Estadística aplicada para ingenieros en MATLAB

Se presenta un curso que proporciona una base de Estadística Aplicada para Ingenieros. En el curso se desarrollan los fundamentos teóricos necesarios, así como su desarrollo con el programa informático MATLAB, para tratar temas tan importantes en la ingeniería como la estadística descriptiva, las funciones de probabilidad, las simulaciones o el ajuste de datos por regresión.

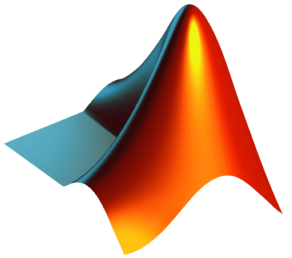
En lo referente a su contenido, en cada tema se describen las funciones que utiliza MATLAB y se resuelven distintos ejemplos. Además, al finalizar el tema se proponen algunos problemas de ingeniería, cuya solución se aporta al final de la parte teórica.

MATLAB Course

Statistical Toolbox

Tutorials and Exercises

Dr Lourdes Jalon Ramirez



Outline

1. File Inputs/Outputs
2. Descriptive Statistics
3. Probability Distributions
4. Sampling
5. Regression Analysis

1. File Inputs/Outputs
1.1 Inputs
1.2 Outputs

2. Descriptive Statistics

3. Probability Distributions

4. Sampling

5. Regression Analysis

1.1 Inputs

- Importdata: to import data from files.

» a=importdata ('data.txt')

- a is a structure with fields: *data, textdata*, and *colheaders*:

```
a =
    struct with fields:
        data: [6×2 double]
    textdata: {'cycles'  'stress'}
  colheaders: {'cycles'  'stress'}
```

- To read the fields:

» x=a.data;

» names=a.colheaders;

data.txt

cycles	stress
200	2000000
420	856126
468	550000
250	186235
549	1256398
246	359412

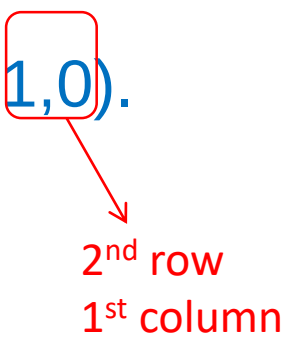
1.1 Inputs

- dlmread: Read ASCII-delimited file of numeric data into matrix.

» a=dlmread('data.txt','',R1,C1)

- start reading at row offset R1 and column offset C1. For example, the offsets R1=0, C1=0 specify the first value in the file.

» x=dlmread('data.txt','',1,0).



1.1 Inputs

- load: Load variables from file into workspace.

```
» a=load('WANA_T_1052046');
```

- xlsread: Read Microsoft Excel spreadsheet file.

```
» [num,txt,raw]=xlsread('data_xlc.xlsx','Sheet2');
```

- *num* contains numbers, *txt* contains strings, *raw* is the entire cell array containing everything. ‘Sheet2’ is the name of the sheet.

1.2 Outputs

```
» A=rand(10,4);
```

- save: Save workspace variables to file.

```
» save myData.mat A;
```

- Save all variables from the workspace in a binary MAT-file

```
» save('pqfile.txt','A','-ascii');
```

- Save all variables from the workspace an ASCII file.

1.2 Outputs

- dlmwrite: Write matrix to ASCII-delimited file.

```
» dlmwrite('myFile.txt',A,'delimiter','\t','precision',3);
```

- delimited by the tab character and using a precision of 3 significant digits

- xlswrite: Write Microsoft Excel spreadsheet file.

```
» xlswrite('randomNumbers',A ,'Sheet1');
```

- we specify the filename, the data and the sheet name

1.2 Outputs

Other import/export functions, with differing degrees of flexibility and ease of use, include *csvread, csvwrite, fgets, fprintf* (which has an optional argument to specify a file), *fscanf, textread*.

Outline

1. File Inputs/Outputs

2. Descriptive Statistics

    2.1 Graphical Methods
    2.2 Numerical Methods
    2.3 Exercises

3. Probability Distributions

4. Sampling

5. Regression Analysis

2.1 Graphical Methods

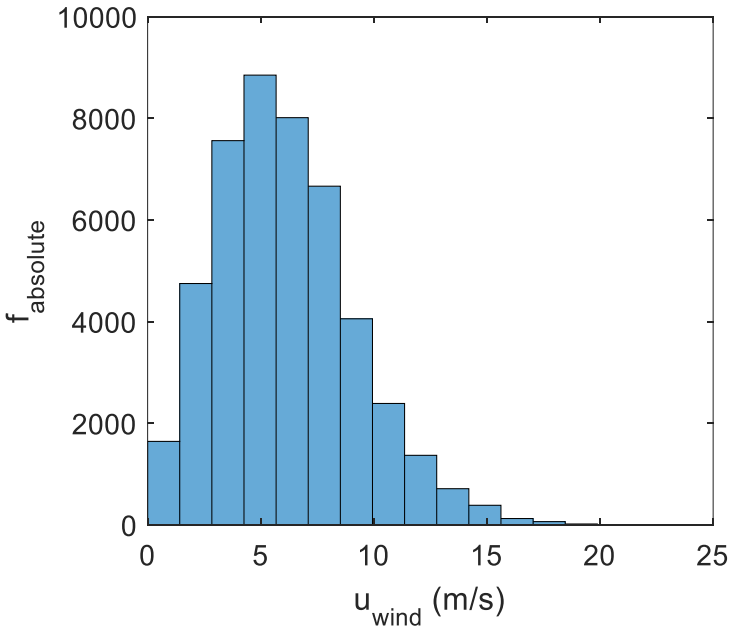
➢ Histogram:

