Group Simulation Activity 1

Emma Chan (20206641)

Charlotte Lombard (20232888)

Jake Moffat (20212243)

Jack Mason (20060348)

Question 1

Part I

The probability of random variable **x** will be uniformly distributed given that all three of the dice are fair. This means that since there are 6 possible outcomes, there is an equal probability of 1/6 for rolling any value. The probability mass function (PMF) of **x** can be seen in Figure 1 below.

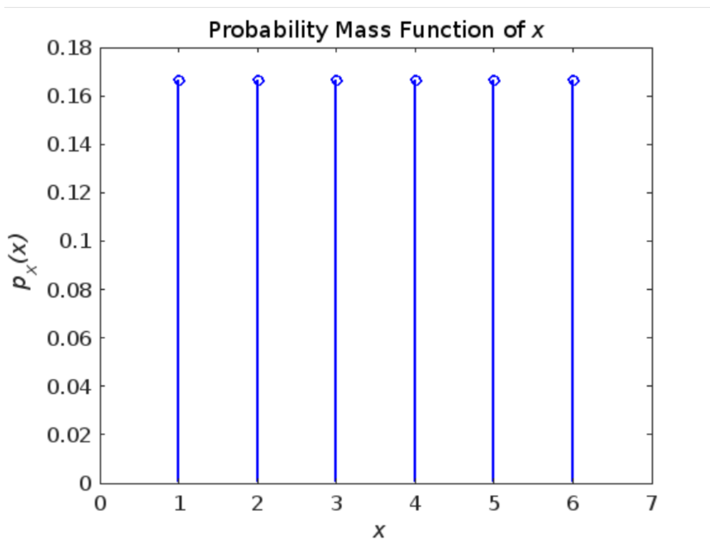

Figure 1: Probability Mass Function of x

The PMF of the discrete random variable **y** was computed through the simulation of each possible outcome of rolling the three dice. Vector *y* was used to store all the possible sums of the three dice. Each time that a sum occurred it was counted and then divided by the total number of all the possible sums. This gave the PMF for **y** which was then plotted, as can be seen in Figure 2 below.

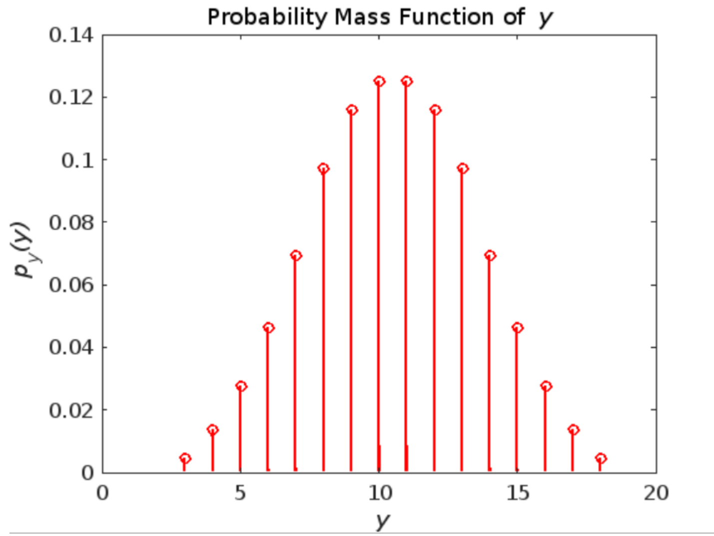

Figure 2: Probability Mass Function of y

Below, in Figure 3 and Figure 4, is the MATLAB code used to create the above plots.

```
% ELEC 326 Group Simulation Activity 1 Question 1 Part 1
% This questions asks you to calculate the PMF of y and then plot the PMFs
% of x and y

%First we need to initialize the array x of potential dice roll values
x = [1, 2, 3, 4, 5, 6];

%Next, we need to build the PMF of a fair dice roll
p_x = (1 / length(x)) * ones(length(x));

%Build y with all possible sums of 3 fair dice rolls
y = 0;
for i = 1:length(x)
    for j = 1:length(x)
        for k = 1:length(x)
            roll_sum = x(i) + x(j) + x(k);
            y = [y roll_sum];
        end
    end
end
y =  y(y > 0);
y = sort(y);

%Remove any duplicates after counting y
[y, ~, counts] = unique(y);
counts = accumarray(counts, 1).';

%Build y's PMF
p_y = zeros(1, length(y));
for i = 1:length(y)
    p_y(i) = counts(i) / sum(counts);
end

%Plot the PMF of x
fig1 = figure('Name', 'Probability Mass Function of x');
fig1.Units = 'centimeters';
stem(x, p_x, 'b', 'LineWidth', 1.5);
title('Probability Mass Function of {\itx}', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax1 = gca;
ax1.XLabel.String = '{\it x}';
ax1.XLim = [0, 7];
ax1.YLabel.String = '{\it p_{x}(x)}';
```

Figure 3: Code used in Q1 part I

```
%Plot the PMF of y
fig2 = figure('Name', 'Probability Mass Function of y');
fig2.Units = 'centimeters';
stem(y, p_y, 'r', 'LineWidth', 1.5);
title('Probability Mass Function of {\it y}', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax2 = gca;
ax2.XLabel.String = '{\it y}';
ax2.XLim = [0, 7];
ax2.YLabel.String = '{\it p_{y}(y)}';
```

Figure 4: Code continued

Part II:

The cumulative distribution function (CDF) for the discrete random variable **x** was plotted and is displayed in Figure 5 below. It was computed by the summation of the PMF up to each value *x* such that $x \in \mathbf{x}$.

