadsheet provided, showing studentID, gender, age, attendance and their exam results over 2 different exams. The first step is to import this data into Matlab by using the function called 'readtable'. This function creates a table, detects variables and more from data such as a spreadsheet. **(MathWorks)**

```

1  clc
2  clear
3  close all
4
5  %Task1 - Import data and summary statistics
6
7  %Import provided dataset into Matlab
8  Student_Exams = readtable("student_grades.xlsx");
9

```



The screenshot shows the MATLAB variable viewer with a table named 'Student_Exams' highlighted. The table is described as a '100x6 table'.

Student_Exams	100x6 table
---------------	-------------

Figure 2 Code in Matlab showing how to import data from a current folder

The next step is to calculate the overall exam results for the students. To do this the results from exam 1 and exam 2 need to be added together than divided by 2.

```
%Calculate overall exam results and add to summary statistics
Student_Exams.Overall = (Student_Exams.Exam_1 + Student_Exams.Exam_2) / 2;
```

Figure 3 Code in Matlab showing how to work out the overall exam results for each student

After this has been done, the numerical data needs to be separated into a matrix.

```
%Separate numerical data into a matrix
numerical_data = table2array(Student_Exams(:, 3:end));
```

Figure 4 Code in Matlab showing how to separate numerical data into a matrix

Following on from this, analysis of the mean, median, mode, range and standard deviation can be undertaken. The code below shows how to do that.

```
%Find the mean, median, mode, range and standard deviation from numerical data
mean_matrix = mean(numerical_data);
median_matrix = median(numerical_data);
mode_matrix = mode(numerical_data);
min_matrix = min(numerical_data);
max_matrix = max(numerical_data);
range_matrix = range(numerical_data);
std_matrix = std(numerical_data);
```

Figure 5 Code in Matlab showing how to separate numerical data into different variables

The software does its calculations and works out the variables from the numerical data inputted and creates separate tables for each one, in which you can access in the Workspace section in Matlab.

Workspace	
Name ▲	Value
 max_matrix	[53,100,98,98]
 mean_matrix	[36.6500,60.5800,58.2...
 median_matrix	[37.5000,57,58,53.5000]
 min_matrix	[19,20,7,4]
 mode_matrix	[48,27,59,51]
 numerical_data	100x4 double
 range_matrix	[34,80,91,94]
 std_matrix	[9.7643,24.2492,21.67...
 Student_Exams	100x6 table

Figure 6 Workspace in Matlab showing all the different tables of data created from the code inputted

After the separation of the numerical data – it allows you to create a table from the statistics in gathered for each variable using the code below.

```
%Display summary statistics in table format
var_names = {'Age'; 'Attendance'; '1st exam score'; '2nd exam score'; 'O/all exam score'};
col_headings= {'variables'; 'mean'; 'median'; 'mode'; 'range'; 'std'};

summary_table = table(var_names, mean_matrix', median_matrix', mode_matrix', range_matrix', std_matrix', 'VariableNames', col_headings);

disp('Summary Statistics:');
disp(summary_table);
```

Figure 7 Code in Matlab showing how to create a table from the variables inputted

This generates a table in the command window which is a great visual for numerical data. As seen in the below table, the mean results show us that students generally performed better in exam 1 than exam 2. It also clearly shows that the difference between the highest and lowest scores overall was 85.5%.

Summary Statistics:						
variables	mean	median	mode	range	std	
{ 'Age' }	36.65	37.5	48	34	9.7643	
{ 'Attendance' }	60.58	57	27	80	24.249	
{ '1st exam score' }	58.27	58	59	91	21.673	
{ '2nd exam score' }	55.08	53.5	51	94	22.428	
{ 'O/all exam score' }	56.675	54.5	25.5	85.5	21.452	

