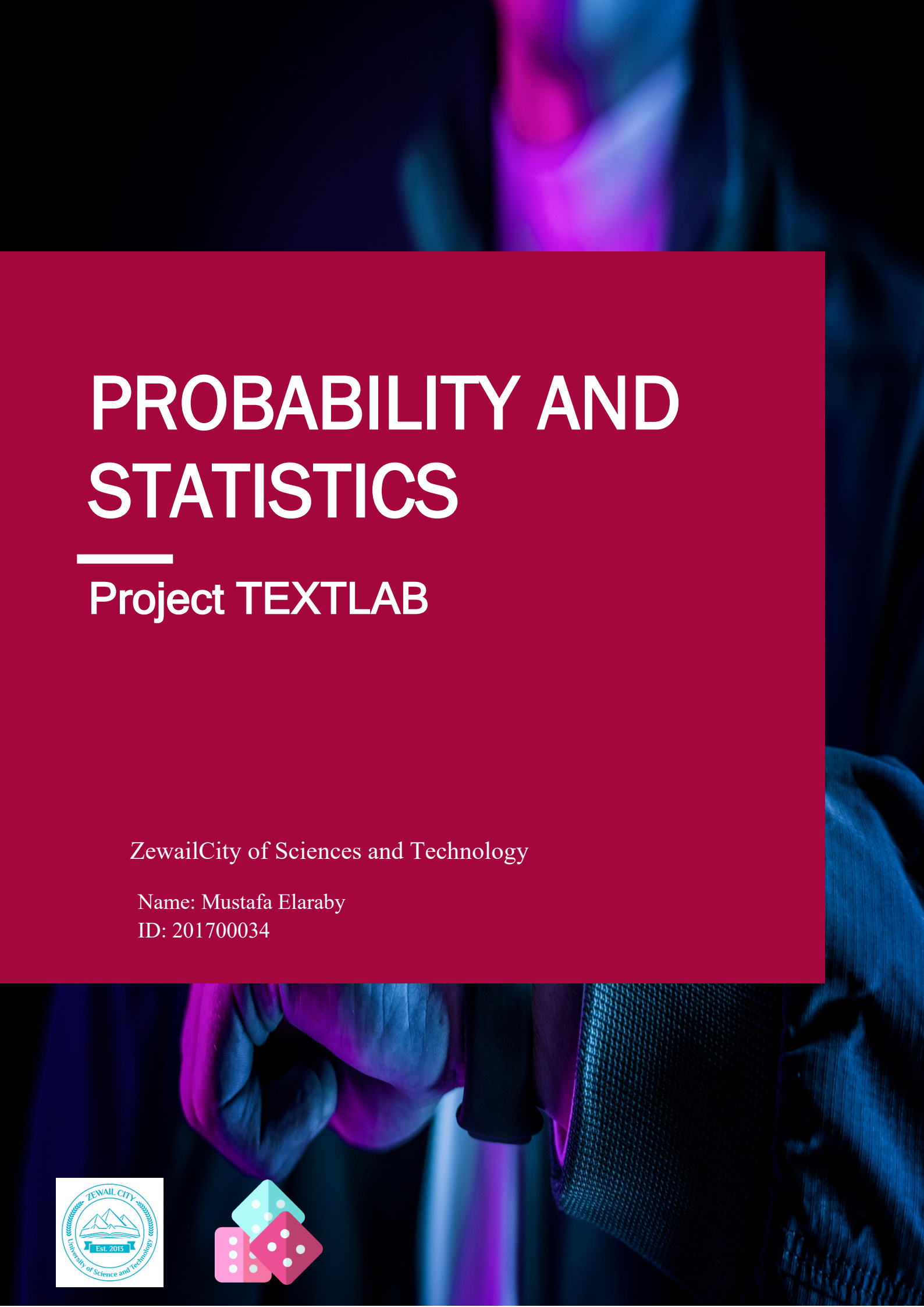


PROBABILITY AND STATISTICS

Project TEXTLAB

ZewailCity of Sciences and Technology

Name: Mustafa Elaraby
ID: 201700034

TEXTLAB APPLICATION

Dr. Samy S. Soliman, Eng. Amal Mohamed, Eng. MennatAllah Mohamed Hassan, Mustafa Elaraby.

⌘| Guidelines

❶ | Abstract.

This is a comprehensive report about TEXTLAB application. TEXTLAB is a GUI-based tool that allows a user to add any text file and results in some statistics related to the text written in the file. it filters the text within the input text file and extract only the English alphabet case sensitive and numbers from 0 to 9, it assigns a decimal value for every character from 0 to 61, for example the letter 'a' is assigned a value of 10, 'A' is assigned a value of 11, 'b' is assigned a value of 12 and 'B' is assigned a value of 13, and so on till 'Z' assigned a value of 61. TEXTLAB takes the .txt file and plots the probability mass function, the cumulative distribution function, calculates the mean, variance, skewness and kurtosis of the random variable representing the sample text. It also calculates the most repeated character in the text sample, and the number of repetitions. It is a very good tool to analyze text samples.

❷ | Definitions and Basic Concepts.

Definition 1.1: The Probability Mass function.

The set of ordered pairs $(x, f(x))$ is a **probability function**, **probability mass function**, or **probability distribution** of the discrete random variable X if, for each possible outcome x,

1. $f(x) \geq 0$,
2. $\sum_x f(x) = 1$,
3. $P(X = x) = f(x)$.

Definition 1.2: The Probability Density function.

The **cumulative distribution function** $F(x)$ of a discrete random variable X with probability distribution $f(x)$ is

$$F(x) = P(X \leq x) = \sum_{t \leq x} f(t), \quad -\infty < x < \infty$$

Definition 1.3: The Expected value (mean).