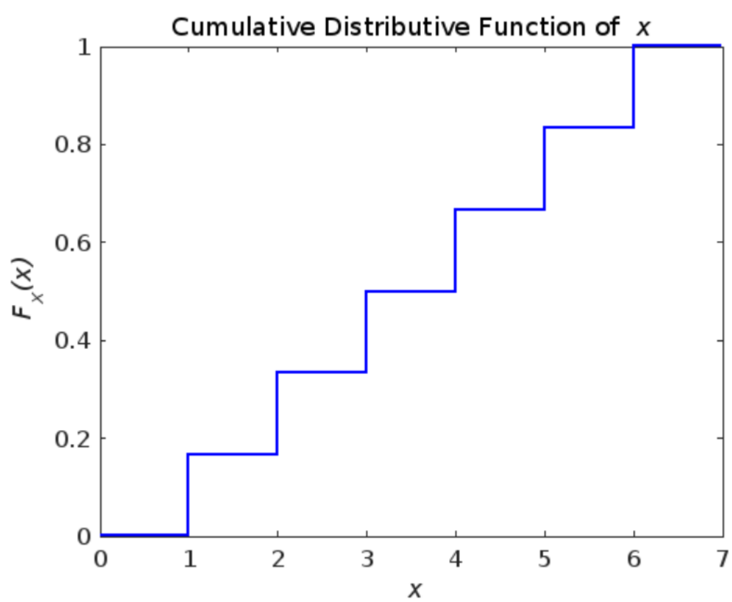

Figure 5:Cumulative distribution function of x

The CDF of discrete random variable **y** was computed using the same process as described above. The PMF was summed up to each value *y* such that $y \in \mathbf{y}$. This gave the staircase function that was plotted and can be seen in Figure 6.

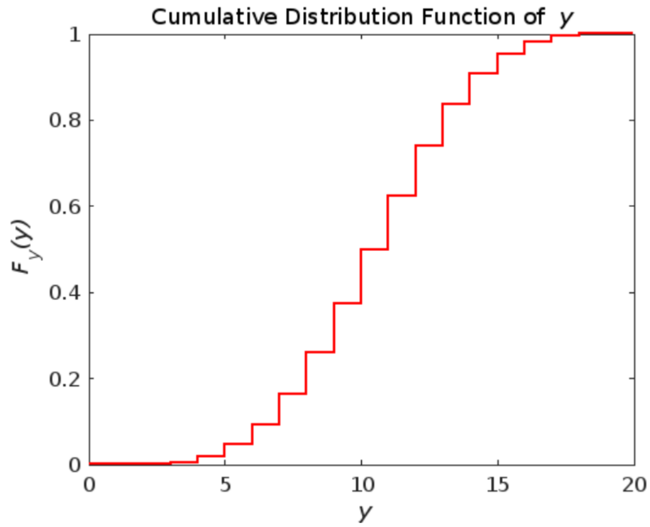

Figure 6: Cumulative distribution function of y

The MATLAB code that was used for this section can be referenced in Figure 7 and Figure 8 below.

```
%First we need to initialize the array x of potential dice roll values
x = [1, 2, 3, 4, 5, 6];

%Next, we need to build the PMF of a fair dice roll
p_x = (1 / length(x)) * ones(length(x));

%Build the CDF of x
F_x = zeros(1, length(x));
for i = 1:length(x)
    F_x(i) = sum(p_x(1:i));
end

%Build y with all possible sums of 3 fair dice rolls
y = 0;
for i = 1:length(x)
    for j = 1:length(x)
        for k = 1:length(x)
            roll_sum = x(i) + x(j) + x(k);
            y = [y roll_sum];
        end
    end
end
y =  y(y > 0);
y = sort(y);

%Remove any duplicates after counting y
[y, ~, counts] = unique(y);
counts = accumarray(counts, 1).';

%Build y's PMF
p_y = zeros(1, length(y));
for i = 1:length(y)
    p_y(i) = counts(i) / sum(counts);
end
```

Figure 7: Code for Q1 part II

```
%Build the CDF of y
F_y = zeros(1, length(y));
for i = 1:length(y)
    F_y(i) = sum(p_y(1:i));
end

%Plot the CDF of x
fig1 = figure('Name', 'Cumulative Distributive Function of x');
fig1.Units = 'centimeters';
stairs([0 x 7], [0 F_x 1], 'b', 'LineWidth', 1.5);
title('Cumulative Distributive Function of {\it x}', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax1 = gca;
ax1.XLabel.String = '{\it x}';
ax1.XLim = [0, 7];
ax1.YLabel.String = '{\it F_{x}(x)}';

%Plot the CDF of y
fig2 = figure('Name', 'Cumulative Distribution Function of y');
fig2.Units = 'centimeters';
stairs([0 y 20], [0 F_y 1], 'r', 'LineWidth', 1.5);
title('Cumulative Distribution Function of {\it y}', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax2 = gca;
ax2.XLabel.String = '{\it y}';
ax2.XLim = [0, 20];
ax2.YLabel.String = '{\it F_{y}(y)}';
```

