By the end of this lab, the student should be able to:

EO1: Perform a Frequency Analysis on a text

EO2: Determine the most likely substitution using Frequency Analysis

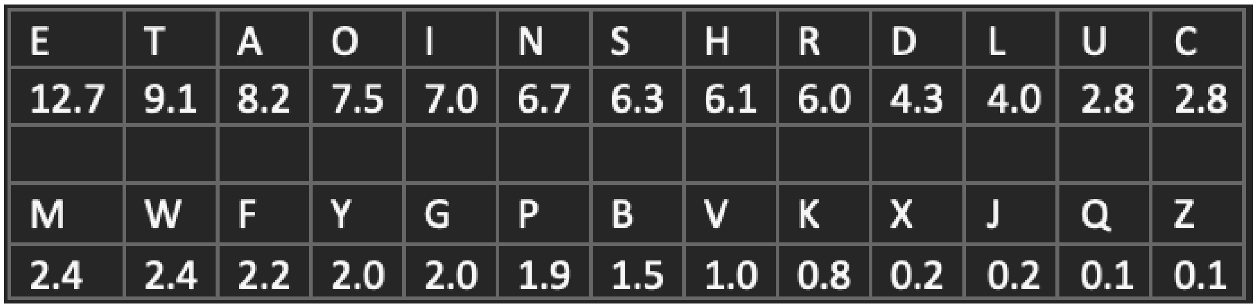
EO3: Crack an unknown substitution text using Frequency Analysis

---

Abstract

Substitution Ciphers are one of the easiest ciphers to create and the easiest to crack. These ciphers are prone to brute force attacks, especially when using analysis tools to find patterns in the text.

One readily apparent pattern is the frequency of letters. A large study of English texts shows that statistically, the following frequencies have been established:



E	T	A	O	I	N	S	H	R	D	L	U	C
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8
M	W	F	Y	G	P	B	V	K	X	J	Q	Z
2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.2	0.2	0.1	0.1

This table indicates that for large volumes of English prose, the letter **E** occurs about 12.7% of the time. That does not mean it will necessarily be the most common letter in any given text. For example, scientific writing may favor letters that are far less common, such as the letters **Q** and **X**, perhaps, in a chemical paper.

Additionally, the English language relies heavily on diphthongs, such as the vowel combination **ai** in the words fair, hair, and airplane. Additionally, letters like **ght**, **str**, **st**, **th**, and many others frequently occur in words and can be used in the analysis.

Other languages have similar peculiarities and patterns that can be used to decipher, but each is unique to that language.

Requirements & Configuration

System Requirements

The provided lab environment.

Network Requirements

This lab should be done on a local Virtual Machine or the host computer. No network is necessary unless Google or online tools are needed.

Software Requirements

Python 3.x or later

If familiar with the Thonny IDE (thonny.org), complete this lab using Thonny.

Data Requirements

Download the CRY100-M1-1Files.zip

Download CRY100-M1-1Files.zip

file and unzip it. This file contains the following:

Python Scripts

decoder.py

decoderTS.py

encoder.py

frequencyAnalysis.py

Text Files for Analysis

alice.txt

Substitution.txt

tomsawyer.txt

treasureisland.txt

ts1.txt

ts2.txt

twocities.txt

Ensure all files (python scripts and text files) are in the same directory.

Procedure – Detailed Lab Steps

Lab Execution

Frequency Analysis

In the **CRY100 Student Resource folder**, find the script called **frequencyAnalysis.py**. This script will parse through a text file and count the occurrence of each letter.

Open the file called **alice.txt** in a text editor (Notepad, Notepad++, or any other editor). This is the first chapter of Alice's Adventures in Wonderland by Lewis Carroll.

Run this through the frequency analysis script. This can be done directly using Python or using Thonny.

