

1.

## Introduction

In this assignment, we aim to find the probabilities of the English alphabet and determine the ten most frequent such letters in the two given text files. We also calculate each alphabet's entropies (uncertainties) in the second file and further analyse our results using two additional files.

## Data

We are given 4 text files. The first appears to be a set of letters separated by commas. The second is 'JANE EYRE', an autobiography by Charlotte Bronte. The third is titled 'HANDBOOK OF SUMMER ATHLETIC SPORTS', and the fourth, 'The Time Machine; An Invention' by H. G. Wells. Bront's writing is more down-to-earth and relates more to everyday life as compared to the other two books (which are a part of the Sports and Science Fiction disciplines respectively).

## Methodology

The following code was implemented in MATLAB to analyse the digital files. The comments have been colored in green for an easy read.

```
%%Load text files
folderPath = 'C:\Users\aalay\Downloads'; % Update this path
fileA = fullfile(folderPath, 'fileA.txt');
fileB = fullfile(folderPath, 'fileB.txt');
fileC = fullfile(folderPath, 'fileC.txt');
fileD = fullfile(folderPath, 'fileD.txt');

%% Helper function to clean text (remove special characters and convert to lowercase)
% This function takes raw text and outputs a clean string containing only lowercase letters
preprocessText = @(text) lower(regexprep(text, '[^a-zA-Z]', ''));

%% Part (a): Analyze the probability of each letter in fileA
% Read and preprocess the content of fileA
textA = fileread(fileA);
processedTextA = preprocessText(textA);
```

```

% Calculate the total number of letters in the processed text
totalCharsA = length(processedTextA);

% Define the set of alphabets ('a' to 'z')
alphabets = 'a':'z';

% Count the occurrences of each alphabet in the text
alphabetCountsA = arrayfun(@(ch) sum(processedTextA == ch), alphabets);

% Calculate the probability of each alphabet
alphabetProbabilitiesA = alphabetCountsA / totalCharsA;

% Identify the top 10 most frequent alphabets by sorting probabilities in descending order
[~, sortedIndicesA] = sort(alphabetProbabilitiesA, 'descend');
topTenAlphabets = alphabets(sortedIndicesA(1:10));
topTenProbabilities = alphabetProbabilitiesA(sortedIndicesA(1:10));

% Display the top 10 alphabets and their probabilities
disp('Top 10 most frequent alphabets in fileA and their probabilities:');
for i = 1:10
    fprintf('%c: %.3f\n', topTenAlphabets(i), topTenProbabilities(i));
end

%% Part (b): Calculate the entropy of alphabets in fileB
% Read and preprocess the content of fileB
textB = fileread(fileB);
processedTextB = preprocessText(textB);

% Calculate the total number of letters in the processed text
totalCharsB = length(processedTextB);

% Count the occurrences of each alphabet in the text
alphabetCountsB = arrayfun(@(ch) sum(processedTextB == ch), alphabets);

% Calculate the probability of each alphabet
alphabetProbabilitiesB = alphabetCountsB / totalCharsB;

% Compute the entropy using the formula  $H = \sum(-p \log p)$ 
entropyB = -sum(alphabetProbabilitiesB .* log2(alphabetProbabilitiesB + eps));

```

```

% Display the calculated entropy
fprintf('Entropy of alphabets in fileB: %.3f\n', entropyB);

%% Part (c): Analyze word probabilities and entropy for fileC and fileD
% Helper function to clean text for word analysis (retain spaces and remove special
characters)
% This function processes the text to keep words intact for analysis
preprocessWords = @(text) split(lower(regexprep(text, '[^a-zA-Z\s]', '')));

% Analyze fileC for word probabilities
% Read and preprocess the content of fileC
textC = fileread(fileC);
processedWordsC = preprocessWords(textC);

% Identify unique words and count their occurrences
uniqueWordsC = unique(processedWordsC);
wordCountsC = cellfun(@(word) sum(strcmp(processedWordsC, word)), uniqueWordsC);

% Calculate the total number of words
totalWordsC = sum(wordCountsC);

% Calculate the probability of each word
wordProbabilitiesC = wordCountsC / totalWordsC;

% Identify the top 10 most frequent words by sorting probabilities in descending order
[~, sortedIndicesC] = sort(wordProbabilitiesC, 'descend');
topTenWordsC = uniqueWordsC(sortedIndicesC(1:10));
topTenWordProbabilitiesC = wordProbabilitiesC(sortedIndicesC(1:10));

% Display the top 10 words and their probabilities
disp('Top 10 most frequent words in fileC and their probabilities:');
for i = 1:10
    fprintf('%s: %.3f\n', topTenWordsC{i}, topTenWordProbabilitiesC(i));
end

% Analyze fileD for word entropy
% Read and preprocess the content of fileD, following same process as file B
textD = fileread(fileD);
processedWordsD = preprocessWords(textD);

% Identify unique words and count their occurrences
uniqueWordsD = unique(processedWordsD);
wordCountsD = cellfun(@(word) sum(strcmp(processedWordsD, word)), uniqueWordsD);

