Monoalphabetic Substitution Ciphers using Letter Frequency Analysis in Python

ddas.tech/monoalphabetic-substitution-ciphers-using-letter-frequency-analysis-in-python/

September 13, 2022

Created By: Debasis Das (12-Sep-2022)

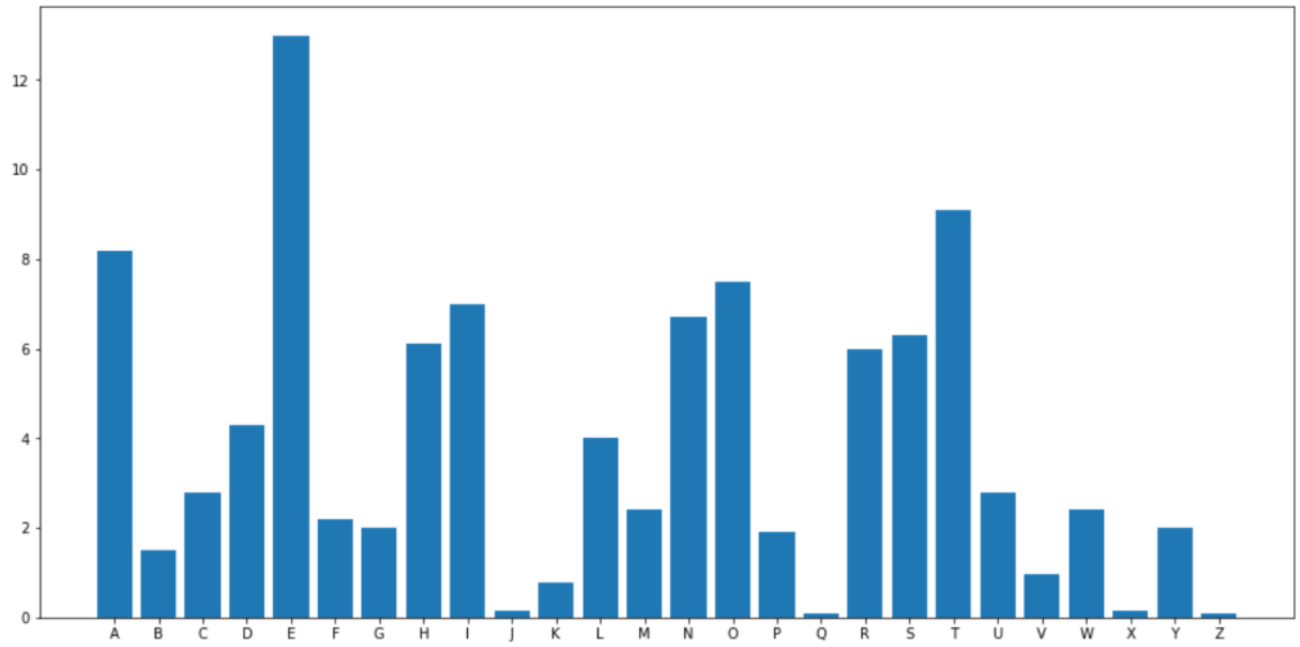
Substitution ciphers are one of the most common form of ciphers used through out the history. There are common ciphers such as Caesar (very easily breakable) and Vigenere Cipher.

Substitution ciphers work by replacing each letter of the plain text with another letter or symbol. The monoalphabetic substitution cipher uses a fixed replacement structure. that is the substitution is fixed for each letter of the alphabet.

In this post we will try to decipher a mono-alphabetic cipher text using **Letter Frequency Analysis**

Letter Frequency Analysis is based on counting the number of occurrences of each letter to identify the most recurrent letters

```
import matplotlib.pyplot as plt
# Relative frequency in english language
letter_freq_text = {
    'A':8.2,
    'B':1.5,
    'C':2.8,
    'D':4.3,
    'E':13,
    'F':2.2,
    'G':2,
    'H':6.1,
    'I':7,
    'J':0.15,
    'K':0.77,
    'L':4,
    'M':2.4,
    'N':6.7,
    'O':7.5,
    'P':1.9,
    'Q':0.095,
    'R':6,
    'S':6.3,
    'T':9.1,
    'U':2.8,
    'V':0.98,
    'W':2.4,
    'X':0.15,
    'Y':2,
    'Z':0.074
}
fig = plt.figure(figsize = (16, 8))
plt.bar(range(len(letter_freq_text)), list(letter_freq_text.values()),
tick_label=list(letter_freq_text.keys()))
plt.show()
```