If X has a discrete distribution with probability mass function $f(x)$ (so that S is countable), then the expected value of X is defined as follows (assuming that the sum is well defined):

$$E(X) = \mu = \sum_{x} x\, f(x)$$

Definition 1.4: The variance and standard deviation.

The variance and standard deviation of X are defined by

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

$$\sigma = \sqrt{Var(X)}$$

Definition 1.5: The skewness of X.

The skewness of X is the third moment of the standard score of X:

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

Definition 1.6: The Kurtosis of X.

The skewness of X is the third moment of the standard score of X:

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

❸ | Documentation.

❁ Properties.

Property	Documentation
ReferenceDecimalArray	Decimal representation of Chars and equals [0:62];
ReferenceCharactersArray	Array of Alphabet case sensitive and numbers from 0:9, and equals [0,1,2,..,9,a,A,b,B,..,z,Z], size = 62.
Filename	path to the text file referenced to parent location of pc.
Sample	filtered sample text array.
PMF	PMF array, size = 61, sum = 1;
CDF	CDF array, size = 61, CDF(62) = 1;
SZ	Size of sample.
MRC	Most repeated character.
NRMRC	Number of repetitions of MRC.
Mean	Mean of the sample.
Variance	Variance of the sample.
Sigma	Standard deviation of the sample.
Skew	Skewness of the sample.
Kurt	Kurtosis of the sample.

❁ Functions.

Function	Documentation
setInitialSatate(app)	returns all properties to initial values.
selectSample(app)	Prompt the user to select a text file to enter it as input.
filterSample(app,file)	takes the random sample text file as input and filters the text inside to extract the Alphabet and numbers from 0 to 9 and stores them in array.
getData(app,s)	takes the filtered array of the Alphabet and numbers from 0 to 9 as input and returns the app.PMF, app.CDF, app.SZ, app.MRC, app.NRMRC properties.
Moments(app,pmf)	takes the PMF array of a discrete random variable representing the letters and numbers sample and returns the mean, variance, sigma, skewness, and kurtosis of X.

❹ | Codes.

❁ Properties.

```
properties (Access = private)
    referenceDecimalArray = 0:61; % decimal representation of chars.
    referenceCharactersArray =
    ['0','1','2','3','4','5','6','7','8','9','a','A','b','B','c','C','d','D','e','E','
    f','F','g','G','h','H','i','I','j','J','k','K','l','L','m','M','n','N','o','O','p'
    ,'P','q','Q','r','R','s','S','t','T','u','U','v','V','w','W','x','X','y','Y','z','
    Z'];

    filename;       % path to the text file.
    N;             % Number of most reapeated Characters entered by the user.
    sample;         % filtered sample text.
    PMF;            % PMF array.
    CDF;            % CDF array.
    SZ;             % sample size.
    MRCs;           % Most Repeated Character.
    NRMRCs;         % Number of Repetition of Most Repeated Character.
    Mean;           % Mean of the sample.
    Variance;       % Variance of the sample.
    sigma;          % Standard deviation of the sample.
    skew;           % Skewness of the Sample
    kurt;           % Kurtosis of the sample.
End
```

❁ Functions.

```
methods (Access = private)
    function setInitialSatate(app)
        app.sample = zeros;
        app.filename = '';
        app.N = 0;
        app.PMF = zeros;
        app.CDF = zeros;
        app.SZ = 0;
        app.MRCs = 0;
        app.NRMRCs = 0;
        app.Mean = 0;
        app.Variance = 0;
        app.sigma = 0;
        app.skew = 0;
        app.kurt=0;
        app.StatusLamp.Color = [0.8 0.8 0.8];
        app.HTML.HTMLSource = "<p></p>";
        app.HTML2.HTMLSource = "<p></p>";
        app.meanDisplay.Value = 0;
        app.meanDisplay.FontColor = "white";
        app.varianceDisplay.Value = 0;
        app.varianceDisplay.FontColor = "white";
        app.skewnessDisplay.Value = 0;
        app.skewnessDisplay.FontColor = "white";
        app.kurtosisDisplay.Value = 0;
        app.kurtosisDisplay.FontColor = "white";
        cla(app.UIAxes);
        cla(app.UIAxes_2);
    end
```

```
function selectSample(app)
   [file,path] = uigetfile('*.txt');
   app.filename = string(path)+string(file);
end
```

```
function filterSample(app,file)

   %check if file exist.
   if exist(file,"file")
      text = fopen(file);

      % store the text in the txt file in a string array.
      str = fscanf(text,'%s');

      %filter the string array to get only alphapets and numbers.
      s=str((double(str) > 64 & double(str)<91 )|(double(str) > 96 & ...
      double(str)<123 )|(double(str) > 47 & double(str)<58 ));

      %check if filtered array is empty.
      if isempty(s)
         app.StatusLamp.Color = "blue";
         app.HTML2.HTMLSource="<html><head><style>p
         {color:blue;}</style></head><body><p>File contains no alphabet or
         numbers!</p></body></html>";
         app.sampleInput.FontColor = "blue";
         else

         %once filtered array is not empty assign it to
         %app.sample.
         app.sample = s;
      end

      else

      % if file doesn't exist tell the user to enter valid file.
      app.StatusLamp.Color = "red";
      app.HTML2.HTMLSource="<html><head><style>p {color:red;font-
      family:arial}</style></head><body><p>Enter a valid
      filename!</p></body></html>";
      app.sampleInput.FontColor = "red";
   end
end
```

```
function getData(app,s)

   % initialize an copy the app. referenceCharactersArray.
   X = app.referenceCharactersArray;

   %initialize a zeros array NREC to store the number of repeateition
   %for every character in the referenceCharactersArray.
   NREC = zeros([length(X)],1);

   %calculate number of repeatition in the sample text for every
   %character in the app.referenceCharactersArray.
   for i=1:length(X)
      for j=1:length(s)
         if X(i)==s(j)
            NREC(i) = NREC(i)+1;
         end
      end
```

```
    end

    %calculate PMF for every character = number of repeatition/
    %total number of the characters in the app.sample.
    app.PMF = NREC/length(s);

    %initialize a zeros array to store the CDF for every character
    %in the referenceCharactersArray.
    b = zeros([length(X)],1);

    %calculate CDF = summation(PMF).
    sum =0;
    for i=1:length(app.PMF)
        sum = sum+app.PMF(i);
        b(i) = sum;
    end

    %assign b matrix generated to app.CDF.
    app.CDF = b;

    %calculate app.SZ = Number of elements in the app.sample.
    app.SZ=length(s);

    % Get MRCs and NRMRCs.
    [M,I] = maxk(NREC,int8(app.N));

    c = "";
    d = "";

    for i=1:length(I)
        c(i)=X(I(i));
        d(i)=M(i);
    end
    app.MRCs = transpose(c);
    app.NRMRCs = transpose(d);
end

function Moments(app,pmf)

    % Calcultae Moments of the sample.
    E_X_1 = 0;
    E_X_2 = 0;
    E_X_3 = 0;
    E_X_4 = 0;

    for i=0:(length(pmf)-1)

        % First Moment = summation(i*PMF(i)).
        E_X_1 = E_X_1 + i*pmf(i+1);

        % Second Moment = summation(i^2*PMF(i)).
        E_X_2 = E_X_2 + (i^2)*pmf(i+1);

        % Thirdd Moment = summation(i^3*PMF(i)).
        E_X_3 = E_X_3 + (i^3)*pmf(i+1);

        % Fourth Moment = summation(i^4*PMF(i)).
        E_X_4 = E_X_4 + (i^4)*pmf(i+1);
    end
```

```
        % Calculate app.Mean.
        app.Mean = E_X_1;

        % Calculate app.Variance.
        app.Variance = ((E_X_2) - (app.Mean^2));

        % Calculate app.sigma.
        app.sigma = sqrt(app.Variance);

        % Calculate app.skewness.
        app.skew = ((E_X_3 - (3*app.Mean*app.Variance)-(app.Mean^3))/(app.sigma^3));

        % Calculate app.kurtosis.
        app.kurt = ((E_X_4 - (4*app.Mean*E_X_3) + (6*(app.Mean^2)*app.Variance)
        +(3*(app.Mean^4)))/(app.sigma^4));
    end
end
```