Figure 8: Code continued

Part III:

The provided *rand.gen(x, pmf, N)* function was used to simulate 100 trials of three dice rolls, with the possible values of **x** and it's PMF from Part I. The results of these trials were stored in vectors and summed, giving the values of **y** for each trial. The MATLAB code that was used for this section of the project can be found in Figure 9.

```
% ELEC 326 Group Simulation Activity 1 Question 1 Part 3
% Genere N trials of 3 dice rolls and store results in vectors x1, x2, x3
% for each die and y for their sum

% Initialize N
N = 100;

% Initialize x as an array
x = [1, 2, 3, 4, 5, 6];

% Build the PMF of fair die
p_x = (1 / length(x)) * ones(1, length(x));

% Conduct N = 100 die rolls for each die
x1 = rand_gen(x, p_x, N);
x2 = rand_gen(x, p_x, N);
x3 = rand_gen(x, p_x, N);

% Build y as the array of sums of all 3 die values for each trials
y = x1 + x2 + x3;
```

Figure 9: Simulating the 100 trials then store all the values in x and y

Part IV:

Plotted in Figure 10 below are the observed values of **x**, which were counted and then organized into vectors.

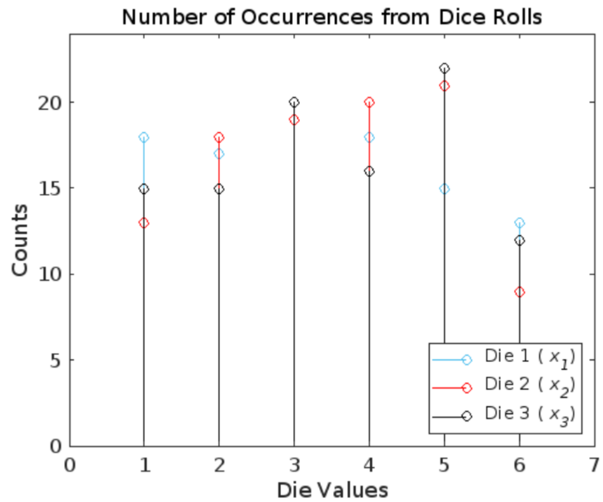

Figure 10: Number of Occurrences

As can be seen in the plot, the count for each dice value is quite similar across each of the die, however they are not identical. Though this plot largely resembles the plot of the PMF of **x** seen in Figure 1, the discrepancies can be explained by the fact that the number of trials in this simulation were limited to 100. Had each count value been divided by N, the number of trials, then the experimental probability distributions of the 100 trials for each of the die would have been displayed. The MATLAB code that was generated to create this section of the report can be seen in Figure 11 below.

```
% Initialize N
N = 100;

% Initialize x as an array
x = [1, 2, 3, 4, 5, 6];

% Build the PMF of fair die
p_x = (1 / length(x)) * ones(1, length(x));

% Conduct N = 100 die rolls for each die
x1 = rand_gen (x, p_x, N);
x2 = rand_gen (x, p_x, N);
x3 = rand_gen (x, p_x, N);

% Count the number of occurrences for each die value for each die
H1 = zeros(1, length(x));
H2 = zeros(1, length(x));
H3 = zeros(1, length(x));
for i = 1:N
    H1(x1(i)) = H1(x1(i)) + 1;
    H2(x2(i)) = H2(x2(i)) + 1;
    H3(x3(i)) = H3(x3(i)) + 1;
end

% Plot the occurrence counts for each die
fig = figure('Name', 'Occurrence Counts');
stem(x, H1, 'Color', [0.3010 0.7450 0.9330], 'LineWidth', 1.25);
hold on;
stem(x, H2, 'r', 'LineWidth', 1.25);
stem(x, H3, 'k', 'LineWidth', 1.25);
hold off;
title('Number of Occurrences from Dice Rolls', 'FontWeight', 'normal');
legend('Die 1 ({\it x_1})', 'Die 2 ({\it x_2})', 'Die 3 ({\it x_3})', 'Location', 'southeast');
set(gca, 'FontSize', 12);
ax = gca;
ax.XLabel.String = 'Die Values';
ax.XLim = [0,7];
ax.YLabel.String = 'Counts';
ax.YLim = [0, max([H1 H2 H3]) + 2];
```

Figure 11: Code used for Q1 Part IV

Part V

Plotted in Figure 12 below are the observed values of **y**, which were counted and then organized into vectors.

