جامعة العلوم والتكنولوجيا
University of Science and Technology

# Communications and Information Engineering Program

Probability and Stochastic Processes (CIE 327)

Omar Wael Abdelmohsen Ayyad (202000806)

Project link:

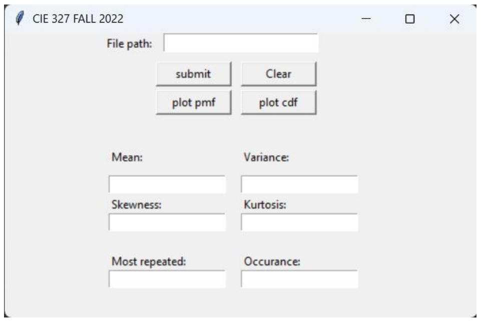
https://github.com/Omar3yyad25/Letters-Probabilty-Project

## 1. Introduction

First of all, this project is a feature of a data analysis procedure based on a probability research. The method for assessing data loaded in the form of files is critical since the results are used in various research, including scientific and literary objectives. This project is simply a basic code for analyzing a file that you can add through the place designated for it in the GUI, and it will display the necessary statistics and graphs.

## 2. Code Structure

The Cod is divided into two section which are the GUI as the following figure, and the implementation code which also consisted of 4 main function related to the 4 button provided in the GUI, the functions as following:



- `analysis()` : it is the main function for recalling rest of calculations functions and set the values which would be shown on the GUI in the specified fields.
- `clear()` : it is the function requested to clear all the variables values to start new analysis and set all the text boxes in the GUI equals to zero

```
my_text_box.delete("1.0","end")
mean.set(0)
var.set(0)
ske.set(0)
kurt.set(0)
most_letter.set(0)
most_letterOc.set(0)
```

- `show_pmf()` : it is the function required to plot our PMF graph, which takes the values of probabilities for each number and letter and start plotting it using embedded library called **matplot.**

```
names = list(original_dic.keys())
values = list(prob_dic.values())
plt.bar(range(len(prob_dic)), values, tick_label=names)
```

- `show_cdf()` : it is the function required to plot our CDF graph, using **numpy** library and function called **cumsun()** which record the CDF values and plot as the previous PMF.

## **3. Managing Data**

As the data in the file provided not filtered as wanted, there was a mission to filter the text by using embedded function in python called **`isalnum()`** which make your text only have letters and numbers.

```
text = ''.join(e for e in string if e.isalnum())
d = Counter(''.join(text))
```

Unfortunately, the file may have another strange letters like **á , é** which cannot be caught by the previous function. Otherwise, the following solution helped us to solve the problem by making constant dictionary and the **if condition** in the analysis to deal with the only items' key provided in it.

```
original_dic = {'0': 0,'1': 0,'2': 0,'2': 0,'3': 0,'4': 0,'5': 0,'6': 0,'7': 0,'8': 0
,'9': 0 ,'a': 0,'A': 0,'b': 0,'B': 0,'c': 0,'C': 0,'d': 0,'D': 0,'e': 0,'E': 0 ,'f':
0,'F': 0,'g': 0,'G': 0,'h': 0,'H': 0,'i': 0,'I': 0,'j': 0,'J': 0,'k': 0 ,'K': 0 ,'l':
0,'L': 0,'m': 0,'M': 0,'n': 0,'N': 0,'o': 0,'O': 0,'p': 0,'P': 0 ,'q': 0,'Q': 0,'r':
0,'R': 0,'s': 0,'S': 0,'t': 0,'T': 0,'u': 0,'U': 0,'v': 0 ,'V': 0,'w': 0,'W': 0,'x':
0,'X': 0,'y': 0,'Y': 0,'z': 0,'Z': 0}
prob_dic = {'0': 0,'1': 0,'2': 0,'2': 0,'3': 0,'4': 0,'5': 0,'6': 0,'7': 0,'8': 0
,'9': 0 ,'a': 0,'A': 0,'b': 0,'B': 0,'c': 0,'C': 0,'d': 0,'D': 0,'e': 0,'E': 0 ,'f':
0,'F': 0,'g': 0,'G': 0,'h': 0,'H': 0,'i': 0,'I': 0,'j': 0,'J': 0,'k': 0 ,'K': 0 ,'l':
0,'L': 0,'m': 0,'M': 0,'n': 0,'N': 0,'o': 0,'O': 0,'p': 0,'P': 0 ,'q': 0,'Q': 0,'r':
0,'R': 0,'s': 0,'S': 0,'t': 0,'T': 0,'u': 0,'U': 0,'v': 0 ,'V': 0,'w': 0,'W': 0,'x':
0,'X': 0,'y': 0,'Y': 0,'z': 0,'Z': 0}
```

```
for l in letters:
     if l in original_dic:
         original_dic[l] = letters[l]
```