Figure 8 Table results from the code entered for the variables

Using the data so far in Matlab, a correlation analysis can be performed using a function called 'corrplot'. "Pearson's correlation coefficients between all pairs of variables in the input matrix of time series data. " (MathWorks)

```
%Use corrplot to visualise a correlation matrix
figure('Name', 'Correlation Coefficient');
var_names_corr = {'Age'; 'Attendance'; 'Exam1'; 'Exam2'; 'Overall'};
corrplot(numerical_data, 'varNames', var_names_corr, 'TestR', 'on');

% Display the correlation matrix
disp('Correlation Matrix:');
```

Figure 9 Code in Matlab for corrplot function

The code above will create a correlation matrix using the variables and entered. A correlation matrix is a table that will display the correlation coefficients for all the different variables that have been included in it. This can be a powerful tool for identifying patterns in large datasets.

<https://corporatefinanceinstitute.com/resources/excel/correlation-matrix/> (CFI Team at CorporateFinanceInstitute)

Below shows the correlation matrix from all the variables inputted, below.

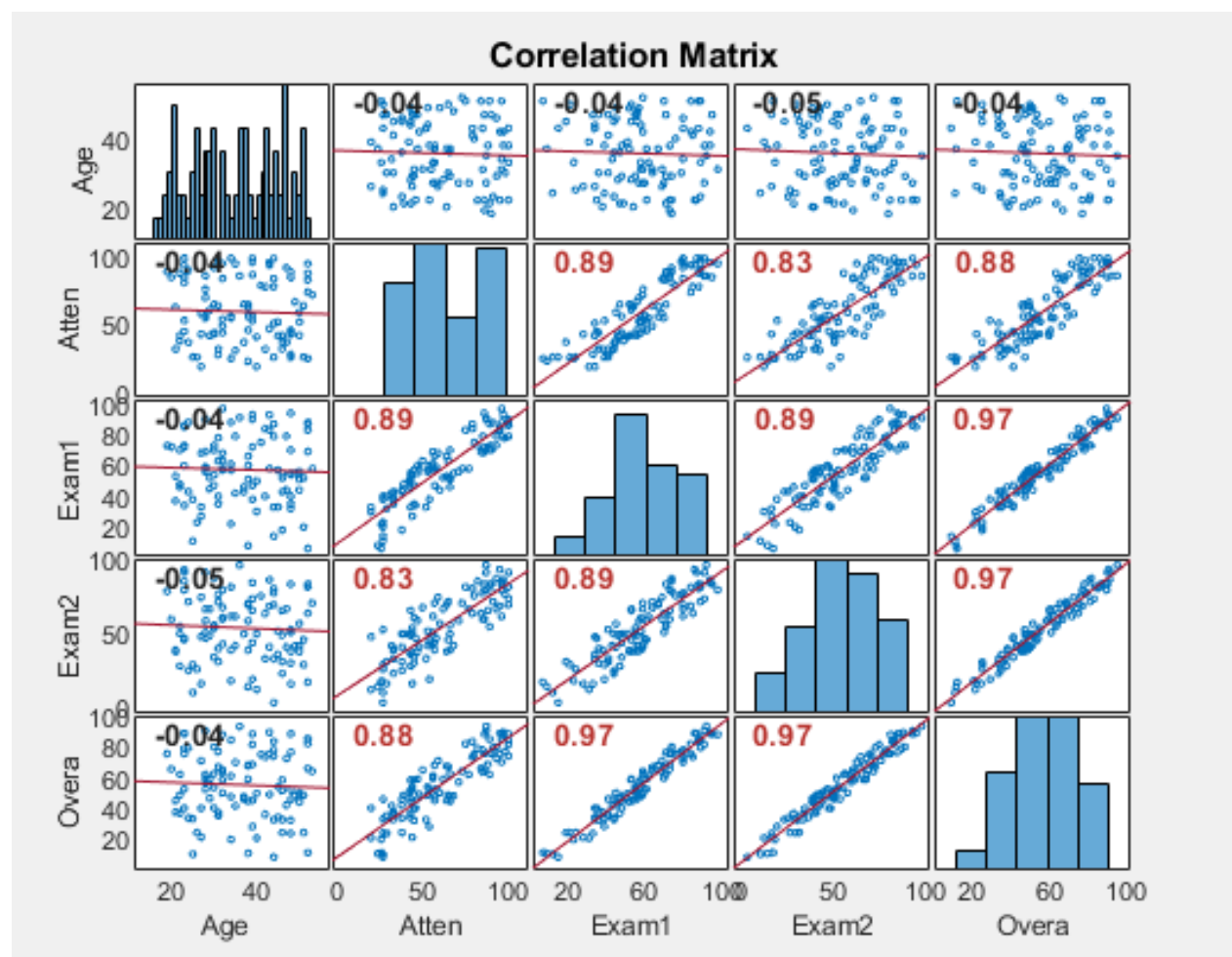


Figure 10 A correlation matrix created in Matlab using corrplot code

A relationship between gender and overall exam results can be made using the numerical data that Matlab has. The next stage is to organise the numerical data into gender and overall exam scores, to compare the results for male and female. The following code allows that.

```
%Organise numerical data by gender and overall exam results
gender_Score = table(Student_Exams.gender, Student_Exams.Overall, 'VariableNames', {'Gender', 'Overall'});

%Group data by gender
Male_score = gender_Score(strcmp(gender_Score.Gender, 'Male'), 'Overall');
Female_score = gender_Score(strcmp(gender_Score.Gender, 'Female'), 'Overall');
```

Figure 11 Code in Matlab showing how to group the data into the two different genders

To test this hypothesis, a function called 'ttest2' can be used. MathWorks' description of this function is *"returns a test decision for the null hypothesis that the data in vectors x and y comes from independent random samples from normal distributions with equal means and equal but unknown variances, using the two-sample t-test."* **(MathWorks)**

Code must be provided for ttest2 to work its calculations which can be found below.

```