❁ Callbacks.

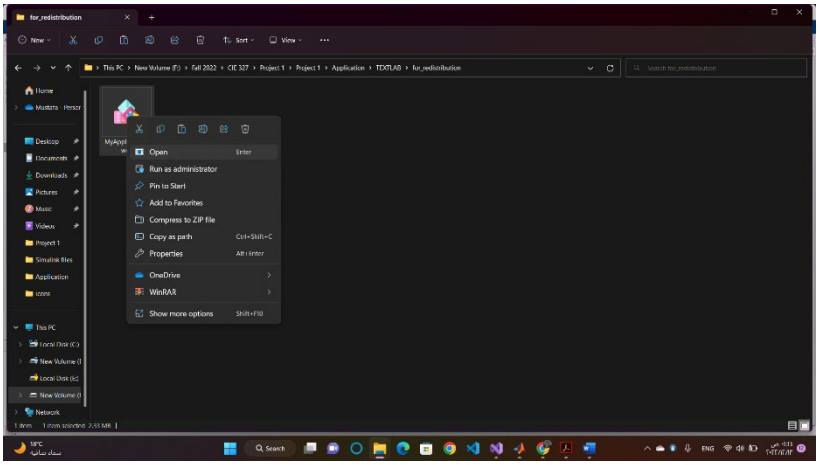
```
% Callbacks that handle component events
methods (Access = private)
    % Button pushed function: RunButton
    function RunButtonPushed(app, event)
        app.setInitialSatate();
        app.N = app.MRCInput.Value;

        if ismatrix(app.sampleInput.Value)
            app.filterSample(app.sampleInput.Value);
            app.getData(app.sample);
            app.Moments(app.PMF)

            if (ismatrix(app.PMF) && & int8(sum(app.PMF))==1)
                app.StatusLamp.Color = "green";
                app.meanDisplay.Value = app.Mean;
                app.meanDisplay.FontColor = [0 0 0];
                app.varianceDisplay.Value = app.Variance;

                if (app.Sigma~=0)
                    app.skewnessDisplay.Value=app.Skew;
                    app.skewnessDisplay.FontColor = [0 0 0];

                    app.kurtosisDisplay.Value=app.Kurt;
                    app.kurtosisDisplay.FontColor = [0 0 0];
                else
                    app.HTML.HTMLSource="<html><head><style>p
                    {color:red;}</style></head><body><p>Skewness and kurtosis are not
                    defined since variance = 0 </p></body></html>";
                end

                if (app.N>0)
                    tdata = table(char(app.MRCs),app.NRMRCs,'VariableNames',{'Character',
                    'Occurances'});
                    app.UITable.Data=tdata;
                    app.UITable.Visible=1;
                else
                    app.UITable.Visible=0;
                end
                bar(app.PMFGraph,app.referenceDecimalArray,app.PMF);
                plot(app.CDFGraph,app.referenceDecimalArray,app.CDF);
```

```
        end
    end
end

% Image clicked function: Image
function ImageClicked(app, event)
    app.selectSample();
    app.sampleInput.Value = app.filename;
    app.sampleInput.FontColor = [0 0 0];
    app.sampleInput.FontSize = 18;
end

% Button pushed function: ClearButton
function ClearButtonPushed(app, event)
    app.StatusLamp.Color = [0.8 0.8 0.8];
    app.sampleInput.Value = '';
    app.meanDisplay.Value = 0;
    app.meanDisplay.FontColor = "white";
    app.varianceDisplay.Value = 0;
    app.varianceDisplay.FontColor = "white";
    app.skewnessDisplay.Value = 0;
    app.skewnessDisplay.FontColor = "white";
    app.kurtosisDisplay.Value = 0;
    app.kurtosisDisplay.FontColor = "white";
    app.HTML.HTMLSource = "<p></p>";
    app.HTML2.HTMLSource = "<p></p>";
    app.UITable.Visible=0;
    cla(app.PMFGraph);
    cla(app.CDFGraph);
end

% Value changing function: sampleInput
function sampleInputValueChanging(app, event)
    changingValue = event.Value;

    if(changingValue)
        app.sampleInput.FontColor = [0 0 0];
        app.HTML.HTMLSource = "<p></p>";
        app.HTML2.HTMLSource = "<p></p>";

    end
end
end
```