A histogram is a way to graphically represent the absolute frequency of a data set. The absolute frequency of an event *i* represents the number *ni* of times that a value appears within a given interval:

$$f_{abs,i} = n_i$$
$$\sum_{i=1}^{t} ni = Nt$$

where Nt is the total size of the data.

Bin	Absolute frequency
0-1.42 m/s (bin width=1.42)	1647
1.42-2.84 m/s(bin width=1.42)	4753
2.84-4.26 m/s (bin width=1.42)	7564



2.1 Graphical Methods

➢ Histogram:

The basic MATLAB package has a function for calculating and plotting a histogram:

» histogram (X);

- computes the number of elements in the bin (absolute frequency) with a uniform width.

» histogram (X, nbins);

- uses a number of bins specified by the scalar *nbins*

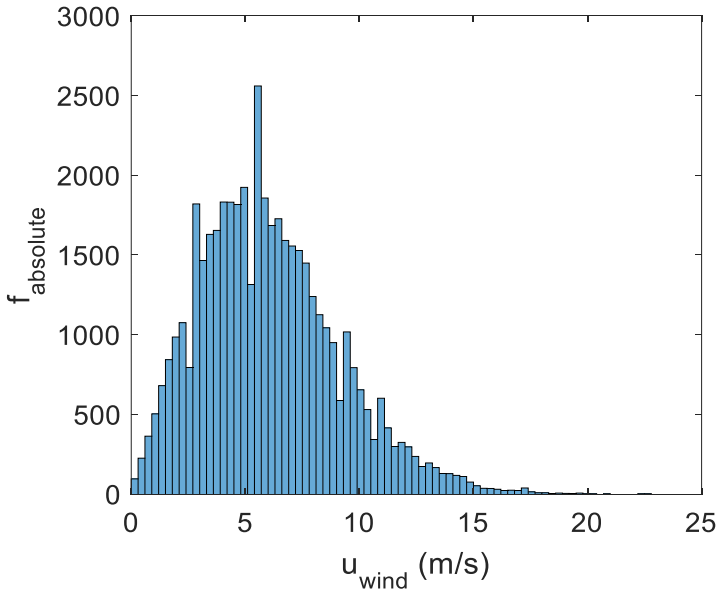
When you specify an output argument to the histogram function (hist=histogram(X)), it returns a histogram object. You can use this object to inspect the properties of the histogram, such as the number of bins or the width of the bins.

2.1 Graphical Methods

➢ Histogram:

Example 2.1. Read the data in wana.txt. These data include sea state parameters recorded every three hours from 1995 to 2011. Represent the histogram of the wind velocity, and find the number of bins, and the absolute frequencies.

```
» data=dlmread('wana.txt','',1,0);
» w=data(:,17);
» hist=histogram(w);
» xlabel ('u_{wind} (m/s)')
» ylabel ('f_{absolute}')
```



2.1 Graphical Methods

- Histogram:

```
» hist
                Histogram with properties:
                      Data: [46661×1 double]
                    Values: [1×76 double]
                   NumBins: 76
                  BinEdges: [1×77 double]
                  BinWidth: 0.3000
                 BinLimits: [0 22.8000]
             Normalization: 'count'
                 FaceColor: 'auto'
                 EdgeColor: [0 0 0]
» hist.NumBins
                ans = 76
» Fabs= hist.Values;
```

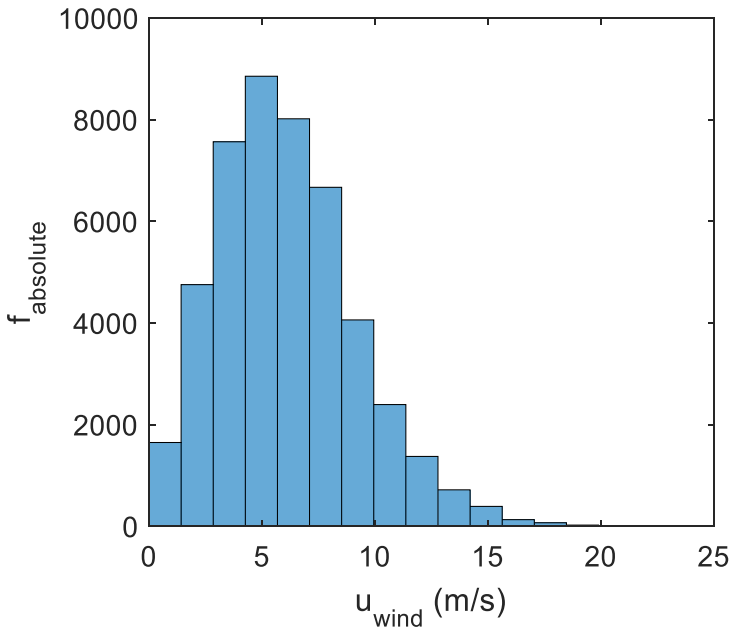
2.1 Graphical Methods

➢ Histogram:

Example 2.2. Represent the histogram of the wind velocity, considering a number of bins N=16. After that, calculate the relative frequency of the wind velocity, and its sum.

```
» Nbins=16;
» figure()
» hist2=histogram(w,Nbins);
» xlabel ('u_{wind} (m/s)')
» ylabel ('f_{absolute}')
» frel=hist2.Values./Nt;
» sum(frel)
```

ans = 1



2.2 Numerical Methods

- Measures of central tendency: mean, median, mode.

Function	Syntax	Description
Mean: average or mean value	M=mean(A)	returns the mean of the elements of A
	M=mean(A,dim)	returns the mean of the elements of A along dimension dim. Dim=1 (column), Dim=2(row)
Median: median value	M=mean(A)	returns the median of the elements of A
	M=mean(A,dim)	returns the median of the elements of A along dimension dim
Mode: most frequent values	M = mode(A)	returns the mode of the elements of A
	M = mode(A,dim)	returns the mode of the elements of A elements along dimension dim
	[M,F] = mode(___)	also returns a frequency array F

2.2 Numerical Methods

➢ Measures of central tendency: mean, median, mode.

Example 2.3. Calculate the mean, median, mode as well as the number of times it occurs, of the wind velocity.

» M=mean(w)

M = 6.0865

» M=median(w)

M = 5.7000

» [M,F]=mode(w)

M= 5.5000

F= 688

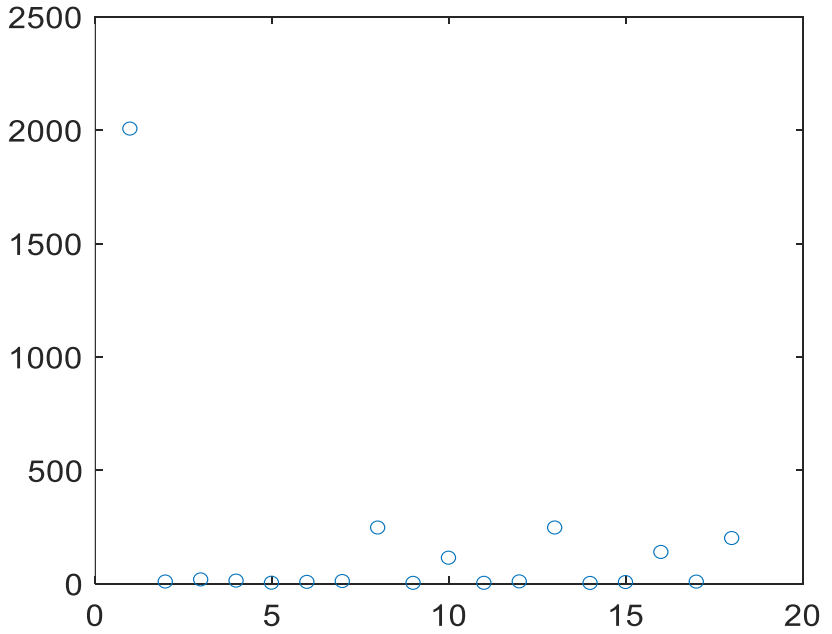
2.2 Numerical Methods

➢ Measures of central tendency: mean, median, mode.

Example 2.4. Calculate the mean of the data included in wana.txt along the columns (DIM=1), and represent the different obtained values.

```
» data=dlmread('wana.txt','',1,0);
» M=mean(data,1);
» figure()
» plot(M,'o')
```

column



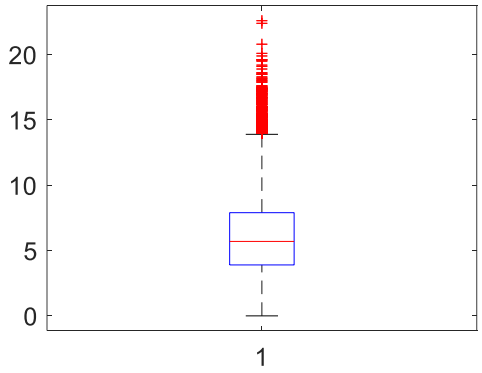
2.2 Numerical Methods

➢ Quantile, percentile.

Function	Syntax	Description
quantile: quantiles of a data set	Y = quantile(X,p)	returns quantiles of the values in data vector or matrix X for the cumulative probability or probabilities p in the interval [0,1]
	Y = quantile(X,p,dim)	returns quantiles along dimension dim
prctile: percentiles of a data set	Y = prctile(X,p)	returns percentiles of the values in a data vector or matrix X for the percentages p in the interval [0,100]
	Y = prctile(X,p,dim)	returns percentiles along dimension dim.