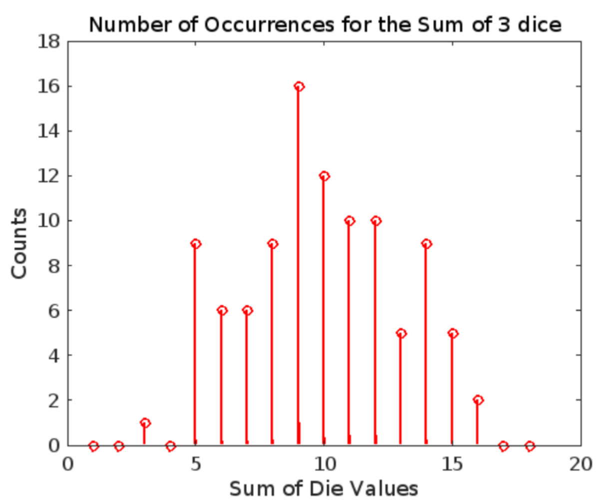

Figure 12: Number of Occurrences for sum of three dice

When comparing the PMF of **y** that is displayed in Figure 2 with Figure 12 above, both plots share a similar shape, but Figure 12 does not perfectly portray the expected results. This is again due to the limiting of the number of trials to just 100. If the count of occurrences were to be divided by the number of trials (N), this would give the experimental probability distribution of the summation of die values when rolling the three dice 100 times. Found below in Figure 13 is the MATLAB code that was used for this section.

```
% ELEC 326 Group Simulation Activity 1 Question 1 Part 5
% Generate N trials of 3 dice rolls and store results in vectors x1, x2, x3
% for each die and y for their sum. Then count number of occurrences for
% each sum and plot them.

% Initialize N
N = 100;

% Initialize x as an array
x = [1, 2, 3, 4, 5, 6];

% Build the PMF of fair die
p_x = (1 / length(x)) * ones(1, length(x));

% Conduct N = 100 die rolls for each die
x1 = rand_gen(x, p_x, N);
x2 = rand_gen(x, p_x, N);
x3 = rand_gen(x, p_x, N);

% Build y as the array of sums of all 3 die values for each of the trials
y = x1 + x2 + x3;

% Count the number of occurrences for each sum
H = zeros(1, 18);
for i = 1:N
    H(y(i)) = H(y(i)) + 1;
end

% Plot the occurrence counts for each die
fig = figure('Name', 'Occurrence Counts');
stem(1:18, H, 'Color', 'r', 'LineWidth', 1.5);
title('Number of Occurrences for the Sum of 3 dice', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax = gca;
ax.XLabel.String = 'Sum of Die Values';
ax.XLim = [0,20];
ax.YLabel.String = 'Counts';
ax.YLim = [0, max(H) + 2];
```

Figure 13: Code for Q1 part V

Part VI:

For this part, the number of trials (N) was increased from 100 to 1 million and the simulation was repeated. Again, the observed values of **x** were counted and stored in vectors. This was then plotted and can be seen displayed in Figure 14 below.

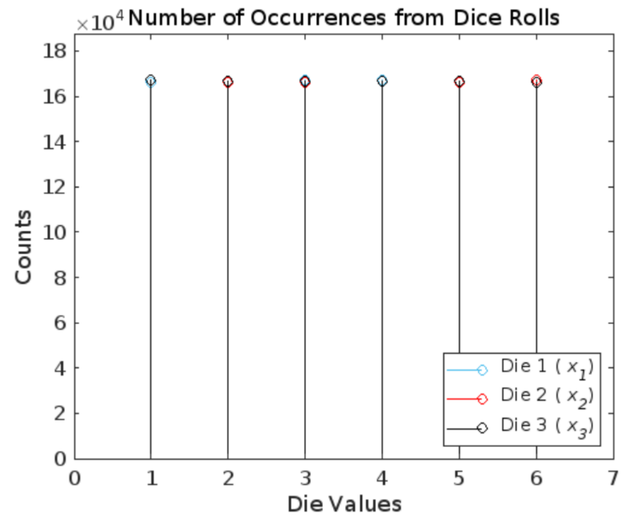

Figure 14: The counted occurrences of each die for the three individual rolls in one million dice rolls

As done in Part V, the observed values of discrete random variable **y** were counted and organized into vectors. This was then plotted and can be seen in Figure 15 below.

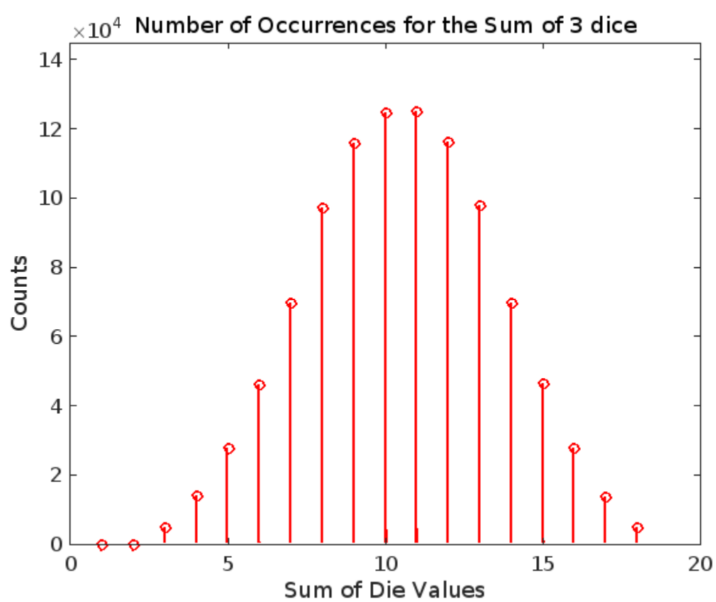

Figure 15: The counted occurrences of the sum of three dice for the one million rolls of three dice