%Use ttest2 to test hypothesis
[h, p] = ttest2(Male_score, Female_score);

%Display the results
if h == 0
    hVal_Str = "shows that the means are roughly equal based on the data provided";
else
    hVal_Str = "shows that the means are not roughly equal based on the data provided";
end

if p > 0.05
    pVal_Str = "exceeds an alpha of 0.05, showing insufficient evidence to reject the null hypothesis (h0)";
else
    pVal_Str = "falls below an alpha of 0.05, showing insufficient evidence to reject the null hypothesis (h0)";
end

fprintf("The hypothesis test gave a h value of = %i %s\n", h, hVal_Str);
fprintf("The hypothesis test gave a value of p = %f %s\n", p, pVal_Str);
```

Figure 12 Code in Matlab for the ttest2 function

The 'fprintf' function as seen in the above code is what dictates the text output in the command window. The outcome of the hypothesis can be seen below.

```
Correlation Matrix:
The hypothesis test gave a h value of = 0 shows that the means are roughly equal based on the data provided
The hypothesis test gave a value of p = 0.437006 exceeds an alpha of 0.05, showing insufficient evidence to reject the null hypothesis
```

Figure 13 Hypothesis results from the ttest2 shown in the command window in Matlab

Histograms are a great visual tool for analysing data, used industry-wide when dealing with data. They show data results in a bar graph form making them easy to decipher – extremely useful for reading and sharing data findings. Matlab can easily create histograms with coding, showing data from exams 1 and 2 and the overall exam results.

Firstly, you can enter code into Matlab which will dictate the appearance of the histogram bar graphs when they appear in a separate tab. This also dictates how much data is to be shown using a function called 'histcounts'. The code below shows this.

```
% Use histcounts to calculate frequency
[E1_count, E1_edges] = histcounts(numerical_data(:, 3), 'BinWidth', 10);
[E2_count, E2_edges] = histcounts(numerical_data(:, 4), 'BinWidth', 10);
[Overall_count, Overall_edges] = histcounts(numerical_data(:, 5), 'BinWidth', 10);
```

Figure 14 Code in Matlab using the histcounts function dictating how the histograms will be presented

After describing the histograms using histcounts – subplots need to be coded in for each individual histogram that is to be shown. A separate one for exam 1, exam 2 and the overall results.