A box plot helps to visualize these statistics:

- The tops and bottoms of each "box" are the 25th and 75th percentiles of the samples, respectively.
- The line in the middle of each box is the sample median.



» boxplot(x) : creates a box plot of the data in x

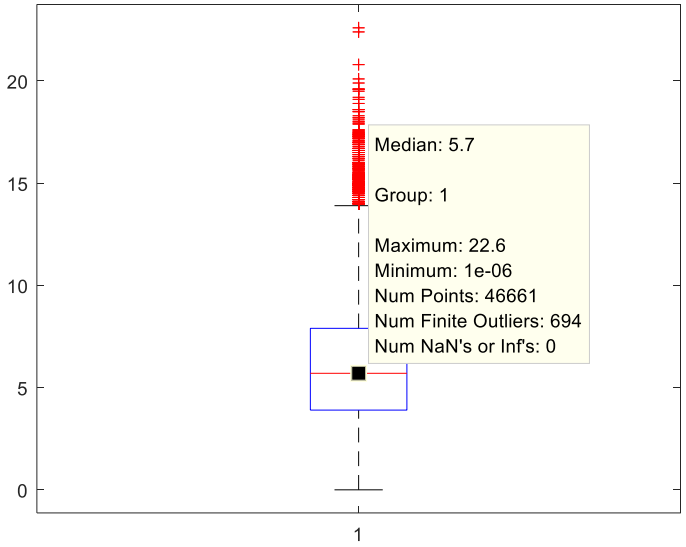
» boxplot(x,g) : creates a box plot using one or more grouping variables contained in g

2.2 Numerical Methods

➢ Quantile, percentile.

Example 2.5. Calculate the quantile for the cumulative probability 0.025, 0.25, 0.5, 0.75, 0.9, and the 2.5^{th}, 25^{th}, 50^{th}, 75^{th},90^{th} percentiles of the wind velocity. Represent the boxplot of wind velocity, and find the median.

```
» pQ=[0.025 0.25 0.50 0.75 0.90];
» Q=quantile(w,pQ)
Q =1.2000   3.9000   5.7000   7.9000   10.1400
» pY=[2.5,25,50,75,90];
» Y=prctile(w,pY)
Y =1.2000   3.9000   5.7000   7.9000   10.1400
» boxplot(w)
```



2.2 Numerical Methods

- Measures of the variation or dispersion of a set of data values: variance, standard deviation.

Function	Syntax	Description
Var: variance	V = var(A)	returns the variance of the elements of A
std: standard deviation. It is the square of the variance.	S = std(A)	returns the standard deviation of the elements of A

Example 2.6. Calculate the variance and standard deviation.

```
» V=var(w)
V=9.2242

» D=std(w)
D= 3.0371
```

2.3 Exercises

1. Read the data in wana.txt. These data include sea state parameters recorded every three hours from 1995 to 2011. Represent the histogram of the significant wave height (column 5), considering a number of bins specified by N=(1+3.3*log(Nt)), where Nt is the total size of the data. Save the absolute frequency in binary MAT.file, in ASCII file, and in Excel spreadsheed file.
Verify that the sum of the absolute frequencies is equal to the total size of the data.

2.3 Exercises

2. Calculate the mean, median, mode, the quantile for the cumulative probability 0.5, and the 50th percentile of the significant wave height.

3. Calculate the standard deviation with the variance, and verify that it is the same calculated with Matlab function *std*.

4. Represent the annual variability of the significant wave height with the boxplot. The data of years correspond to the first column of the wana.txt.

Outline

1. File Inputs/Outputs

2. Descriptive Statistics

3. Probability Distributions
    - 3.1 Introduction
    - 3.2 Empirical Estimation
    - 3.3 Common Theoretical Distributions
        - 3.3.1 Graphic User Interfaces
        - 3.3.2 Command Line Functions
    - 3.4 Exercises