```

```

% Calculate the total number of words
totalWordsD = sum(wordCountsD);

% Calculate the probability of each word
wordProbabilitiesD = wordCountsD / totalWordsD;

entropyD = -sum(wordProbabilitiesD .* log2(wordProbabilitiesD + eps));

% Display the calculated entropy
fprintf('Entropy of words in fileD: %.3f\n', entropyD);

```

## Results:

<p>Top 10 most frequent alphabets in file A and their probabilities:</p> <p>s: 0.042 y: 0.041 z: 0.041 f: 0.041 w: 0.040 t: 0.040 u: 0.040 x: 0.040 j: 0.040 k: 0.040</p>	<p>Top 10 most frequent words in file C and their probabilities:</p> <p>the: 0.081 of: 0.036 and: 0.030 to: 0.028 a: 0.025 in: 0.024 is: 0.015 be: 0.012 as: 0.010 that: 0.009</p>
Entropy of alphabets in file B: 4.176	Entropy of words in file D: 9.214


## Discussion:

The most commonly used word in the English language is 'the' [1], which appears twice for the second most common word [2] 'of', followed by 'and' [3]. From this information, we would expect that the most common alphabet would be 'e', as it appears in both 'the' and 'be'. **Zipf's law** is an empirical formula discovered by **George Zipf** in 1930s that describes the relationship between the frequency of words in language corpus and their rank in a frequency sorted list.[4] Zipf's law is given by "The second most used word appears half as often as the most used word, the third most used word appears one-third the number of times the most used word appears, and so on." So, we can expect to see a trend in the probabilities of a word appearing according to it.

## Conclusion:

The most common alphabet was 's' not the one we had expected. This could be due to the reference file itself (since file A was not filled with words but individual characters). The words in File D were more uncertain than the ones in File B (since it has higher H value).

## References:

- [1] <https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt>
- [2] Vsauce. "The Zipf Mystery." YouTube video, 0:51. Uploaded by Vsauce, September 16, 2015.  The Zipf Mystery
- [3] <https://word-lists.com/word-lists/the-1000-most-common-words-in-english/>
- [4] <https://www.geeksforgeeks.org/zipfs-law/>

## 2,3:

### Introduction:

This experiment explores two widely used probability distributions: the **Uniform Distribution** and the **Gaussian Distribution**. We aim to observe how the sample mean and variance behave as the sample size increases, comparing them to their theoretical population values.

### Data:

#### Uniform Distribution

The **Uniform Distribution** generates random numbers that are equally likely to fall anywhere between 0 and 1. The theoretical mean of the uniform distribution is 0.5, and the variance is  $1/12$ . To test how the sample mean behaves as the sample size increases, we generated random numbers for sample sizes of **5, 50, 500, 5000, and 50000**. For each sample size, we calculated the mean and variance of the generated numbers and compared them to the theoretical values.[\[1\]](#)

#### Gaussian Distribution

The **Gaussian Distribution**, or Normal Distribution, is one of the most commonly used distributions in statistics. In this experiment, we set the population mean to 4 and the population standard deviation to 3. Similarly, random numbers were generated for the same sample sizes, and their means and variances were calculated.[\[2\]](#)

## Methodology:

The following code was implemented in MATLAB to analyse the digital files. The comments have been colored in green for an easy read. This script explores the behavior of random numbers generated from two distributions:

1. Uniform distribution (values between 0 and 1)
2. Gaussian (normal) distribution with mean 4 and standard deviation 3

```
%% Experiment 1: Uniformly Distributed Random Numbers
% Define sample sizes for the experiments
sampleSizes = [5, 50, 500, 5000, 50000];

% Initialize arrays to store means and variances for each sample size
uniformMeans = zeros(size(sampleSizes));
uniformVariances = zeros(size(sampleSizes));

% Loop through each sample size and calculate mean and variance
for i = 1:length(sampleSizes)
    n = sampleSizes(i); % Current sample size
    randomNumbers = rand(1, n); % Generate n random numbers uniformly distributed
    between 0 and 1
    uniformMeans(i) = mean(randomNumbers); % Calculate the mean
    uniformVariances(i) = var(randomNumbers); % Calculate the variance
end

% Display results for uniform distribution
fprintf('Uniform Distribution Results:\n');
for i = 1:length(sampleSizes)
    fprintf('n = %d: Mean = %.3f, Variance = %.4f\n', sampleSizes(i), uniformMeans(i),
    uniformVariances(i));
end

%% Plot Uniform Distribution
figure;
subplot(2, 1, 1); % Create a subplot for the uniform distribution
plot(sampleSizes, uniformMeans, '-o', 'LineWidth', 2, 'MarkerSize', 8);
hold on;
yline(0.5, 'r--', 'LineWidth', 2); % Population mean (0.5 for uniform distribution)
title('Uniform Distribution: Sample Mean vs. Population Mean');
xlabel('Sample Size');
ylabel('Sample Mean');
legend('Sample Mean', 'Population Mean', 'Location', 'Best');
grid on;
```

```

%% Experiment 2: Gaussian (Normal) Distributed Random Numbers
% Define Gaussian distribution parameters
meanGaussian = 4;
stdGaussian = 3;

% Initialize arrays to store means and variances for each sample size
gaussianMeans = zeros(size(sampleSizes));
gaussianVariances = zeros(size(sampleSizes));

% Loop through each sample size and calculate mean and variance
for i = 1:length(sampleSizes)
    n = sampleSizes(i); % Current sample size
    randomNumbers = meanGaussian + stdGaussian * randn(1, n); % Generate n Gaussian
    random numbers
    gaussianMeans(i) = mean(randomNumbers); % Calculate the mean
    gaussianVariances(i) = var(randomNumbers); % Calculate the variance
end

% Display results for Gaussian distribution
fprintf('\nGaussian Distribution Results:\n');
for i = 1:length(sampleSizes)
    fprintf('n = %d: Mean = %.3f, Variance = %.3f\n', sampleSizes(i), gaussianMeans(i),
    gaussianVariances(i));
end

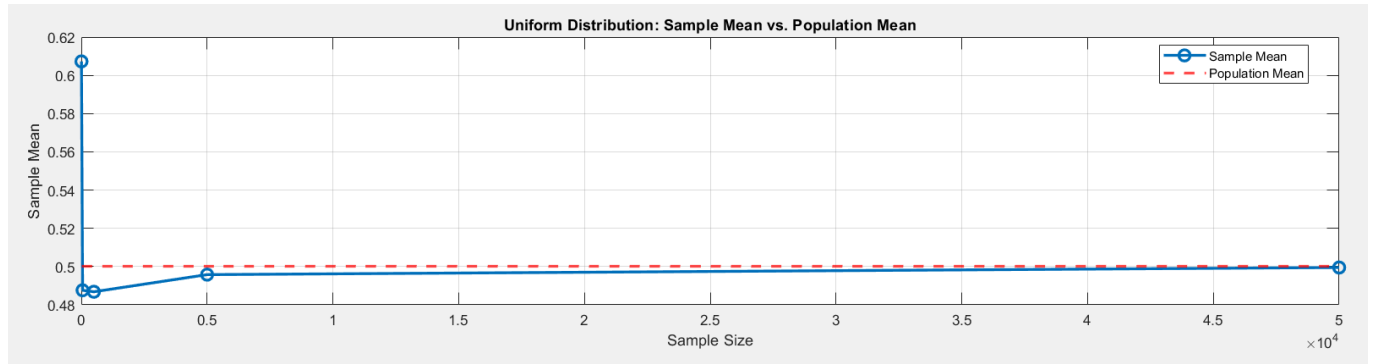
%% Plot Gaussian Distribution
subplot(2, 1, 2); % Create a subplot for the Gaussian distribution
plot(sampleSizes, gaussianMeans, '-o', 'LineWidth', 2, 'MarkerSize', 8);
hold on;
yline(meanGaussian, 'r--', 'LineWidth', 2); % Population mean (meanGaussian for Gaussian
distribution)
title('Gaussian Distribution: Sample Mean vs. Population Mean');
xlabel('Sample Size');
ylabel('Sample Mean');
legend('Sample Mean', 'Population Mean', 'Location', 'Best');
grid on;