Below is the code used to create the histogram for the results from exam 1.

```
% Histogram Exam 1
figure('Name', 'Histograms');
subplot(1, 3, 1);
histogram('BinCounts', E1_count, 'BinEdges', E1_edges);
title('Exam 1 Histogram');
xlabel('Class limits');
ylabel('Frequency');
ylim([0, 22]);
grid on;
```

Figure 15 Code used to create a histogram for the results in Exam 1

As seen in the below histogram, the bulk of students (21 of them) had just over the 50% mark in exam 1.

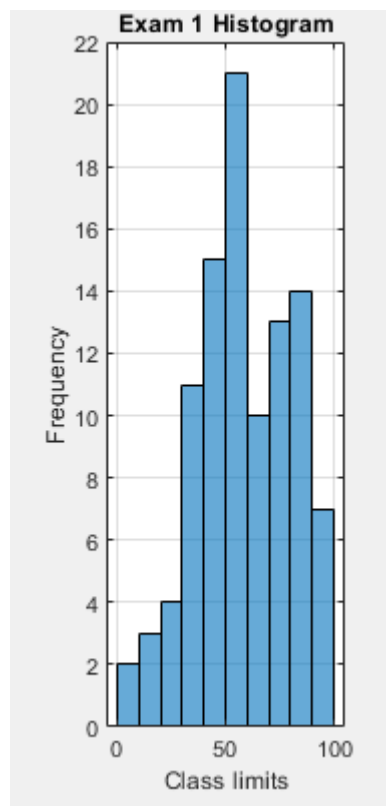


Figure 16 Histogram visualising numerical data for the results in exam 1

Below is the code used to create the histogram for the results from exam 2.

```
%Histogram Exam 2
subplot(1, 3, 2);
histogram('BinCounts', E2_count, 'BinEdges', E2_edges);
title('Exam 2 Histogram');
xlabel('Class limits');
ylabel('Frequency');
ylim([0, 22]);
grid on;
```

Figure 17 Code used to create a histogram for the results in Exam 2

After entering the code for the exam 1 histogram and running the script, it showed us the histogram results for exam 1 alone. The code for exam 2 is then entered next and when the script is run, it will show the histogram graph for both exam 1 and 2 together, due to how it was described earlier on.

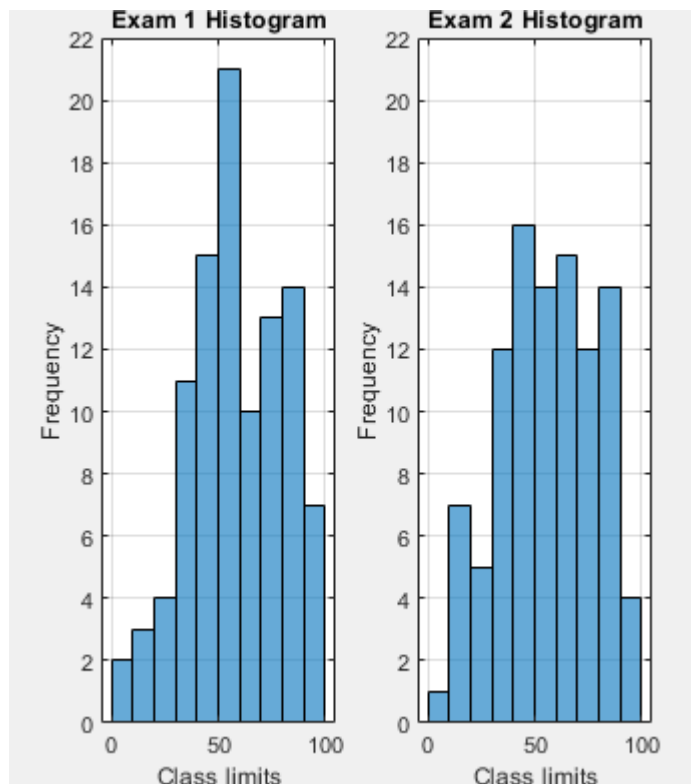


Figure 18 Histograms showing exam 1 and 2 results next to each other, great for comparisons