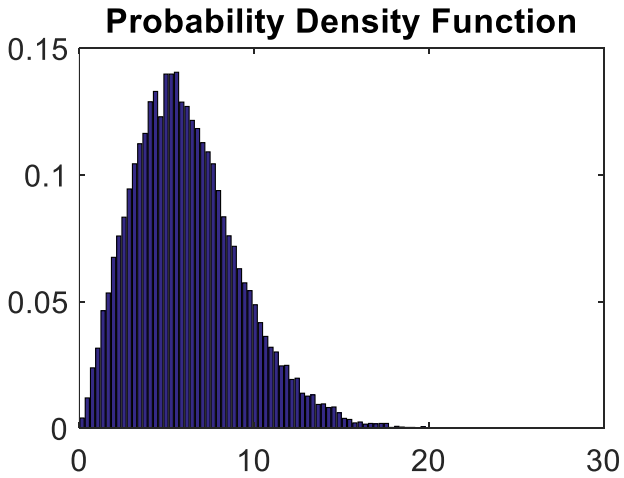
4. Sampling

5. Regression Analysis

3.1 Introduction

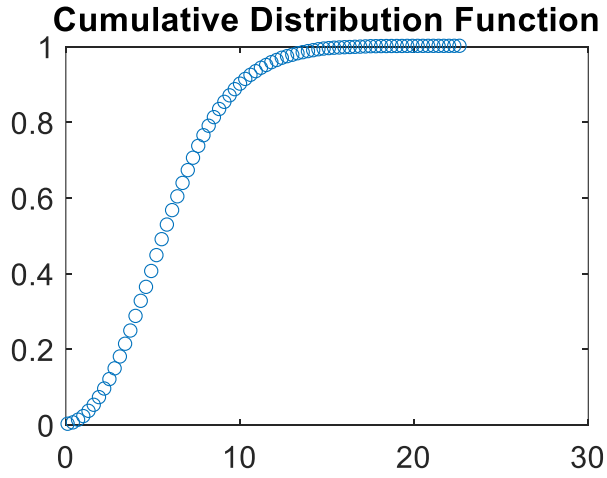
The probability density function (*f (x)*) represents the ratio of the probability that a value *x* is within an infinitesimal interval and the length of the infinitesimal interval

$$f_X(x) = \Pr\left(x \in (x_i, x_i + \Delta x)\right) / \Delta x$$



The cumulative distribution function (cdf) is defined as:

$$F_X(x) = \Pr(X \le x) = \int_{-\infty}^{x} f_X(x)dx$$



3.2 Empirical Estimation

- The empirical probability density function could be estimated from the histogram as:

$$f_X(x) = \Pr\left(x \in (x_i, x_i + \Delta x)\right) / \Delta x$$

$$f_X(x) = \frac{n_i}{Nt \cdot \Delta x}$$

The code will be the following:

```
» N=length(x)
» hist = histogram(x)
» n=hist.Values;
» dx=hist.BinWidth,
» frel=n/N;
» pdf = frel./(dx);
» b= hist.BinEdges;
» xc= (b(2:end)+b(1:end-1))./2;
» bar(xc,pdf)
```

3.2 Empirical Estimation

- The empirical cumulative distribution function could be estimated as the cumulative sum of the relative frequency:

» cdf=cumsum(frel)

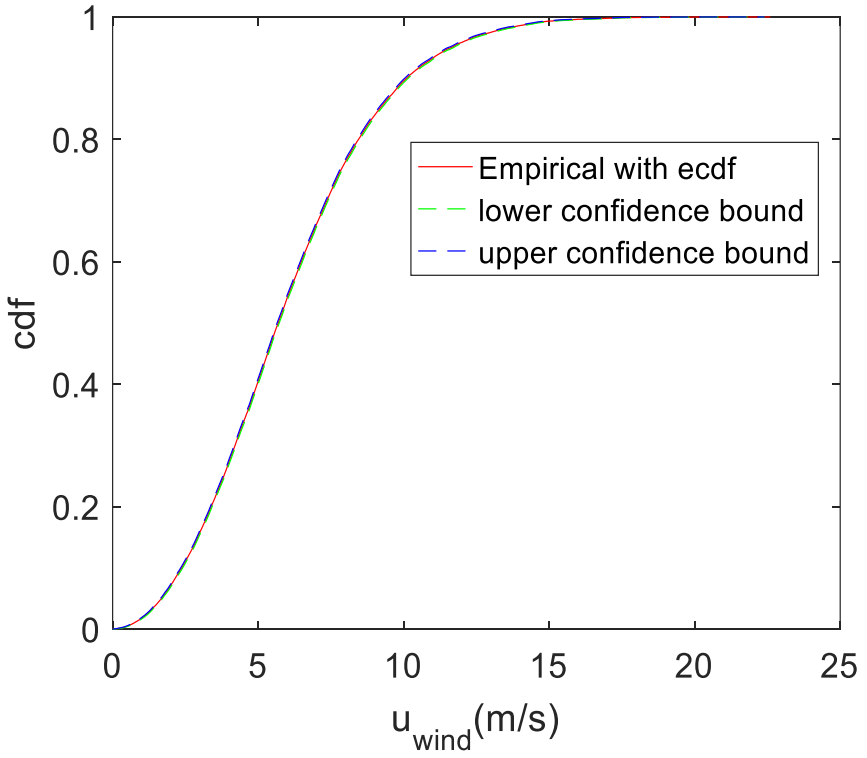
- Also it is possible to use a Matlab function:

» [f,x] = ecdf(y): returns the empirical cumulative distribution function f, evaluated at the points in x, using the data in the vector y.
» [f,x,flo,fup] = ecdf(___): also returns the 95% lower and upper confidence bounds for the evaluated function values.

3.2 Empirical Estimation

Example 3.1 Represent the cumulative distribution function and consider the 95% lower and upper confidence bounds of the wind velocity.