Enter the following on the command line:

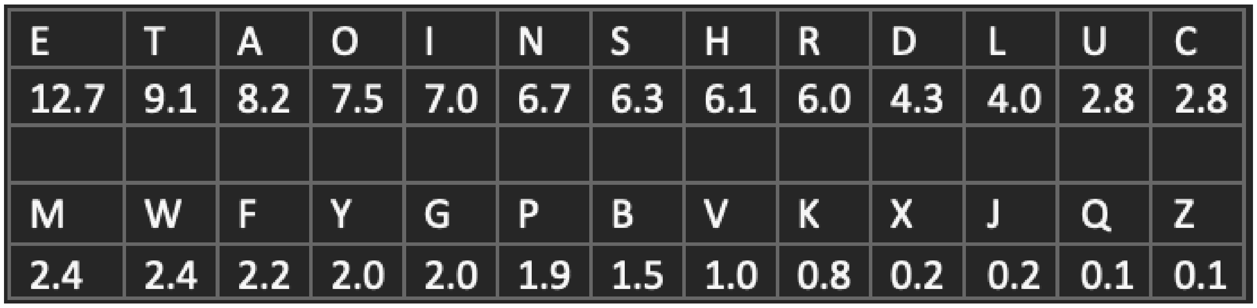
```
python3 frequencyAnalysis.py
```

If using Thonny, load **frequencyAnalysis.py** into Thonny and click **run**.

When it requests the file to open, enter **alice.txt** and hit **enter**.

This will show an output indicating which letter occurs most frequently, which is second most frequent, etc.

These will correspond to the letters of the English language based on a frequency table:



E	T	A	O	I	N	S	H	R	D	L	U	C
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8
M	W	F	Y	G	P	B	V	K	X	J	Q	Z
2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.2	0.2	0.1	0.1

The analysis is statistically based, so they may not line up exactly.

Notice in the **alice.txt** file, the letter **E** occurs 1075 times and is 12.5%, pretty close to the prediction in the table above. **T** occurs next, at 10.05%, a little more than expected. **O** is next, at 8.07%, and so on. The actual occurrence of the letters is not exactly what was predicted, but this is just a statistical estimate, so it is a good starting point.

Run the frequency analysis script on **treasureisland.txt** and **twocities.txt**. These are the first chapters of Treasure Island, by Robert Louis Stevenson, and A Tale of Two Cities, by Charles Dickens, respectively.

Observe how close the analysis holds up across these three texts.

Analysis Methodology

Open the text file **ts1.txt** in a text editor. This is a text that is encrypted using a simple substitution cipher. Identify the text (hint: If successful, please teach the instructors and classmates how this was done!).

Open **ts2.txt** in a text editor. This is the exact same encryption, using the same substitution, but keeping the original punctuation. This would be far easier to decipher, but generally, a cryptanalyst does not have access to the original text. But what if it was available? Open **tomsawyer.txt** and compare this to **ts1.txt** and **ts2.txt**. Obviously, this is the same text. In the encrypted text, a B is substituted for the T, W for the O, U for the M, and so on. Comparing the encrypted and plaintext in a substitution cipher makes the key, or mapping algorithm, easier to see. In fact, it is almost trivial.

What if the plaintext were not available and the pattern needed to be discerned from the frequency analysis?

Run **ts1.txt** through the frequency analysis script. The first four lines of the results are displayed here:

The letter **I** occurred 1062 times, which is 11.0590%
The letter **B** occurred 939 times, which is 9.7782%
The letter **W** occurred 830 times, which is 8.6431%
The letter **E** occurred 746 times, which is 7.7684%
In the encrypted text, the letter **I** occurs most frequently. Based on our earlier analysis, what letter should this correspond to?

E

The letter **B** is the second most frequent so corresponds to the letter **T**, by expectation. Since the original text allowed us to already know this to be true, this confirms that this analysis is on the right track.

For each letter in the frequency analysis output, map it to the expected letter based on the known analysis table. In other words:

I = E
B = T
W = A

and so on.

Open **decoder.py** and look at the mapper variable. This is a Python Dictionary object.

The dictionary allows the association of a key to a value, called a key-value pair. This will be used to help decode the **ts1.txt** to the original text.

Run the script and when it asks for a filename, give it **ts1.txt**. What came out? Currently, the mapping is the letter E to the letter E, T to T, A to A, etc., so this will not change the text at all.