The concept of dictionary is as follows, **`dictionary name = {'key' : value}.`**

Then as having two dictionaries called **`original_dic`** and **`prob_dic`** this allowed us to have different ways to get your calculation

- **`original_dic`** : it is to record the number of occurrence of each letter as the value of each letter and number by the following code which return another unfiltered dictionary and start filter it again by the if condition mentioned above ,

```
def word_freq(string):
    text = ''.join(e for e in string if e.isalnum())
    d = Counter(''.join(text))
    sorted_keys = sorted(dict(d).keys())
    sorted_dict = {key:dict(d)[key] for key in sorted_keys}
    return (sorted_dict)
```

جامعة العلوم والتكنولوجيا
University of Science and Technology

This dictionary allowed us to calculate the most repeated letter and its number of occurrence and set it on the GUI interface by using **max()** function provided from **operator** library and its code as follows,

```
most_letter.set(max(original_dic.items(), key=operator.itemgetter(1))[0])
most_letterOc.set(max(original_dic.items(), key=operator.itemgetter(1))[1])
```

**prob_dic**: it is required to store the probability for each key as it is have the same keys in the another dictionary. Knowing that the probability of the letter or the number is : # of occurrence / sum of letters and number. After that, the result of the equation will be stored as the value of the dictionary,

```
sumLetters = sum(original_dic.values())
for n in original_dic:
    prob_dic[n] = original_dic[n]/sumLetters
```

## 4. Probability

As requested, the mean, variance, skewness, and kurtosis, there are 4 function with following names **calc_mean() , calc_var(), calc_kurt(), calc_ske()** to define each objective and return by the value.

General case: using a variable called counter to define the index the used item in the dictionary.

**calc_mean()**: as we know that the equation of the mean is

$$E(X) = \mu = \sum_x x\,f(x)$$

the implementation of the code depended on the equation and used an array called **probability_mean** and get the sum of it as the end.

```
counter =  0
for i in prob_dic:
   x= counter * prob_dic[i]
   counter = counter+1
   probabilty_mean.append(x)
mean_1 = sum(probabilty_mean)
return mean_1
```

**calc_var():** this function is used to calculate the variance and we know its formula as following

$$Var(X) = E(X^2) - E(X)^2$$

getting the excepted if $X^2$ and stored its value in array called **probability_mean2** and get its sum and minus the mean squared from it as following

```
counter = 0
for j in prob_dic:
   y= pow(counter,2)*prob_dic[j]
   probabilty_mean2.append(y)
   counter = counter +1
var_1 = sum(probabilty_mean2) - pow(sum(probabilty_mean),2)
return var_1
```

**calc_kurt():** as the previous, getting the excepted of X power 4 and store it in **probability_mean4** and get the kurtosis, but with different handling as the dominator should not equals to zero,

$$kurt(x) = \frac{E(X^4) - 4\mu E(X^3) + 6\mu^2\sigma^2 + 3\mu^4}{\sigma^4}$$

```
counter = 0
for j in prob_dic:
   y= pow(counter,4)*prob_dic[j]
   probabilty_mean4.append(y)
   counter = counter +1

dom = pow(math.sqrt(sum(probabilty_mean2) - pow(sum(probabilty_mean),2)),4)
if (dom != 0):
    kurt_1 = (sum(probabilty_mean4)-
4*(sum(probabilty_mean))*(sum(probabilty_mean3))+6*(pow(sum(probabilty_mean),
2))*((sum(probabilty_mean2) -
pow(sum(probabilty_mean),2)))+3*(pow(sum(probabilty_mean),4)))/pow(math.sqrt(
sum(probabilty_mean2) - pow(sum(probabilty_mean),2)),4)
    return kurt_1
else:
    return ("None")
```