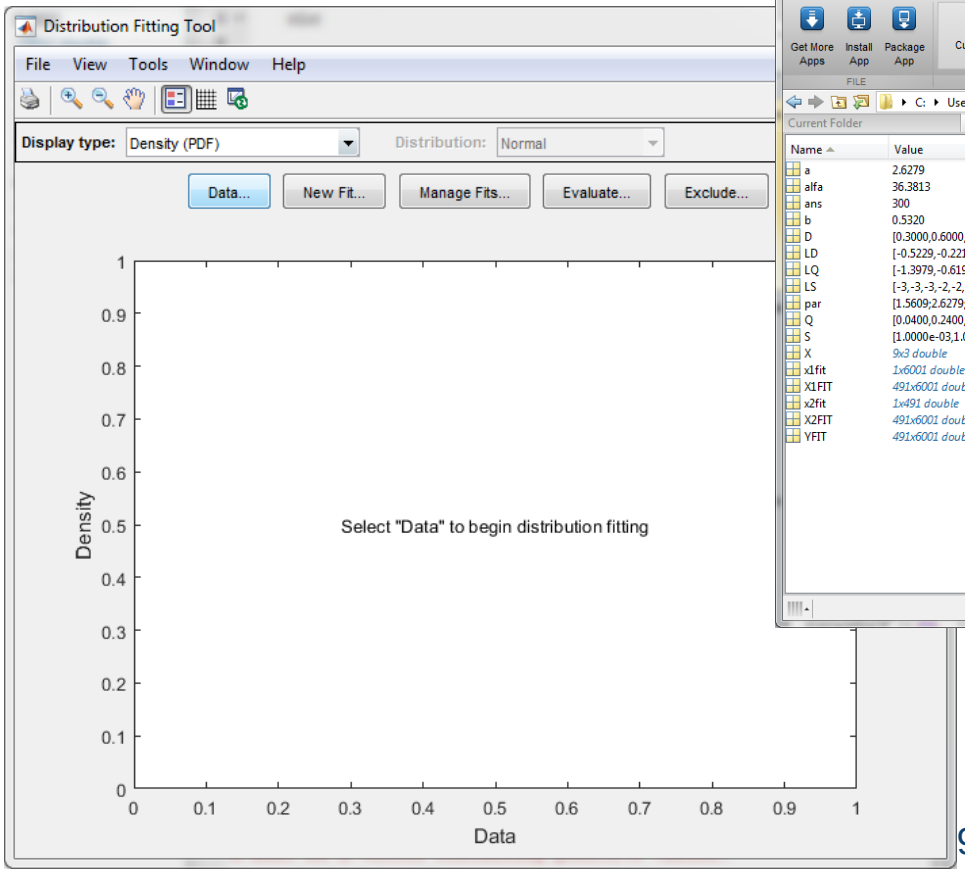
```
» [f,x,flo,fup] = ecdf(w);
» figure()
» plot(x,f,'r'),hold on,
» plot(x,flo,'g--'),hold on
» plot(x,fup,'b--'),
» legend('Empirical with ecdf',...
    'lower confidence bound', …
'upper confidence bound')
» xlabel('u_{wind}(m/s)')
» ylabel('cdf')
```