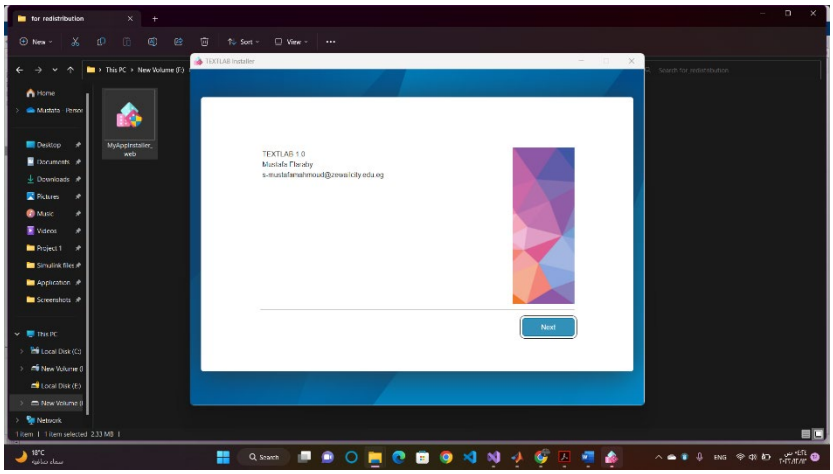
❺ | Setup and Run.

❁ Setup.

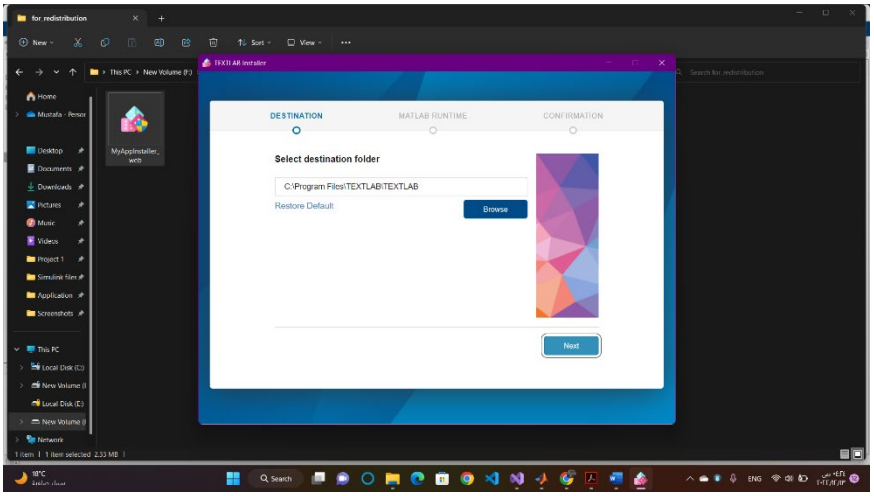
1. Right click in the TEXTlab-installer.exe and select open.



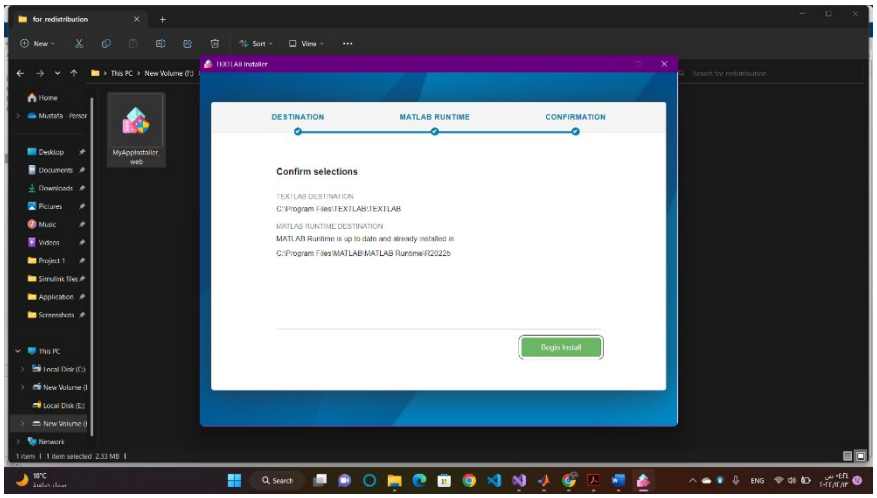
2. Select next.