Having the 2 histograms next to each other is brilliant for analysing data and comparing the values from exams 1 and 2. Exam 1 shows us that the majority of students were around the 50% mark. Exam 2 shows us that the scores from students are more spread across from 0 to 100%, with a lot of the bar spikes being close to level.

After the creation of histograms for the exam results in exams 1 and 2, a final histogram can be made of the overall results. The code for that is below.

```
%Histogram Overall Exam Results
subplot(1, 3, 3);
histogram('BinCounts', Overall_count, 'BinEdges', Overall_edges);
title('Overall Exam Results Histogram');
xlabel('Class Limits');
ylabel('Frequency');
ylim([0, 22]);
grid on;
```

Figure 19 Code to create the histogram showing overall exam results

The below shows us the 3 different histograms next to each other which again is a fantastic visual for analysing data.

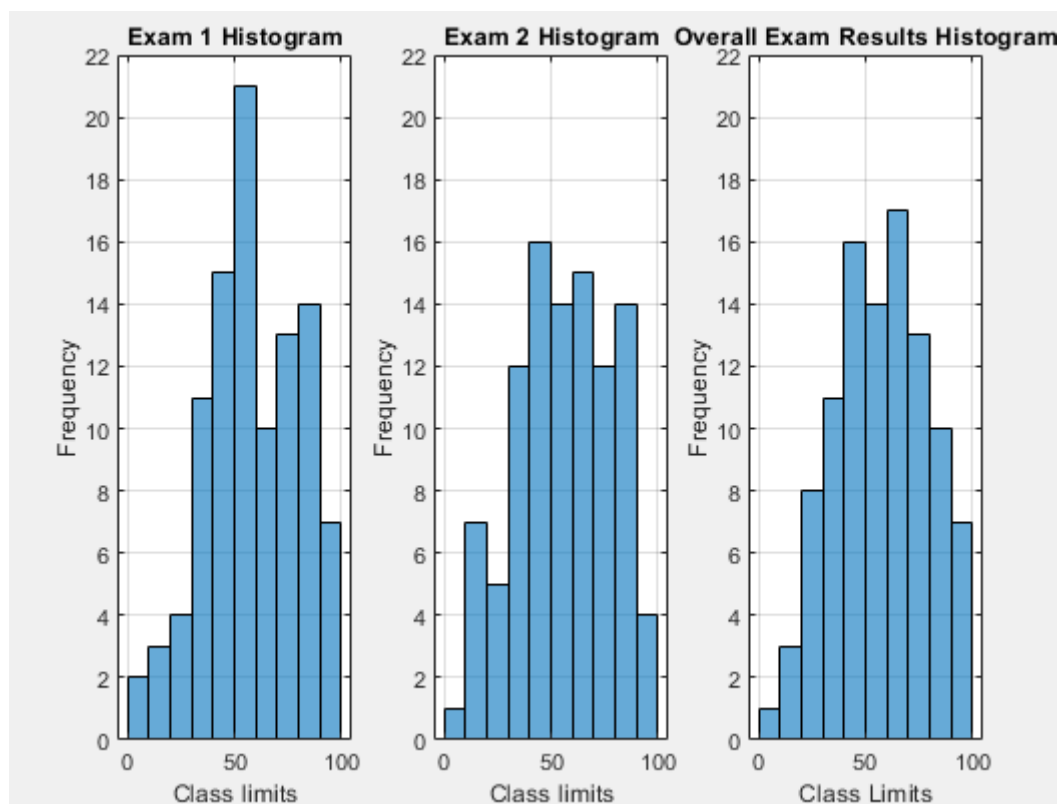


Figure 20 Histograms showing the exam results from exams 1 and 2 next to the overall exam results