```

## Results:

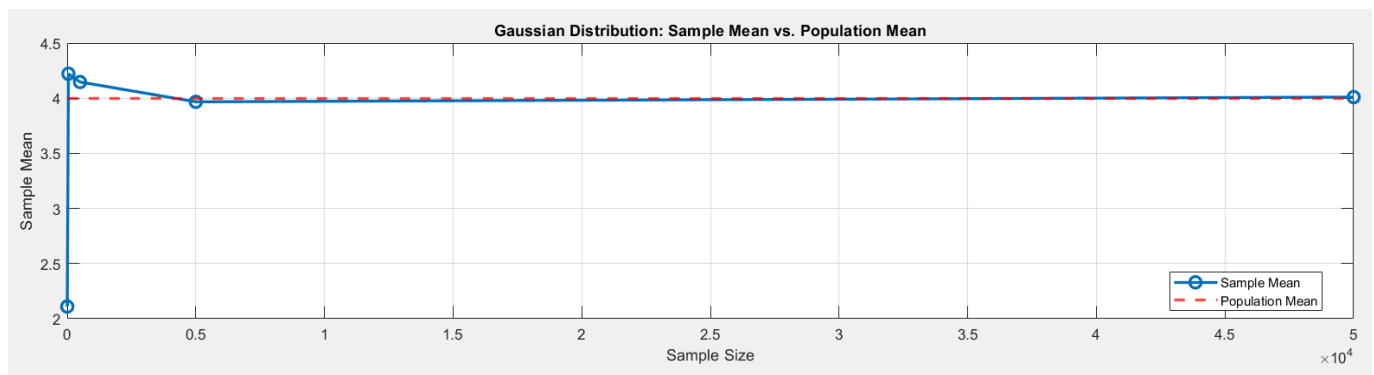
Uniform distribution results:

n = 5:	Mean = 0.607,	Variance = 0.0479
n = 50:	Mean = 0.488,	Variance = 0.0737
n = 500:	Mean = 0.487,	Variance = 0.0880
n = 5000:	Mean = 0.496,	Variance = 0.0830
n = 50000:	Mean = 0.500,	Variance = 0.0834



Gaussian distribution results:

n = 5: Mean = 2.110, Variance = 1.395  
 n = 50: Mean = 4.221, Variance = 7.813  
 n = 500: Mean = 4.147, Variance = 8.018  
 n = 5000: Mean = 3.967, Variance = 8.765  
 n = 50000: Mean = 4.010, Variance = 9.106



## Discussion:

For the **Uniform Distribution**, the sample mean was initially a bit variable, especially for smaller sample sizes (such as 5 and 50), but as we moved to larger sample sizes (5000 and 50000), the sample mean stabilized around 0.5, the true population mean. Similarly, the variance showed an interesting pattern of reducing fluctuations as sample sizes increased.

For the **Gaussian Distribution**, the sample mean behaved in a very similar way, gradually approaching the theoretical value of 4 as the sample size grew. This behavior is expected, as the Gaussian distribution is well-behaved and converges quickly to its expected value even with relatively smaller sample sizes compared to other distributions.



It is interesting to note that both distributions showed larger variance at smaller sample sizes, but the variance also settled down as the sample size increased, reflecting the natural tendency for random variables to display less variability when larger samples are taken. [3]

### **Conclusion:**

As the sample size increases, both the uniform and Gaussian distributions converge towards their theoretical mean and variance. This demonstrates that as the number of trials increases, the sample mean will converge to the expected value of the distribution.

### **References:**

[1] [https://stats.libretexts.org/Bookshelves/Introductory\\_Statistics/Introductory\\_Statistics\\_1e\\_\(Open\\_Stax\)/05%3A\\_Continuous\\_Random\\_Variables/5.03%3A\\_The\\_Uniform\\_Distribution](https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Introductory_Statistics_1e_(Open_Stax)/05%3A_Continuous_Random_Variables/5.03%3A_The_Uniform_Distribution)

[2] <https://www.math.net/gaussian-distribution>

[3] <https://stats.stackexchange.com/questions/129885/why-does-increasing-the-sample-size-lower-the-sampling-variance>

---