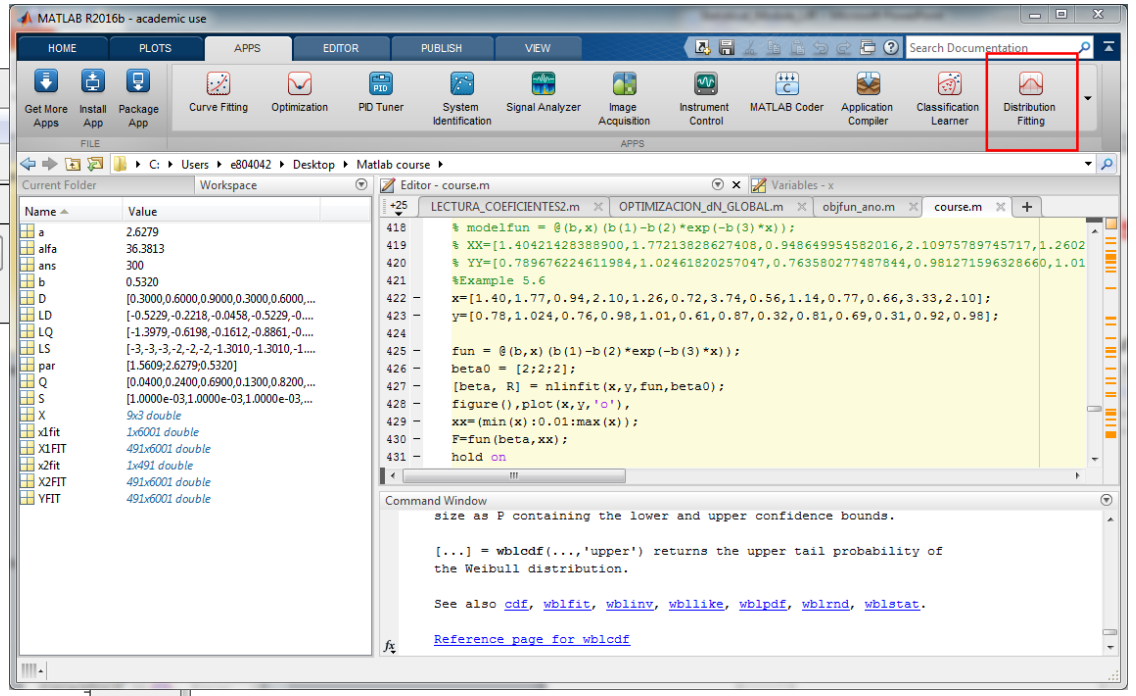


3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

- To open the GUIs, click in *APPS/Distribution Fit*

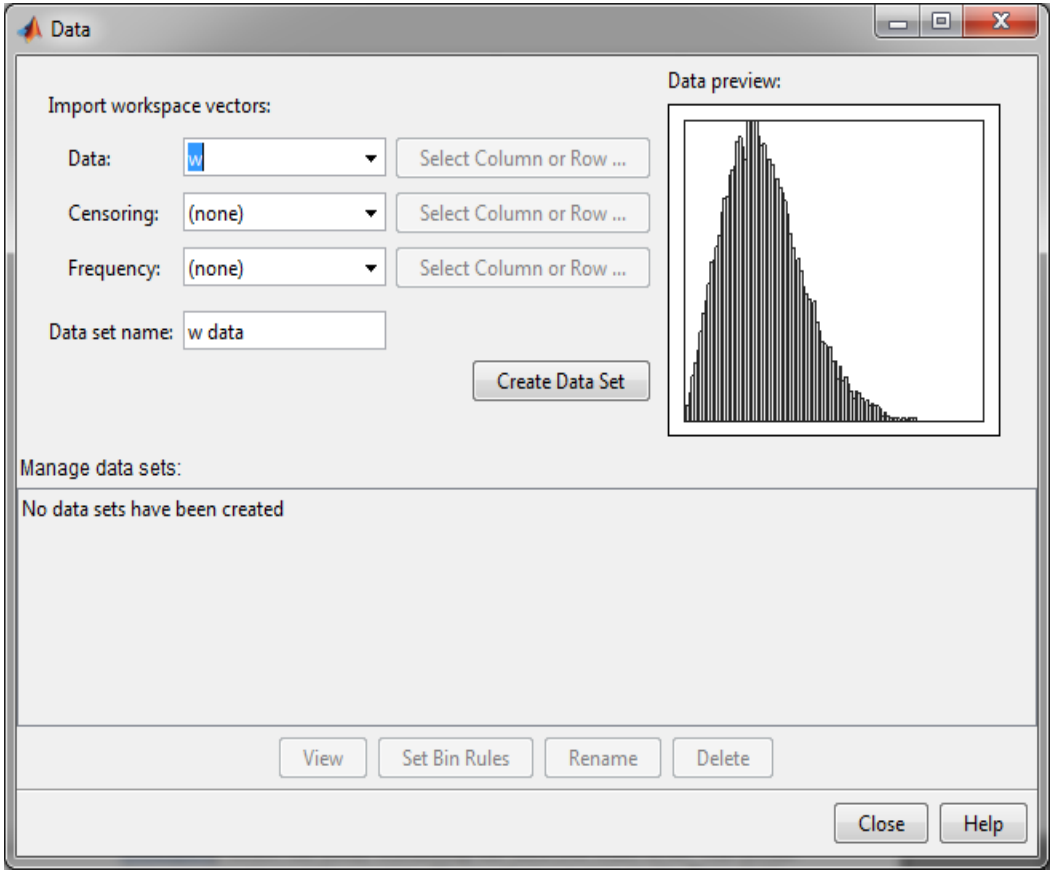




3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

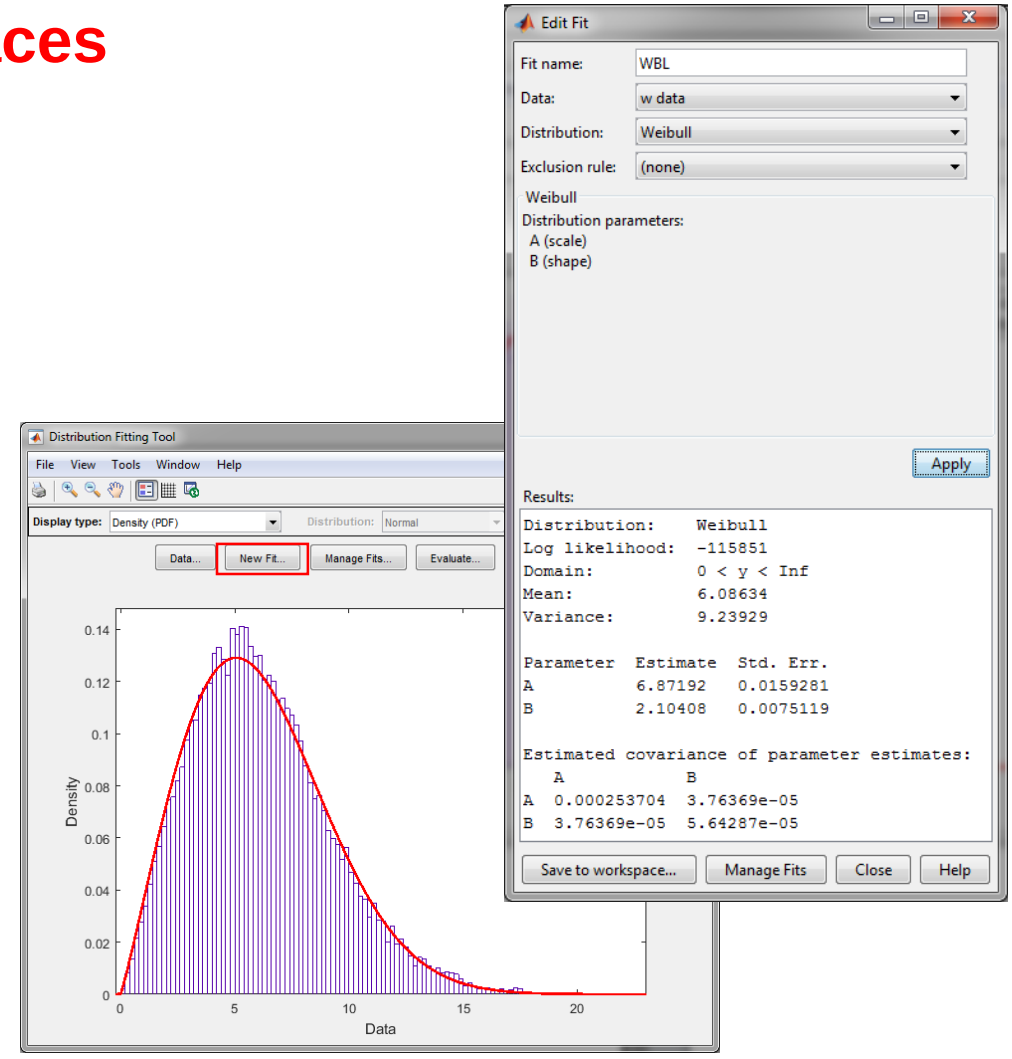
- Import data from workspace.
- Click the Data button in the main window of the Distribution Fitting Tool.
- Select the array containing the data you want to fit from drop-down list in the Data field.