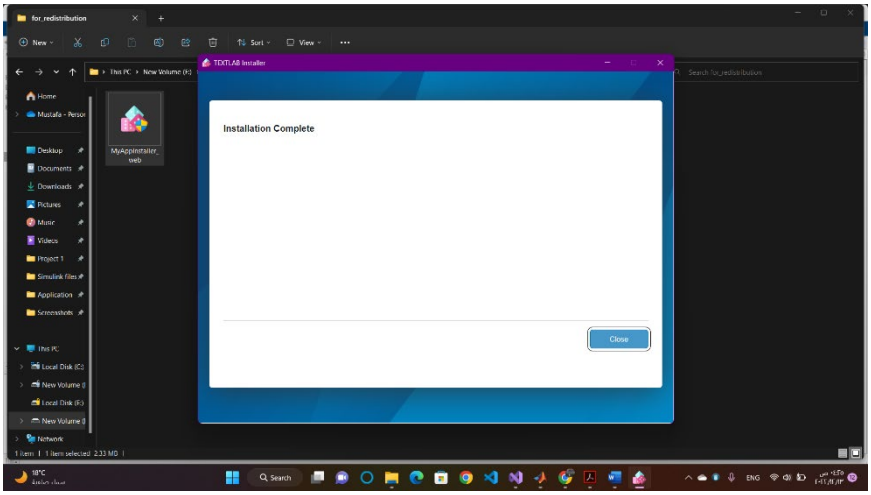
3. Choose the destination where you want setup your app, and select next.



4. Select begin install.

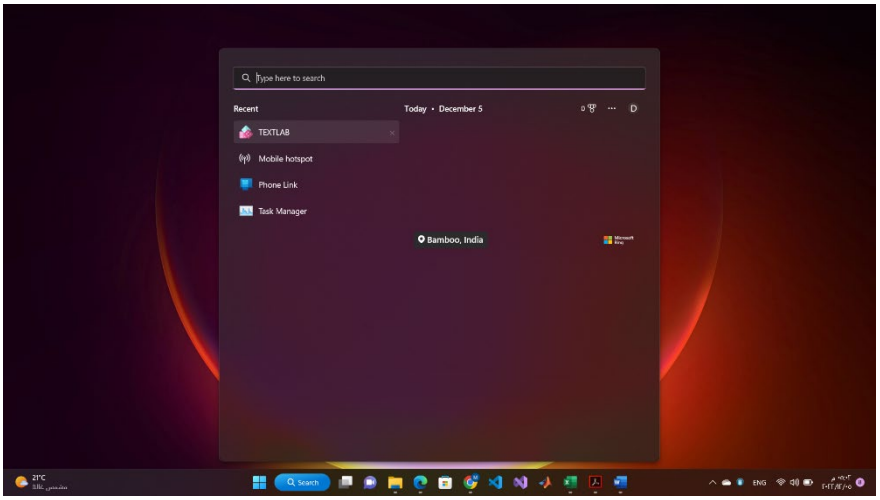


5. nstallation is complete click close.

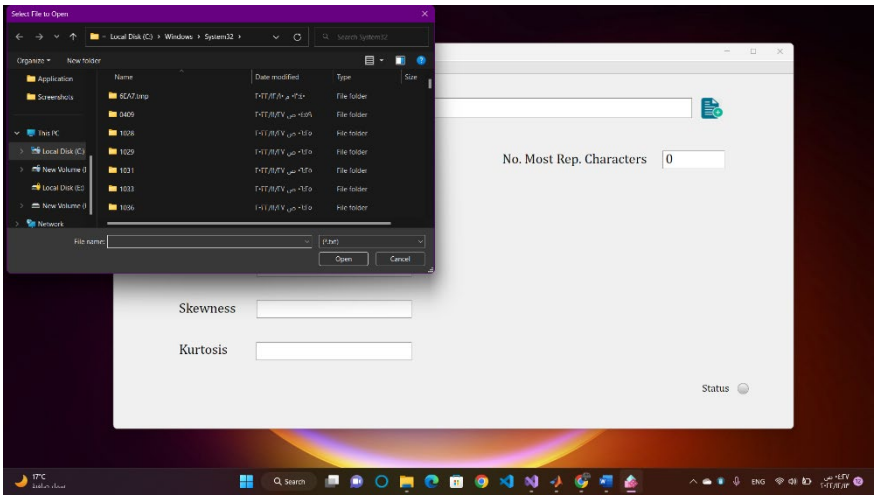


Run.

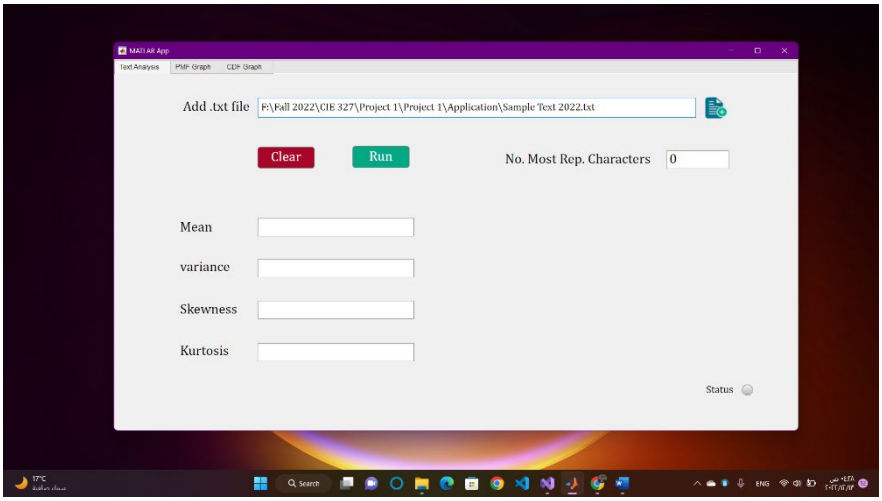
1. Open TEXTLAB from start menue.