Both Figure 14 and Figure 15 show that the counts of the observed values appear to approach their respective PMFs as the number of trials is increased. This means that the experimental probability of observing a value of **x** or **y** approaches the expected results as the number of trials is increased. Below is the MATLAB code written for this section, found in Figure 16 and Figure 17.

```
% ELEC 326 Group Simulation Activity 1 Question 1 Part 6

% Initialize N
N = 1000000;

% Initialize x as an array
x = [1, 2, 3, 4, 5, 6];

% Build the PMF of fair die
p_x = (1 / length(x)) * ones(1, length(x));

% Conduct N = 100 die rolls for each die
x1 = rand_gen(x, p_x, N);
x2 = rand_gen(x, p_x, N);
x3 = rand_gen(x, p_x, N);

% Count the number of occurrences for each die value for each die
H1 = zeros(1, length(x));
H2 = zeros(1, length(x));
H3 = zeros(1, length(x));
for i = 1:N
    H1(x1(i)) = H1(x1(i)) + 1;
    H2(x2(i)) = H2(x2(i)) + 1;
    H3(x3(i)) = H3(x3(i)) + 1;
end

% Plot the occurrence counts for each die
fig1 = figure('Name', 'Roll Occurrence Counts');
stem(x, H1, 'Color', [0.3010 0.7450 0.9330], 'LineWidth', 1.25);
hold on;
stem(x, H2, 'r', 'LineWidth', 1.25);
stem(x, H3, 'k', 'LineWidth', 1.25);
hold off;
title('Number of Occurrences from Dice Rolls', 'FontWeight', 'normal');
legend('Die 1 ({\it x_1})', 'Die 2 ({\it x_2})', 'Die 3 ({\it x_3})', 'Location', 'southeast');
set(gca, 'FontSize', 12);
ax1 = gca;
ax1.XLabel.String = 'Die Values';
ax1.XLim = [0,7];
ax1.YLabel.String = 'Counts';
ax1.YLim = [0, max([H1 H2 H3]) + 20000];
```

Figure 16: Code used for Q1 part VI

```
% Build y as the array of sums of all 3 die values from each trial
y = x1 + x2 + x3;

% Count the umber of occurrences for each sum
H = zeros(1, 18);
for i = 1:N
    H(y(i)) = H(y(i)) + 1;
end

% Plot the occurrence counts for each die
fig2 = figure('Name', 'Occurrence Counts');
stem(1:18, H, 'Color', 'r', 'LineWidth', 1.5);
title('Number of Occurrences for the Sum of 3 dice', 'FontWeight', 'normal');
set(gca, 'FontSize', 12);
ax2 = gca;
ax2.XLabel.String = 'Sum of Die Values';
ax2.XLim = [0,20];
ax2.YLabel.String = 'Counts';
ax2.YLim = [0, max(H) + 20000];
```

Figure 17: Code continued

Question 2

Part I

The mean of each random variable was estimated through the summation of all the observed values, which was then divided by the total number of those values. The variance was estimated for each random variable by the summation of the square differences between each of the observed values and the estimated mean. The sum was then divided by the total number of values. The results can be found below.

$\mu1 = 50.0030$ $\sigma12 = 100.1685$

$\mu2 = 30.0031$ $\sigma2 = 25.1293$

$\mu3 = 30.0071$ $\sigma32 = 25.1419$

The MATLAB code that was used to generate these values can be found in Figure 18 below.

```
% ELEC 326
% Group Simulation Activity 1
% Question 2 - Part I
% Emma Chan, Charlotte Lombard, Jack Mason, Jake Moffat

% Prompt: Write a MATLAB code to estimate the mean and variance of each of
% the RVs.

% Load the RVs
RV1 = load('RV1.mat').RV1;
RV2 = load('RV2.mat').RV2;
RV3 = load('RV3.mat').RV3;

% First random variable
% Mean Estimate
mean1 = sum(RV1)/length(RV1);

% Variance Estimate
variance1 = 0;
for i = 1:length(RV1)
    variance1 = variance1 + (RV1(i) - mean1)^2;
end
variance1 = variance1/length(RV1);

% Second random variable
% Mean Estimate
mean2 = sum(RV2)/length(RV2);

% Variance Estimate
variance2 = 0;
for i = 1:length(RV2)
    variance2 = variance2 + (RV2(i) - mean2)^2;
end
variance2 = variance2/length(RV2);

% Third random variable
% Mean Estimate
mean3 = sum(RV3)/length(RV3);

% Variance Estimate
variance3 = 0;
for i = 1:length(RV3)
    variance3 = variance3 + (RV3(i) - mean3)^2;
end
variance3 = variance3/length(RV3);
```

Figure 18: Code used for Q2 part I

Part II

The values for each random variable were recorded and each occurrence of unique values ranging from zero to one hundred were counted and stored in separate variables H1, H2 and H3. Then these vectors were divided by the length of the data set and plotted as seen in Figure 19.

Then the data was compared directly to its own normal distribution that was calculated using the mean and variance of the RV1. As clearly seen in the figure, this random variable follows a normal (gaussian) distribution of data.