Another visual tool that Matlab can easily create is ogives. The code below shows one way that you can ask Matlab to create an ogive graph.

```
%Ogives
figure('Name', 'Ogives');

Exam1_LessThan = [0, cumsum(E1_count / sum(E1_count))];
Exam1_MoreThan = 1 - Exam1_LessThan;

Exam2_LessThan = [0, cumsum(E2_count / sum(E2_count))];
Exam2_MoreThan = 1 - Exam2_LessThan;

Overall_LessThan = [0, cumsum(Overall_count / sum(Overall_count))];
Overall_MoreThan = 1 - Overall_LessThan;
```

Figure 21 Code in Matlab to describe how the ogive graphs will appear

Just like with histograms – subplots need to be coded for each individual ogive graph. The code for the exam 1 ogive is seen below.

```
subplot(1, 3, 1);
plot(E1_edges, Exam1_MoreThan, 'b.-', E1_edges, Exam1_LessThan, 'r.-');
title("Exam 1 Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(E1_edges);
grid on;
```

Figure 22 Code in Matlab to create an ogive for exam 1 results

```
subplot(1, 3, 2);
plot(E2_edges, Exam2_MoreThan, 'b.-', E2_edges, Exam2_LessThan, 'r.-');
title("Exam 2 Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(E2_edges);
grid on;
```

Figure 23 Code in Matlab to create an ogive for exam 2 results

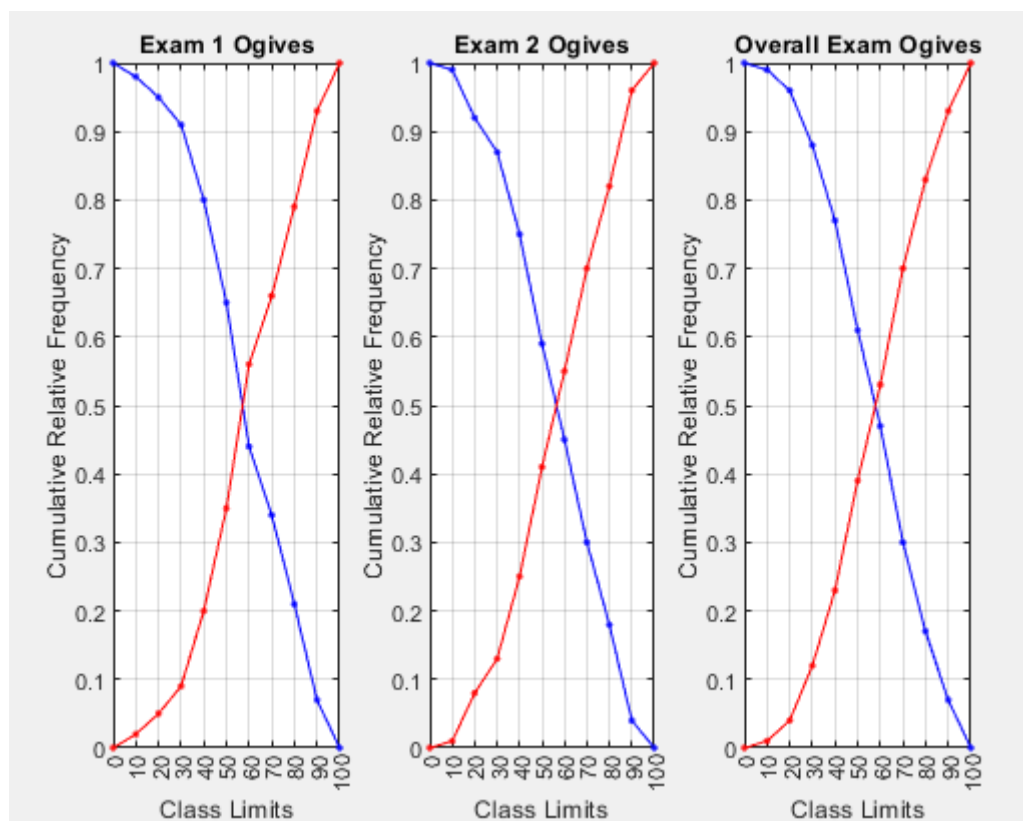
```

subplot(1, 3, 3);
plot(Overall_edges, Overall_MoreThan, 'b.-', Overall_edges, Overall_LessThan, 'r.-');
title("Overall Exam Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(Overall_edges);
grid on;