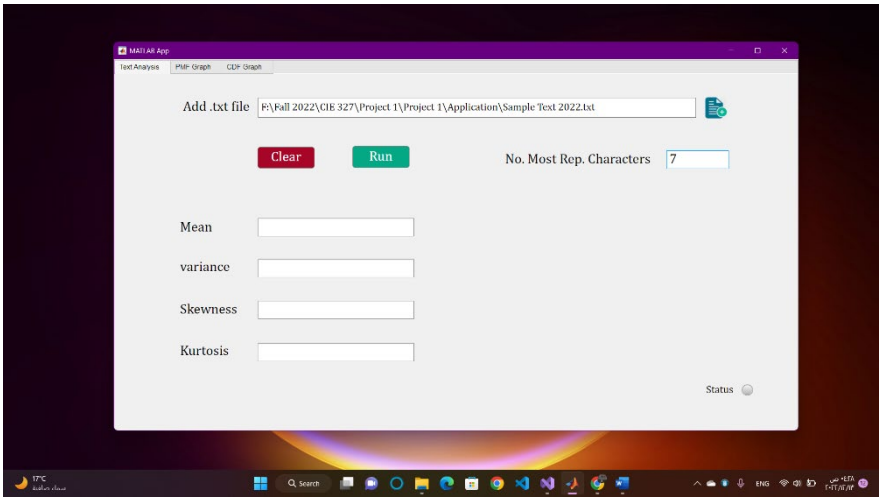
2. Click in the select file icon and choose a .txt file.



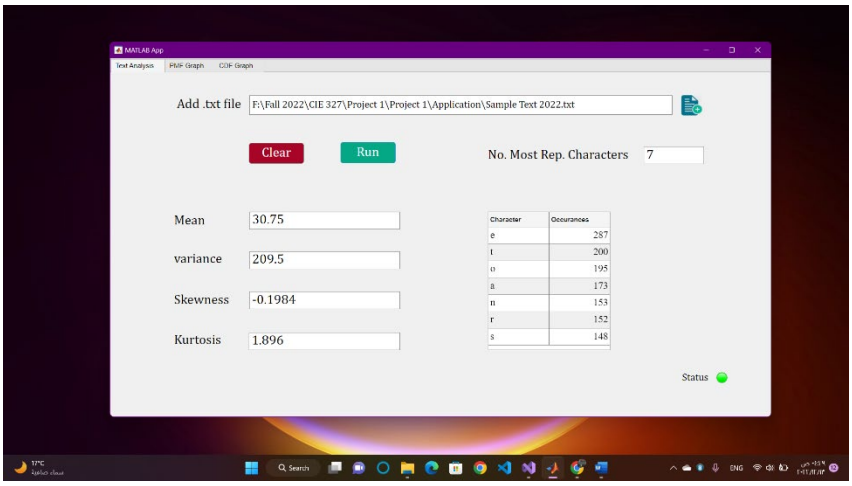
3. Or you can write the file path and name in the text field.



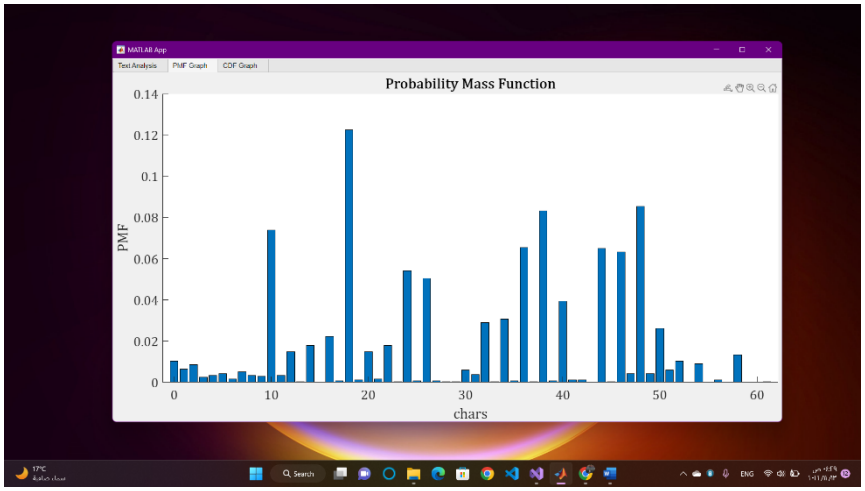
4. Enter the Number of most Repeated characters.



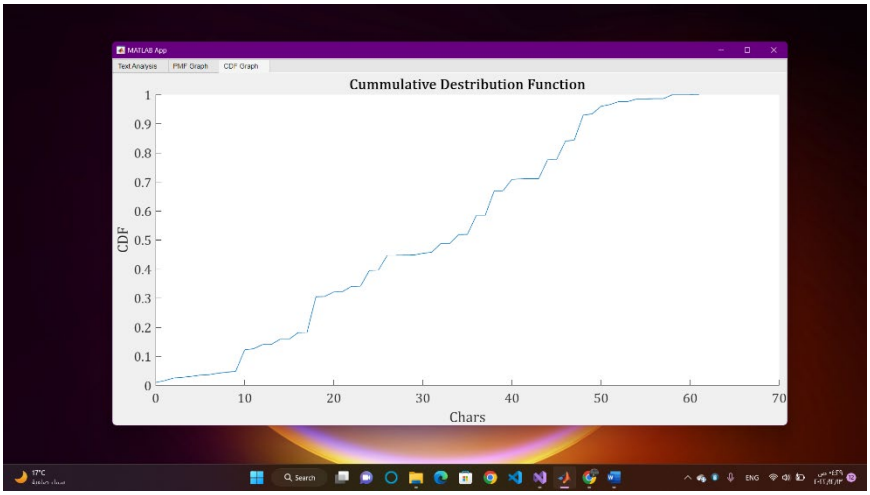
5. Click run and get your correct results and the status lamp will be green.