**calc_ske():** as the previous, getting the excepted of X power 4 and store it in **probability_mean3** and get the skewness,

$$skew(x) = \frac{E(X^3) - 3\mu\sigma^2 - \mu^3}{\sigma^3}$$

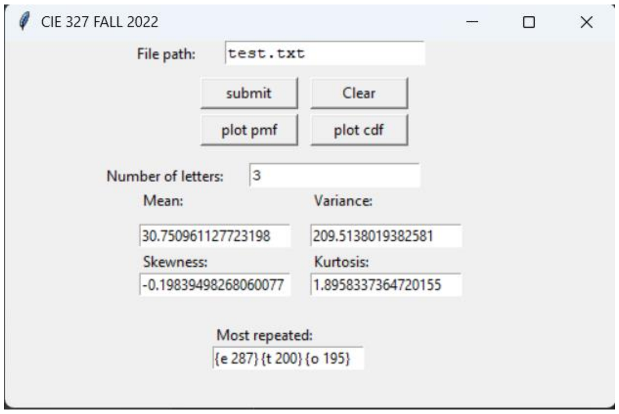
```
counter = 0
for j in prob_dic:
   y= pow(counter,3)*prob_dic[j]
   probabilty_mean3.append(y)
   counter = counter +1

dom = pow(math.sqrt(sum(probabilty_mean2) - pow(sum(probabilty_mean),2)),3)
```

```
if (dom != 0):
    ske_1 = (sum(probabilty_mean3) -
3*sum(probabilty_mean)*(sum(probabilty_mean2) - pow(sum(probabilty_mean),2))-
pow(sum(probabilty_mean),3))/pow(math.sqrt(sum(probabilty_mean2) -
pow(sum(probabilty_mean),2)),3)
    return ske_1
else:
    return ("None")
```

## 5. Testing and Results

- Test 1:

Importing the sample file provided on with documents the GUI gave us the following information,



File path: test.txt

Number of letters: 3

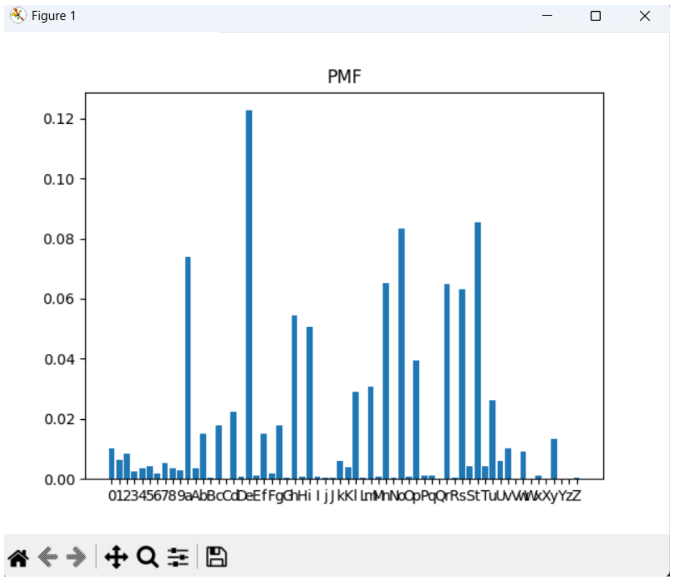
Mean: 30.750961127723198

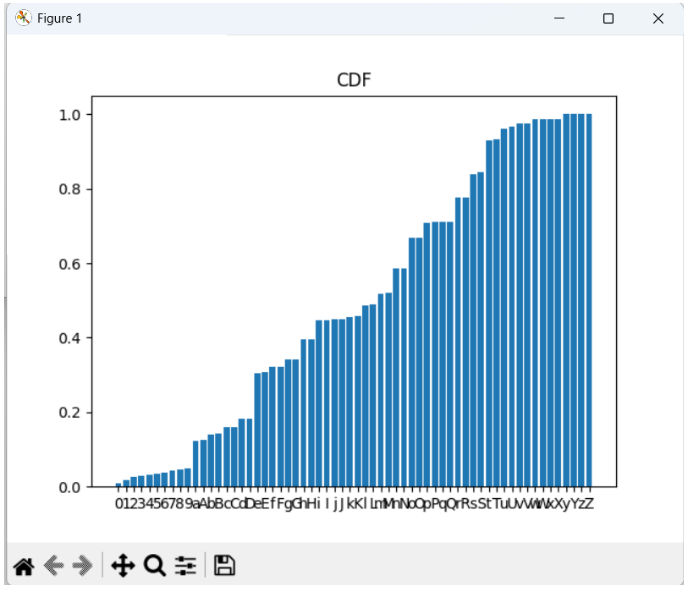
Variance: 209.5138019382581

Skewness: -0.19839498268060077

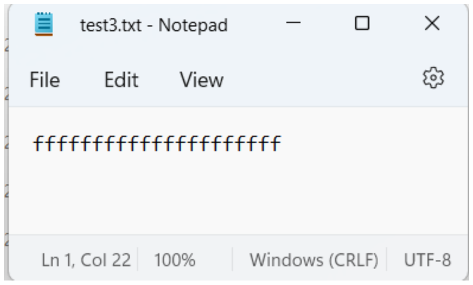
Kurtosis: 1.8958337364720155

Most repeated: {e 287} {t 200} {o 195}





- Test 2:
- Importing new file have only 1 letter by typed more than one time. The result as expected that the variance = 0 , skewness = none , and kurtosis = none