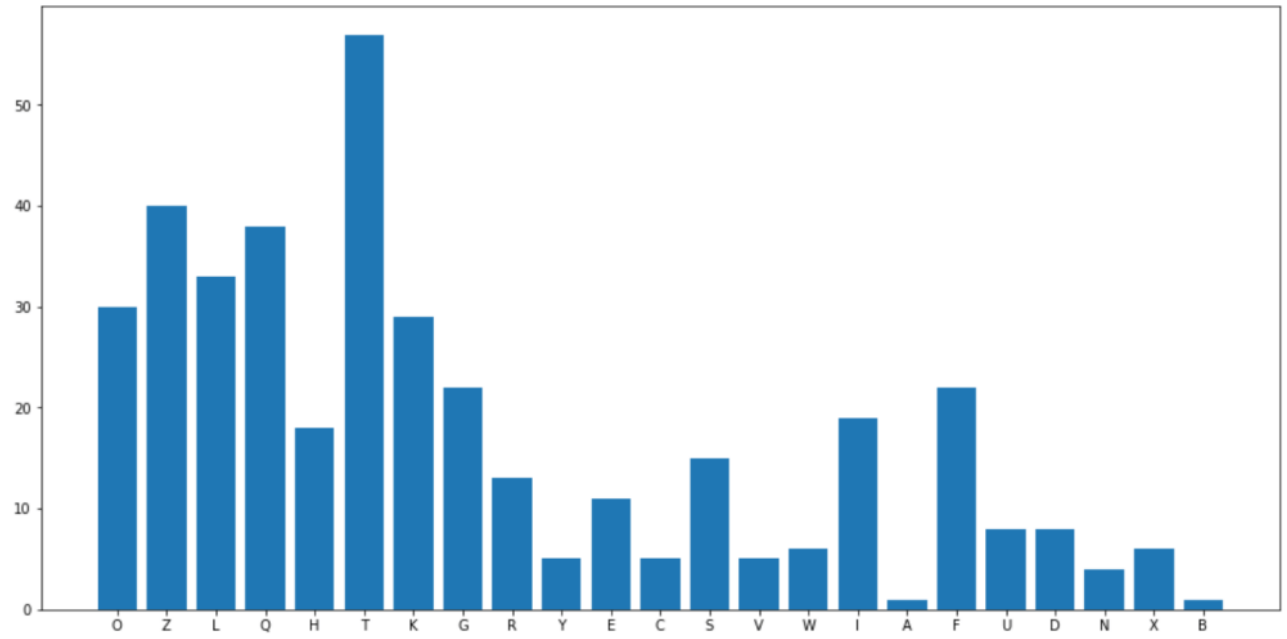
Relative Letter Frequency in English Language

Lets start with with the below cipher text and we will use letter frequency analysis to try de-cipher the text.

```
OZ OL Q HTKOGR GY EOCOS VQK. KTWTS LHQETLIOHL, LZKOAOFU YKGD Q IORRTF WQLT,
IQCT VGF ZITOK YOKLZ COEZGKN QUQOFLZ ZIT TCOS UQSQEZOE TDHOKT. RXKOFU ZIT WQZZST,
KTWTS LHOTL DQFQUTR ZG LZTQS LTEKTZ HSQFL ZG ZIT TDHOKT'L XSZODQZT VTQHGF,
ZIT RTQZI LZQK, QFR LHQET LZQZOGF VOZI TFGXUI HGVTK ZG RTLZKGN QF TFZOKT HSQFTZ.
HXKLXTR WN ZIT TDHOKT'L LOFOLZTK QUTFZL,
HKOFETLL STOQ KQETL IGDT QWGQKR ITK LZQKLIOH,
EXLZGROQF GY ZIT LZGSTF HSQFL ZIQZ EQF LQCT ITK HTGHST QFR KTLZGKT YKTTRGD ZG ZIT
UQSQBN
```

```
original_letters   = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
cipher_text = """
OZ OL Q HTKOGR GY EOCOS VQK. KTWTS LHQETLIOHL, LZKOAOFU YKGD Q IORRTF WQLT,
IQCT VGF ZITOK YOKLZ COEZGKN QUQOFLZ ZIT TCOS UQSQEZOE TDHOKT. RXKOFU ZIT WQZZST,
KTWTS LHOTL DQFQUTR ZG LZTQS LTEKTZ HSQFL ZG ZIT TDHOKT'L XSZODQZT VTQHGF,
ZIT RTQZI LZQK, QFR LHQET LZQZOGF VOZI TFGXUI HGVTK ZG RTLZKGN QF TFZOKT HSQFTZ.
HXKLXTR WN ZIT TDHOKT'L LOFOLZTK QUTFZL,
HKOFETLL STOQ KQETL IGDT QWGQKR ITK LZQKLIOH,
EXLZGROQF GY ZIT LZGSTF HSQFL ZIQZ EQF LQCT ITK HTGHST QFR KTLZGKT YKTTRGD ZG ZIT
UQSQBN
"""
cipher_letter_freq = {}
for char in cipher_text:
    if char in original_letters:
        if char in cipher_letter_freq:
            cipher_letter_freq[char] += 1
        else:
            cipher_letter_freq[char] = 1

fig = plt.figure(figsize = (16, 8))
plt.bar(range(len(cipher_letter_freq)), list(cipher_letter_freq.values()),
tick_label=list(cipher_letter_freq.keys()))
plt.show()
```