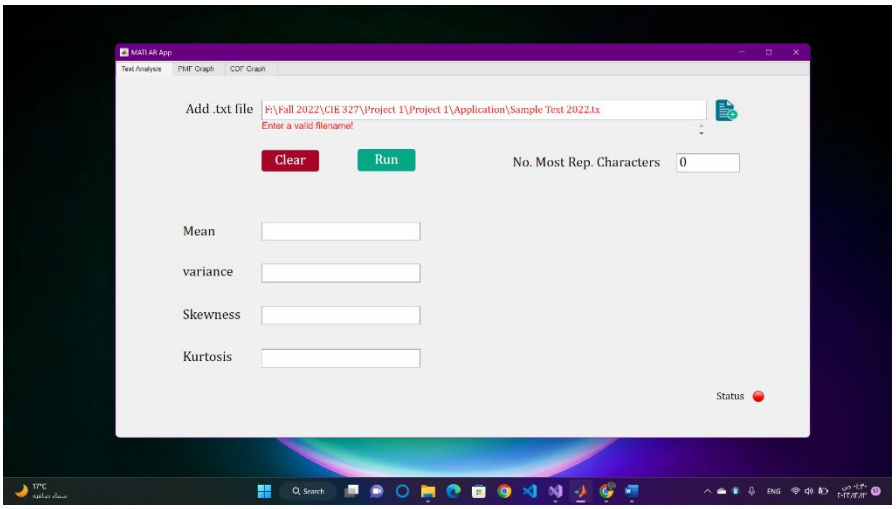
6. Click on PMF Graph to find PMF Plot.



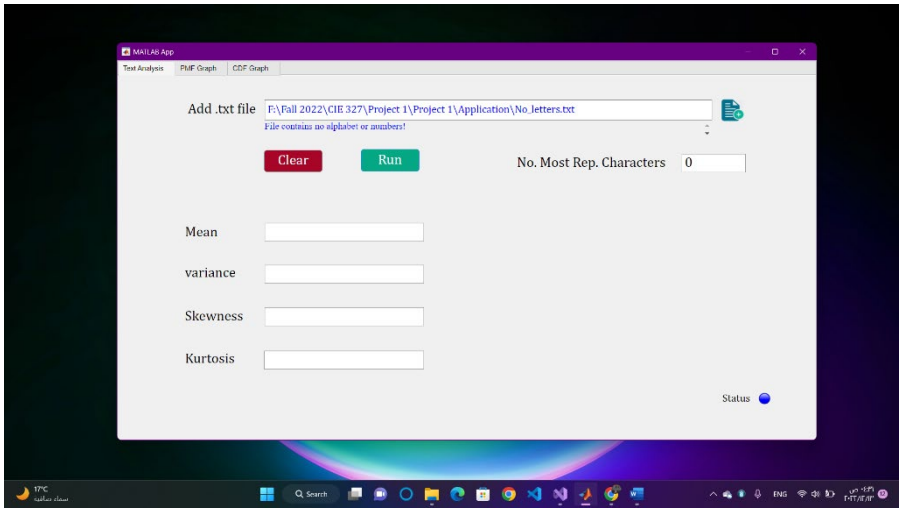
7. Click on CDF Graph to find CDF Plot.



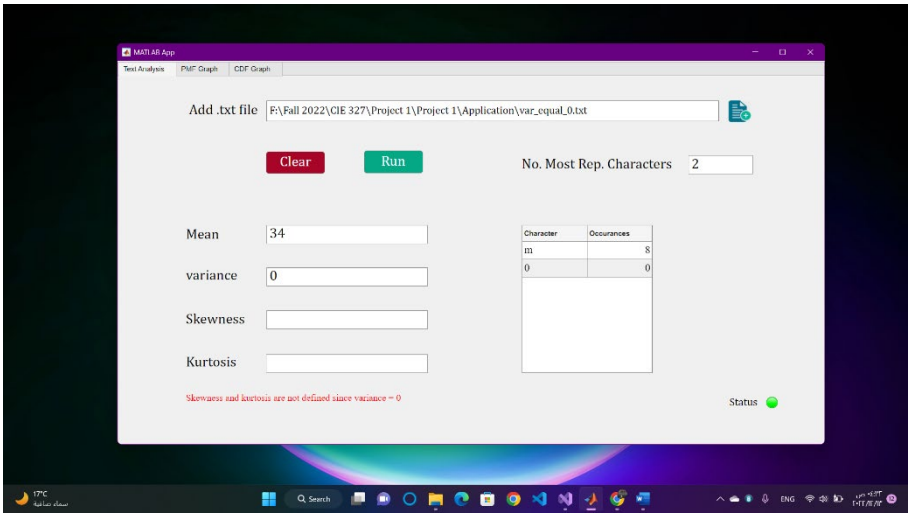
8. If you entered an invalid file path will get warning and the status lamp will be red.



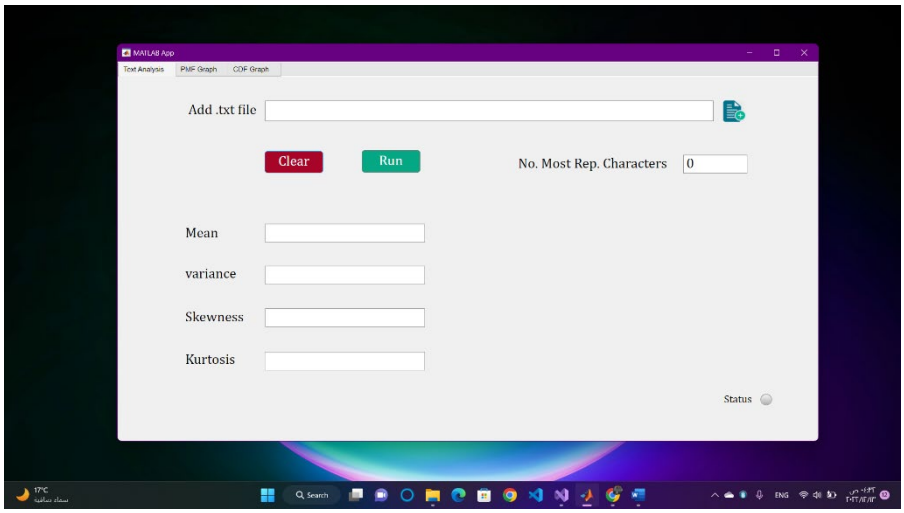
9. If you entered file that contains no alphabet or numbers you will get warning and the status lamp will be blue.