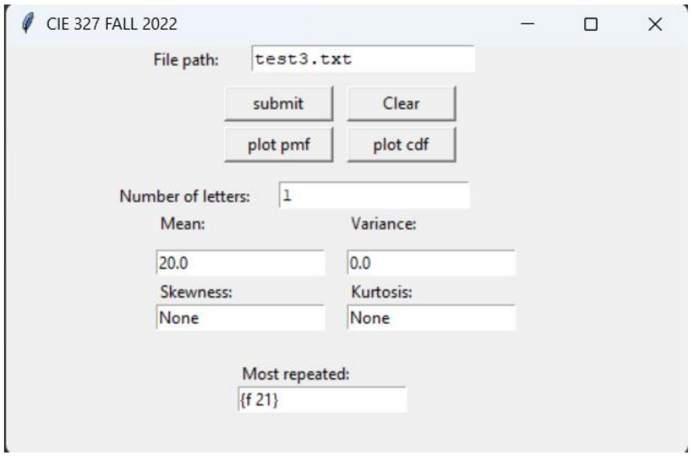


test3.txt - Notepad

File Edit View

```
fffffffffffffffffffff
```

Ln 1, Col 22 | 100% | Windows (CRLF) | UTF-8



CIE 327 FALL 2022

File path: test3.txt

Number of letters: 1

Mean: 20.0

Variance: 0.0

Skewness: None

Kurtosis: None

Most repeated: {f 21}

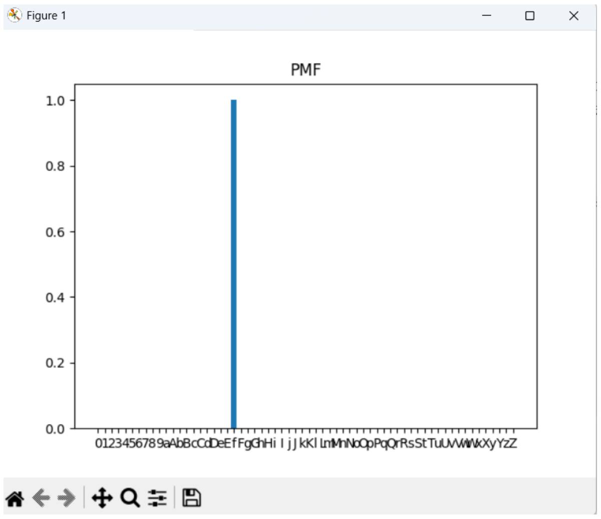


Figure 1
PMF
0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ

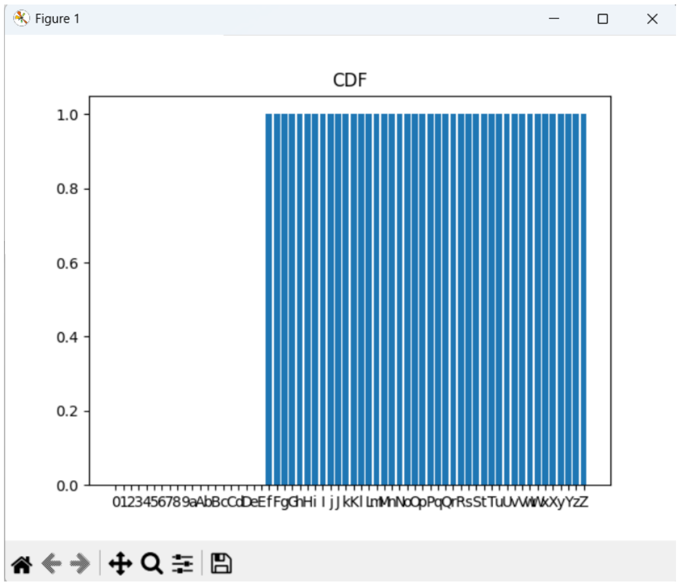


Figure 1
CDF
0123456789aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