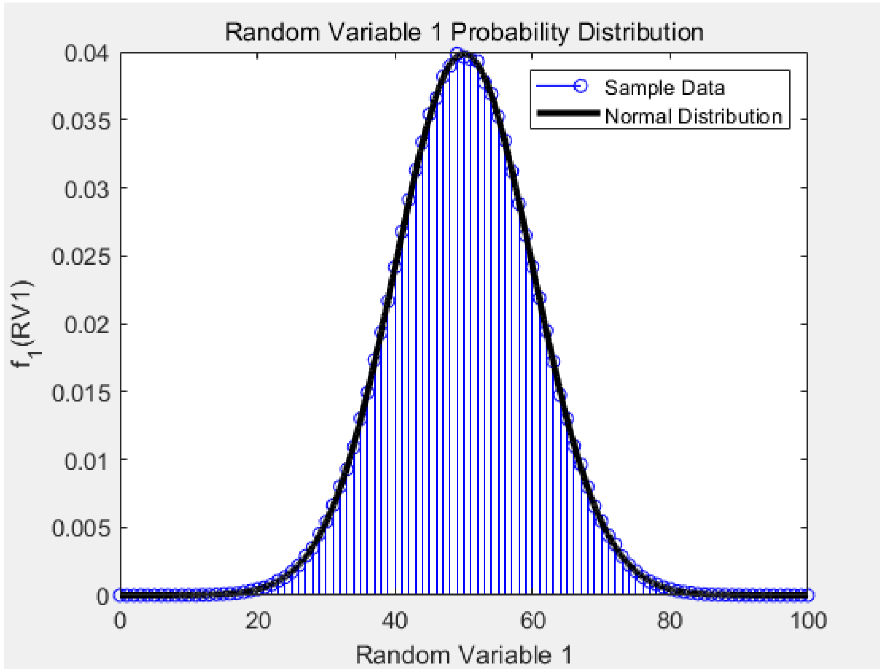

Figure 19: Probability distribution of RV 1 and normal distribution curve

The same process was applied to the other random variables; RV2 and RV3 along with their corresponding occurrences counters; H2 and H3.

Quantitively, both plots seemed to follow a normal distribution similar to the first RV and this was confirmed when their normal distributions were calculated using the mean and variances of each variable. Their plots can be seen below in Figure 20 and Figure 21.