```

Figure 24 Code in Matlab to create an ogive for the overall exam results

Ogive graphs are great in less than and more than scenarios when visualising data. Below shows the 3 different ogive graphs created with the code entered.



That is the code complete. The powerful thing about using Matlab for this use is that this code can be saved for future uses. When students take exams in the future and their results are recorded in Microsoft Excel – all that needs to be changed is the imported data. The script can then be run and will produce everything that it has for this dataset on the new dataset. The correlation matrix, table, hypothesis test, histograms and ogives will all appear within seconds showing the analyser visual data from the new exam results.

3 Conclusions

This report shows an in-depth analysis of how powerful a tool Matlab can be when analysing data. The script of code has taken a simple Microsoft Excel spreadsheet and broken it down into various sections, showing an array of visual data for people to read. This code can then be used to extract visual data for all exam results in the future with a very minor tweak.

To conclude – Matlab can be extremely useful for anyone performing data analysis.

4 References

CorporateFinanceInstitute. “Correlation Matrix.” *Corporate Finance Institute*,

corporatefinanceinstitute.com/resources/excel/correlation-matrix/.

MathWorks. “Create Table from File - MATLAB Readtable - MathWorks United Kingdom.”

Uk.mathworks.com, uk.mathworks.com/help/matlab/ref/readtable.html.

---. “Plot and Return Pearson’s Correlation Coefficients between Variables in Matrix of

Data.” *Mathworks.com*, 2024, uk.mathworks.com/help/econ/corrplot.html. Accessed

29 Sept. 2024.

---. “Two-Sample T-Test - MATLAB Ttest2 - MathWorks United Kingdom.”

Uk.mathworks.com, uk.mathworks.com/help/stats/ttest2.html.