Letter Frequency in the Cipher Text

```
# In the below block of code we will create the placeholder string where we will
start substitution
temp_text = cipher_text
for item in original_letters:
    temp_text = temp_text.replace(item,'-')

print(cipher_text)
print(temp_text)
```

```
OZ OL Q HTKOGR GY EOCOS VQK. KTWTS LHQETLIOHL, LZKOAOFU YKGD Q IORRTF WQLT,
IQCT VGF ZITOK YOKLZ COEZGKN QUQOFLZ ZIT TCOS UQSQEZOE TDHOKT. RXKOFU ZIT WQZZST,
KTWTS LHOTL DQFQUTR ZG LZTQS LTEKTZ HSQFL ZG ZIT TDHOKT'L XSZODQZT VTQHGF,
ZIT RTQZI LZQK, QFR LHQET LZQZOGF VOZI TFGXUI HGVTK ZG RTLZKGN QF TFZOKT HSQFTZ.
HXKLXTR WN ZIT TDHOKT'L LOFOLZTK QUTFZL,
HKOFETLL STOQ KQETL IGDT QWGQKR ITK LZQKLIOH,
EXLZGROQF GY ZIT LZGSTF HSQFL ZIQZ EQF LQCT ITK HTGHST QFR KTLZGKT YKTTRGD ZG ZIT
UQSQBN


-- -- - ------ -- ----- ---. ----- ----------, -------- ---- - ------ ----,
---- --- ----- ----- ------- ------- --- ---- -------- ------. ------ --- ------,
----- ----- ------- -- ----- ------ ----- -- --- ------'- -------- ------,
--- ----- ----, --- ----- ------- ---- ------ ----- -- ------- -- ------ ------.
------- -- --- ------'- -------- ------,
-------- ---- ----- ---- ------ --- --------,
--------- -- --- ------ ----- ---- --- ---- --- ------ --- ------- ------- -- --- ---
---
```

From the cipher text letter frequency analysis we observed the following

- From the Cipher Text Letter Frequency Analysis 'T' is the most frequently used letter # T should correspond to E which has the highest frequency in the dictionary and free text
- we identify there is a single occurrence of letter Q, Highly likely it must be A
- We also replace the second most frequent letter Z with T

```
temp_text_arr = list(temp_text)
for index,item in enumerate(cipher_text):
# From the Cipher Text Letter Frequency Analysis it 'T' is the most frequently used
letter # T should correspond to E
    if item == 'T':
        temp_text_arr[index] = 'E'
# We see there is a single letter occurence and most probably it is going to be A
    if item == 'Q':
        temp_text_arr[index] = 'A'

# Second most frequent letter in Text is T and in the cipher is Z, Lets replace Z
with T and see
    if item == 'Z':
        temp_text_arr[index] = 'T'


final_plain_text = "".join(temp_text_arr)
print(final_plain_text)
```

```
-T -- A -E---- -- ----- -A-. -E-E- --A-E-----, -T------ ---- A ----E- -A-E,
-A-E --- T-E-- ----T ---T--- A-A---T T-E E--- -A-A-T-- E----E. ------ T-E -ATT-E,
-E-E- ---E- -A-A-E- T- -TEA- -E--ET --A-- T- T-E E----E'- --T--ATE -EA---,
T-E -EAT- -TA-, A-- --A-E -TAT--- --T- E----- ---E- T- -E-T--- A- E-T--E --A-ET.
-----E- -- T-E E----E'- -----TE- A-E-T-,
-----E-- -E-A -A-E- ---E A--A-- -E- -TA-----,
---T---A- -- T-E -T--E- --A-- T-AT -A- -A-E -E- -E---E A-- -E-T--E --EE--- T- T-E -A-
A--
```