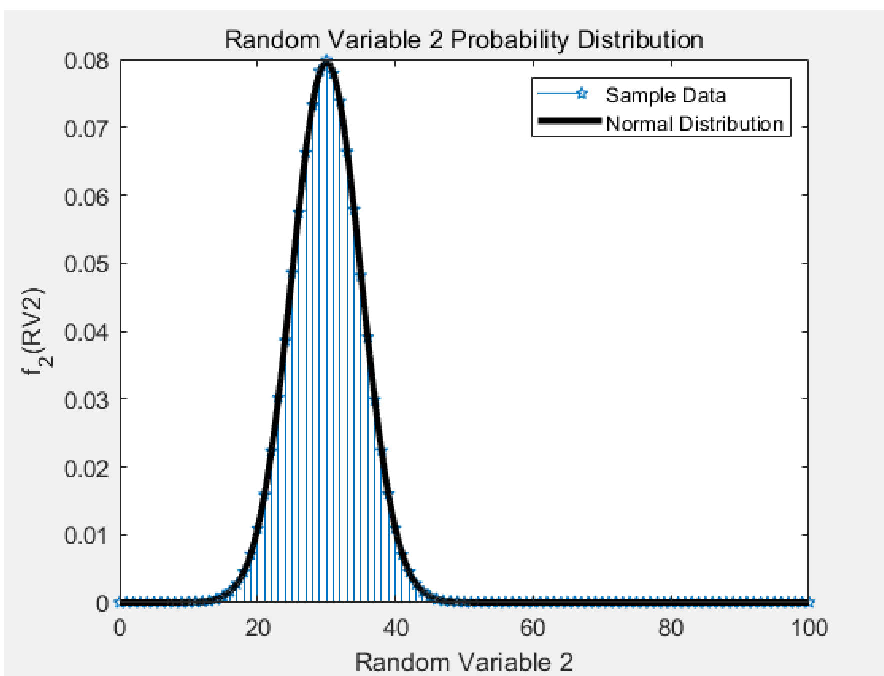

Figure 20: Probability distribution of RV2 and normal distribution of RV2

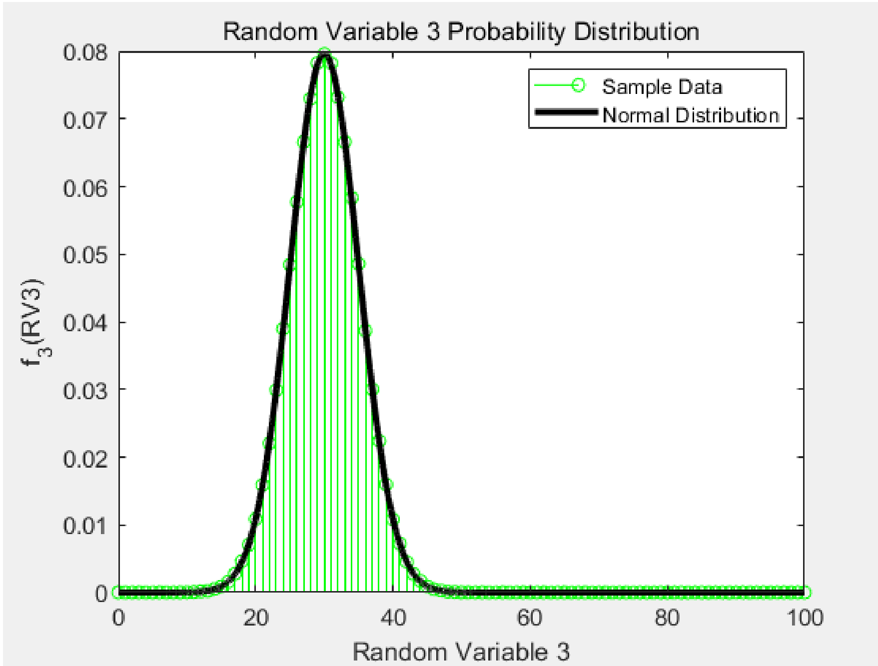

Figure 21: Probability distribution of RV3 and normal distribution of RV3

The code used to display the above plots is shown below in Figure 22 for RV1, Figure 23 for RV2, and Figure 24 for RV3.

```
% ELEC 326
% Group Simulation Activity 1
% Question 2 - Part II
% Emma Chan, Charlotte Lombard, Jack Mason, Jake Moffat

% Prompt: For each RV, write a MATLAB code to count the number of times each number {0,1,2,. . . ,100}
% is observed and save them in vectors H1, H2 and H3. Then divide H1, H2 and H3 by the
% number of trials (1 million) and plot them.

% Load the RVs
RV1 = load('RV1.mat').RV1;
RV2 = load('RV2.mat').RV2;
RV3 = load('RV3.mat').RV3;
RV = 0:100;

% First random variable
% Count the occurences for each value
H1 = zeros(1, 101);
for i = 1:length(RV1)
    H1(RV1(i) + 1) = H1(RV1(i) + 1) + 1;
end
H1 = H1/length(RV1);

% Probability Distribution Plot
plot1 = figure('Name', 'Random Variable');
stem(RV, H1, 'b');
title('Random Variable 1 Probability Distribution', 'FontWeight', 'normal');
set(gca, 'FontSize', 11);
axis1 = gca;
axis1.XLabel.String = 'Random Variable 1';
axis1.XLim = [0, 100];
axis1.YLabel.String = 'f_1(RV1)';

% Mean Estimate
mean1 = sum(RV1)/length(RV1);

% Variance Estimate
variance1 = 0;
for i = 1:length(RV1)
    variance1 = variance1 + (RV1(i) - mean1)^2;
end
variance1 = variance1/length(RV1);

% Normal distribution including the estimated mean and variance
x1 = linspace(0, 100, 10000);
f_x1 = exp(-1.0 * (x1 - mean1) .* (x1 - mean1) / (2.0 * variance1)) / sqrt(2.0 * pi * variance1);

% Normal Distribution Plot
hold on;
plot(x1, f_x1, 'k', 'LineWidth', 3);
legend('Sample Data', 'Normal Distribution', 'Location', 'North East');
hold off;
```

Figure 22: Code for Q2 Part II: RV1

```matlab
% Second random variable
% Count the occurences for each value
H2 = zeros(1, 101);
for i = 1:length(RV2)
    H2(RV2(i) + 1) = H2(RV2(i) + 1) + 1;
end
H2 = H2/length(RV2);

% Probability Distribution Plot
plot2 = figure('Name', 'Random Variable');
stem(RV, H2, 'p');
title('Random Variable 2 Probability Distribution', 'FontWeight', 'normal');
set(gca, 'FontSize', 11);
axis2 = gca;
axis2.XLabel.String = 'Random Variable 2';
axis2.XLim = [0, 100];
axis2.YLabel.String = 'f_2(RV2)';

% Mean Estimate
mean2 = sum(RV2)/length(RV2);

% Variance Estimate
variance2 = 0;
for i = 1:length(RV2)
    variance2 = variance2 + (RV2(i) - mean2)^2;
end
variance2 = variance2/length(RV2);

% Normal distribution including the estimated mean and variance
x2 = linspace(0, 100, 10000);
f_x2 = exp(-1.0 * (x2 - mean2) .* (x2 - mean2) / (2.0 * variance2)) / sqrt(2.0 * pi * variance2);

% Normal Distribution Plot
hold on;
plot(x2, f_x2, 'k', 'LineWidth', 3);
legend('Sample Data', 'Normal Distribution', 'Location', 'North East');
hold off;
```

Figure 23: Q2 for part II: RV2

```
% Third random variable
% Count the occurences for each value
H3 = zeros(1, 101);
for i = 1:length(RV3)
    H3(RV3(i) + 1) = H3(RV3(i) + 1) + 1;
end
H3 = H3/length(RV3);

% Probability Distribution Plot
plot3 = figure('Name', 'Random Variable');
stem(RV, H3, 'g');
title('Random Variable 3 Probability Distribution', 'FontWeight', 'normal');
set(gca, 'FontSize', 11);
axis3 = gca;
axis3.XLabel.String = 'Random Variable 3';
axis3.XLim = [0, 100];
axis3.YLabel.String = 'f_3(RV3)';

% Mean Estimate
mean3 = sum(RV3)/length(RV3);

% Variance Estimate
variance3 = 0;
for i = 1:length(RV3)
    variance3 = variance3 + (RV3(i) - mean3)^2;
end
variance3 = variance3/length(RV3);

% Normal distribution including the estimated mean and variance
x3 = linspace(0, 100, 10000);
f_x3 = exp(-1.0 * (x3 - mean3) .* (x3 - mean3) / (2.0 * variance3)) / sqrt(2.0 * pi * variance3);

% Normal Distribution Plot
hold on;
plot(x3, f_x3, 'k', 'LineWidth', 3);
legend('Sample Data', 'Normal Distribution', 'Location', 'North East');
hold off;
```