5 Appendix

```
clc
clear
close

%Task1 - Import data and summary statistics:

%Import provided dataset into MatLab
Student_Exams = readtable("student_grades.xlsx");

%Calculate overall exam results and add to summary statistics
Student_Exams.Overall = (Student_Exams.Exam_1 + Student_Exams.Exam_2) / 2;

%Separate numerical data into a matrix
numerical_data = table2array(Student_Exams(:, 3:end));

%Find the mean, median, mode, range and standard deviation from numerical data
mean_matrix = mean(numerical_data);
median_matrix = median(numerical_data);
mode_matrix = mode(numerical_data);
min_matrix = min(numerical_data);
max_matrix = max(numerical_data);
range_matrix = range(numerical_data);
std_matrix = std(numerical_data);

%Display summary statistics in table format
var_names = {'Age'; 'Attendance'; '1st exam score'; '2nd exam score'; 'Overall exam score'};
col_headings = {'variables'; 'mean'; 'median'; 'mode'; 'range'; 'std'};

summary_table = table(var_names, mean_matrix, median_matrix, mode_matrix, range_matrix, std_matrix, 'VariableNames', col_headings);

disp('Summary Statistics:');
disp(summary_table);

%Task2 - Correlation Analysis

%Use corrplot to visualise a correlation matrix
figure('Name', 'Correlation Coefficient');
var_names_corr = {'Age'; 'Attendance'; 'Exam1'; 'Exam2'; 'Overall'};
corrplot(numerical_data, 'varNames', var_names_corr, 'TestR', 'on');

%Display the correlation matrix
disp('Correlation Matrix:');

%Task 2C - Create a hypothesis to demonstrate the relationship between gender and overall exam scores

%Create a hypothesis to demonstrate the relationship between gender and overall exam scores

%Organise numerical data by gender and overall exam results
gender_Score = table(Student_Exams.gender, Student_Exams.Overall, 'VariableNames', {'Gender', 'Overall'});

%Group data by gender
```

```

Male_score = gender_Score{strcmp(gender_Score.Gender, 'Male'), 'Overall'};
Female_score = gender_Score{strcmp(gender_Score.Gender, 'Female'), 'Overall'};

%Use ttest2 to test hypothesis
[h, p] = ttest2(Male_score, Female_score);

%Display the hypothesis results
if h == 0
    hVal_Str = "shows that the means are roughly equal based on the data
provided";
else
    hVal_Str = "shows that the means are not roughly equal based on the data
provided";
end

if p > 0.05
    pVal_Str = "exceeds an alpha of 0.05, showing insufficient evidence to reject
the null hypothesis (h0)";
else
    pVal_Str = "falls below an alpha of 0.05, showing insufficient evidence to
reject the null hypothesis (h0)";
end

fprintf("The hypothesis test gave a h value of = %i %s\n", h, hVal_Str);
fprintf("The hypothesis test gave a value of p = %f %s\n", p, pVal_Str);

%Task 3 - Show data from exams 1 and 2 with the overall score

%Use histcounts to calculate frequency
[E1_count, E1_edges] = histcounts(numerical_data(:, 3), 'BinWidth', 10);
[E2_count, E2_edges] = histcounts(numerical_data(:, 4), 'BinWidth', 10);
[Overall_count, Overall_edges] = histcounts(numerical_data(:, 5), 'BinWidth', 10);

%Histogram Exam 1
figure('Name', 'Histograms');
subplot(1, 3, 1);
histogram('BinCounts', E1_count, 'BinEdges', E1_edges);
title('Exam 1 Histogram');
xlabel('Class limits');
ylabel('Frequency');
ylim([0, 22]);
grid on;

%Histogram Exam 2
subplot(1, 3, 2);
histogram('BinCounts', E2_count, 'BinEdges', E2_edges);
title('Exam 2 Histogram');
xlabel('Class limits');
ylabel('Frequency');
ylim([0, 22]);
grid on;

%Histogram Overall Exam Results
subplot(1, 3, 3);
histogram('BinCounts', Overall_count, 'BinEdges', Overall_edges);
title('Overall Exam Results Histogram');
xlabel('Class Limits');
ylabel('Frequency');
ylim([0, 22]);

```

```

grid on;

%Ogives
figure('Name', 'Ogives');

Exam1_LessThan = [0, cumsum(E1_count / sum(E1_count))];
Exam1_MoreThan = 1 - Exam1_LessThan;

Exam2_LessThan = [0, cumsum(E2_count / sum(E2_count))];
Exam2_MoreThan = 1 - Exam2_LessThan;

Overall_LessThan = [0, cumsum(Overall_count / sum(Overall_count))];
Overall_MoreThan = 1 - Overall_LessThan;

subplot(1, 3, 1);
plot(E1_edges, Exam1_MoreThan, 'b.-', E1_edges, Exam1_LessThan, 'r.-');
title("Exam 1 Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(E1_edges);
grid on;

subplot(1, 3, 2);
plot(E2_edges, Exam2_MoreThan, 'b.-', E2_edges, Exam2_LessThan, 'r.-');
title("Exam 2 Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(E2_edges);
grid on;

subplot(1, 3, 3);
plot(Overall_edges, Overall_MoreThan, 'b.-', Overall_edges, Overall_LessThan, 'r.-');
title("Overall Exam Ogives");
xlabel("Class Limits");
ylabel("Cumulative Relative Frequency");
xticks(Overall_edges);
grid on;

```