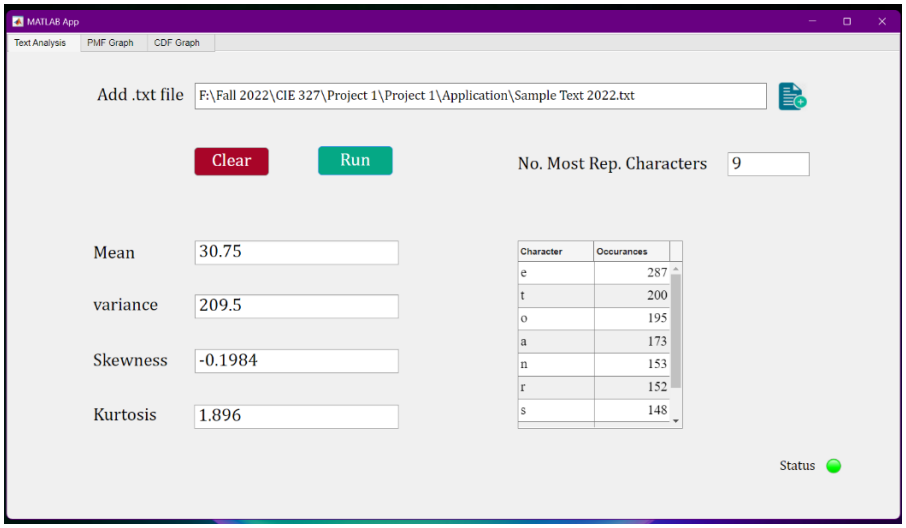
10. If the variance of the sample you will be notified that skew and kurt are not defined.

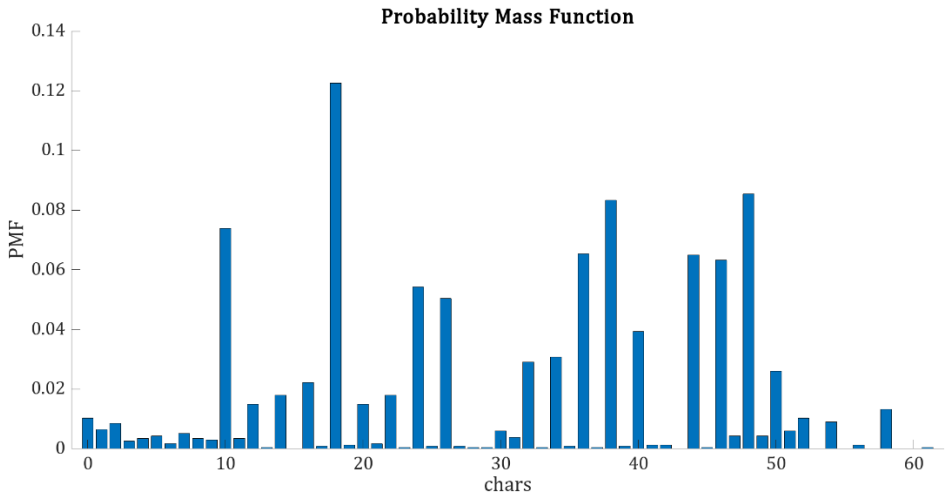


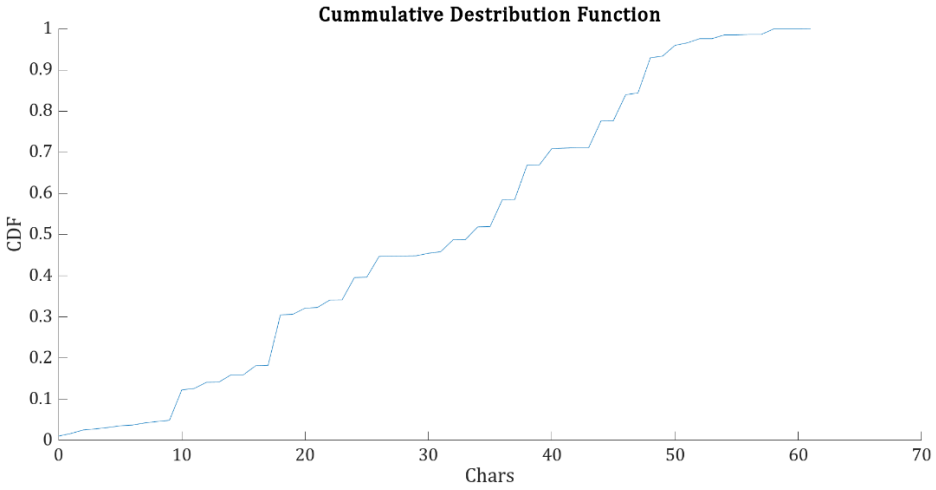
11. You can click on clear button to set everything to initial states for better usage.



❻ | Results of the Sample Test.







❼ | References.

[1] Walpole, Ronald E._ Myers, Raymond H._ Myers, Sharon L._ Ye, Keying - Probability & statistics for engineers & scientists-Pearson (2017)

[2] https://www.randomservices.org/random/expect/Skew.html

*****