Figure 24: Code for Q2 part II: RV3

Part III

The occurrences of each variable were counted and stored in vectors to estimate the probability that the variables take on a value between 10 and 40. Next, the vectors were divided by the total number of occurrences, which gave an experimental probability distribution for each random variable. The vectors were then summed for all indices corresponding to values between 10 and 40 (inclusive).

$$P1[10 \le x \le 40] = 0.1709$$

$$P2[10 \le x \le 40] = 0.9821$$

$$P3[10 \le x \le 40] = 0.9820$$

Using the distributions that were found in Part II, the probabilities for each of the random variables were approximated. This was done by taking the integral of the analytical PDF corresponding to the chosen distribution and setting the bounds of the integral to 10 and 40.

$$P_1 [10 \leq x \leq 40] = 0.1588$$

$$P_2 [10 \leq x \leq 40] = 0.9769$$

$$P_3 [10 \leq x \leq 40] = 0.9768$$

The relative difference between each pair of results from above is only about 7% for the first variable and 0.5% for the other two. Given the results of Part II, it is probable that the second method is slightly more accurate, however either of the two methods can be used to provide an accurate estimation of the probability.

The MATLAB code used for this section is displayed in Figure 25 and Figure 26 below.

```
% ELEC 326
% Group Simulation Activity 1
% Question 2 - Part III
% Emma Chan, Charlotte Lombard, Jack Mason, Jake Moffat

% For each RV, using the vectors H1, H2 and H3, write a MATLAB code to estimate the
% probability that that RV takes values between 10 and 40.

% Load the RVs
RV1 = load('RV1.mat').RV1;
RV2 = load('RV2.mat').RV2;
RV3 = load('RV3.mat').RV3;
RV = 0:100;

% First random variable
% Count the occurences for each value
H1 = zeros(1, 101);
for i = 1:length(RV1)
    H1(RV1(i) + 1) = H1(RV1(i) + 1) + 1;
end
H1 = H1/length(RV1);

% Probability of RV1 between values 10 and 40
pRV1 = sum(H1(11:41));

% Mean Estimate
mean1 = sum(RV1)/length(RV1);

% Variance Estimate
variance1 = 0;
for i = 1:length(RV1)
    variance1 = variance1 + (RV1(i) - mean1)^2;
end
variance1 = variance1/length(RV1);

% Second Probability of RV1 between values 10 and 40
x1 = linspace(0, 100, 10000);
f_x1 = @(x1) exp(-1.0 * (x1 - mean1) .* (x1 - mean1) / (2.0 * variance1)) / sqrt(2.0 * pi * variance1);
pRV1_norm = integral(f_x1, 10, 40);

% Second random variable
% Count the occurences for each value
H2 = zeros(1, 101);
for i = 1:length(RV2)
    H2(RV2(i) + 1) = H2(RV2(i) + 1) + 1;
end
H2 = H2/length(RV2);
```

Figure 25: Code used for Q2 part III

```
% Probability of RV2 between values 10 and 40
pRV2 = sum(H2(11:41));

% Mean Estimate
mean2 = sum(RV2)/length(RV2);

% Variance Estimate
variance2 = 0;
for i = 1:length(RV2)
    variance2 = variance2 + (RV2(i) - mean2)^2;
end
variance2 = variance2/length(RV2);

% Second Probability of RV2 between values 10 and 40
x2 = linspace(0, 100, 10000);
f_x2 = @(x2) exp(-1.0 * (x2 - mean2) .* (x2 - mean2) / (2.0 * variance2)) / sqrt(2.0 * pi * variance2);
pRV2_norm = integral(f_x2, 10, 40);

% Third random variable
% Count the occurences for each value
H3 = zeros(1, 101);
for i = 1:length(RV3)
    H3(RV3(i) + 1) = H3(RV3(i) + 1) + 1;
end
H3 = H3/length(RV3);

% Probability of RV1 between values 10 and 40
pRV3 = sum(H3(11:41));

% Mean Estimate
mean3 = sum(RV3)/length(RV3);

% Variance Estimate
variance3 = 0;
for i = 1:length(RV3)
    variance3 = variance3 + (RV3(i) - mean3)^2;
end
variance3 = variance3/length(RV3);

% Second Probability of RV1 between values 10 and 40
x3 = linspace(0, 100, 10000);
f_x3 = @(x3) exp(-1.0 * (x3 - mean3) .* (x3 - mean3) / (2.0 * variance3)) / sqrt(2.0 * pi * variance3);
pRV3_norm = integral(f_x3, 10, 40);
```

Figure 26: Code used for Q2 part III continued