A **#** in Python code denotes a comment. Comments can be used to explain the code, or to prevent code from being executed.

In the **decoder.py** file, the # prevents the filename from being automatically specified. Remove the initial comment character (#) on the second line:

```
filename=""
#filename = "ts1.txt"    #Uncomment this to specify a file - useful for testing with the same input file
mapper = {}   # Define an empty dictionary object
```

This will prevent having to retype **ts1.txt** for the input every time. To use the **decoder.py** for analysis of other texts, comment it again.

Run this script now. It should not ask for the file name, and the output should be the same as before.

Now for the analysis. Based on the frequency analysis, it is safe to assume that the letter **I** is actually an **E**, so change that in the **decoder.py** script.

Change the E inside the bracket to an I, the T to a B, the A to a W, etc., based on our frequency above.

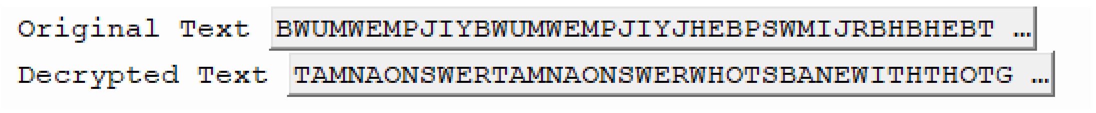
```
mapper['I']='E'
mapper['T']='B'
mapper['W']='A'
```

and so on.

The script will parse through the inputted text and replace every occurrence of the letter I with the letter E: the value of the dictionary. It is a key-value mapping I->E, T->B, W->A, and so on.

Make sure **all 26 letters** are accounted for. There cannot be any repeats inside the brackets or the substitutions will not work correctly.

Once this is completed, the result should be (from Thonny):


```
Original Text  BWUMWEMPJIYBWUMWEMPJIYJHEBPSWMIJRBHBHEBT ...
Decrypted Text  TAMNAONSWERTAMNAONSWERWHOTSBANEWITHTHOTG ...
```

Notice this isn’t correct. The first three letters should be **TOM**, not **TAM**.

What does this mean?

The analysis is not 100% correct and more changes need to occur. Right now, the **W** is mapped to an **A** and should be mapped to an **O**.

Any changes to the mapping needs to be done in pairs. Making a single change will cause duplicate letters in the keys.

Swap the **W** and **E** keys so that the **W** maps to an **O** and the **E** maps to the **A**.

```
mapper['E']='A'    # Expected 8.2%
mapper['W']='O'    # Expected 7.5%
```

Rerun the script. Is this closer? Are there any additional changes that need to make?

Yes! Refer to the **tomsawyer.txt** file to see what needs to change.
The **decoderTS.py** script has the correct mappings. But don’t go directly to this script; instead, try to work out the correct mappings in the **decoder.py** script. The correct mapping will not be given for the final analysis and will need to be created.

Substitution Analysis

The **Substitution.txt** file contains a substitution ciphertext. The first few letters are **RLCCB JUMVB LJCMA BJSJL DMLFA XJWJD BUXGV AICAX**. They are broken into 5 letter groups, so they provide no clues from the formatting.

To analyze this and break the encryption, do a frequency analysis.

Use the **decoder.py** script to try to break the code. Hint: It is another book on the list in the references section below.

Advanced Lab

Analyze the Python scripts presented in this lab and determine how they work. These scripts may show up in another lab.

References

All book references are used under the Public Domain and are found at
https://www.gutenberg.org/browse/scores/top

1

Multiple choice

2 points

Question at position 1
Which letter occurred most in the **substitution.txt** file, with 1210 instances?
E
J
L
A

Question at position 2

2

Multiple choice

2 points

Question at position 2
Which letter occurred least in the **substitution.txt** file, with 7 instances?
P
Q
N
O

Question at position 3

3

Multiple choice

5 points

Question at position 3

What was the plaintext?

Moby Dick; Or, The Whale by Herman Melville

The Adventures of Sherlock Holmes by Arthur Conan Doyle

Grimms' Fairy Tales by Jacob Grimm and Wilhelm Grimm

The Scarlet Letter by Nathaniel Hawthorne