- So far we have replaced three letter and things look positive
- From the cryptanlysis that we have done so far looks like **T*** is probably **TO**
- **T*E** is probably **THE**

```
temp_text_arr = list(final_plain_text)
for index,item in enumerate(cipher_text):
    if item == 'G':
        temp_text_arr[index] = 'O'
    if item == 'I':
        temp_text_arr[index] = 'H'


fpt1 = "".join(temp_text_arr)
print(fpt1)
```

```
-T -- A -E--O- O- ----- -A-. -E-E- --A-E-H---, -T------ --O- A H---E- -A-E,
HA-E -O- THE-- ----T ---TO-- A-A---T THE E--- -A-A-T-- E----E. ------ THE -ATT-E,
-E-E- ---E- -A-A-E- TO -TEA- -E--ET --A-- TO THE E----E'- --T--ATE -EA-O-,
THE -EATH -TA-, A-- --A-E -TAT-O- --TH E-O--H -O-E- TO -E-T-O- A- E-T--E --A-ET.
-----E- -- THE E----E'- -----TE- A-E-T-,
-----E-- -E-A -A-E- HO-E A-OA-- HE- -TA--H--,
---TO--A- O- THE -TO-E- --A-- THAT -A- -A-E HE- -EO--E A-- -E-TO-E --EE-O- TO THE -A-
A--
```

Lets review the above text

- ***HA*E -> HAVE?***
- ***THE** -> THEIR, THERE, THESE ??***
- ***HO*E -> HOPE, HOME, HONE, HOLE, HOSE?***
- ***HE* -> HER?***
- ***EATH -> DEATH?***
- ***A** -> AND?***
- ***O* -> OF vs ON?***

```
temp_text_arr = list(fpt1)
for index,item in enumerate(cipher_text):
    if item == 'C':
        temp_text_arr[index] = 'V'
    if item == 'R':
        temp_text_arr[index] = 'D'
    if item == 'D':
        temp_text_arr[index] = 'M'
    if item == 'K':
        temp_text_arr[index] = 'R'
    if item == 'L':
        temp_text_arr[index] = 'S'
    if item == 'O':
        temp_text_arr[index] = 'I'

fpt2 = "".join(temp_text_arr)
print(fpt2)
```

```
IT IS A -ERIOD O- -IVI- -AR. RE-E- S-A-ESHI-S, STRI-I-- -ROM A HIDDE- -ASE,
HAVE -O- THEIR -IRST VI-TOR- A-AI-ST THE EVI- -A-A-TI- EM-IRE. D-RI-- THE -ATT-E,
RE-E- S-IES MA-A-ED TO STEA- SE-RET --A-S TO THE EM-IRE'S --TIMATE -EA-O-,
THE DEATH STAR, A-D S-A-E STATIO- -ITH E-O--H -O-ER TO DESTRO- A- E-TIRE --A-ET.
--RS-ED -- THE EM-IRE'S SI-ISTER A-E-TS,
-RI--ESS -EIA RA-ES HOME A-OARD HER STARSHI-,
--STODIA- O- THE STO-E- --A-S THAT -A- SAVE HER -EO--E A-D RESTORE -REEDOM TO THE -A-
A--
```

Some further Analysis and trial we can come up with the famous Starwars opening crawl text

IT IS A PERIOD OF CIVIL WAR. REBEL SPACESHIPS, STRIKING FROM A HIDDEN BASE,
HAVE WON THEIR FIRST VICTORY AGAINST THE EVIL GALACTIC EMPIRE. DURING THE BATTLE,
REBEL SPIES MANAGED TO STEAL SECRET PLANS TO THE EMPIRE'S ULTIMATE WEAPON,
THE DEATH STAR, AND SPACE STATION WITH ENOUGH POWER TO DESTROY AN ENTIRE PLANET.
PURSUED BY THE EMPIRE'S SINISTER AGENTS,
PRINCESS LEIA RACES HOME ABOARD HER STARSHIP,
CUSTODIAN OF THE STOLEN PLANS THAT CAN SAVE HER PEOPLE AND RESTORE FREEDOM TO THE GALAXY

Next Steps:

- Create an archive of 2 letters, 3 letters, 4 letters word and cross reference those programmatically to remove the human analysis part.
- Automate the entire analysis with no human deciphering necessary.
- Creating Cipher Text from Plain Text in Python

References

https://en.wikipedia.org/wiki/Letter_frequency