- Click Create Data Set. It is possible to change the histogram bins by click the Set Bin Rules button. Furthermore, it is possible to display bounds selecting Conf Bounds next to de Data Set.



3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

- Creating a New Fit.
- Click the New Fit button
- Select the distribution you want to fit from drop-down list in the Distribution button.
- Click Apply and the Results panel displays the values of the estimated parameters.
- The main window of Distribution Fitting Tool displays the empirical distribution with the theoretical distribution

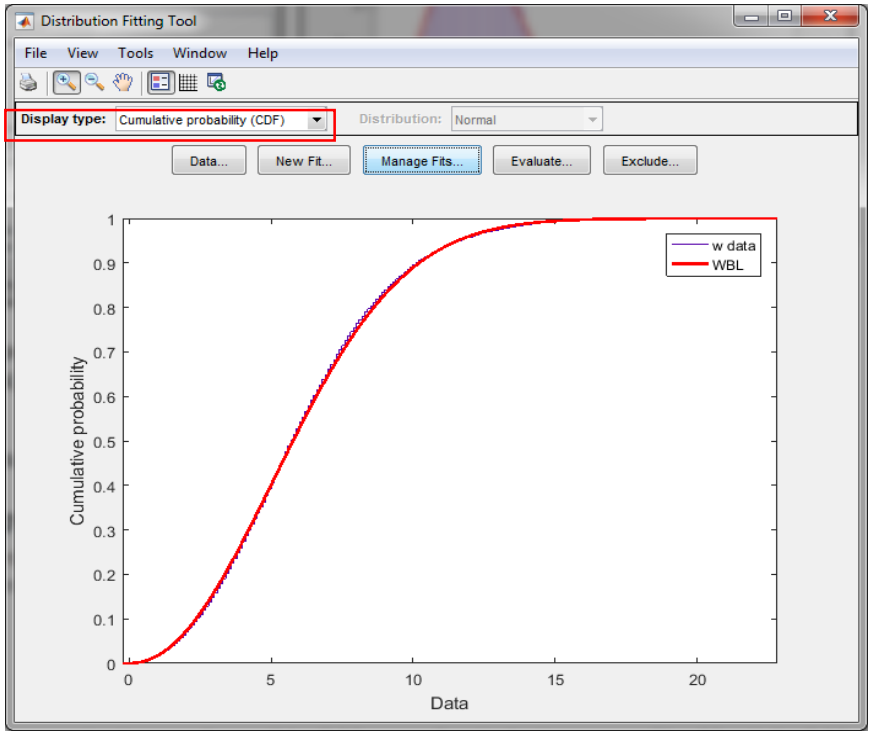


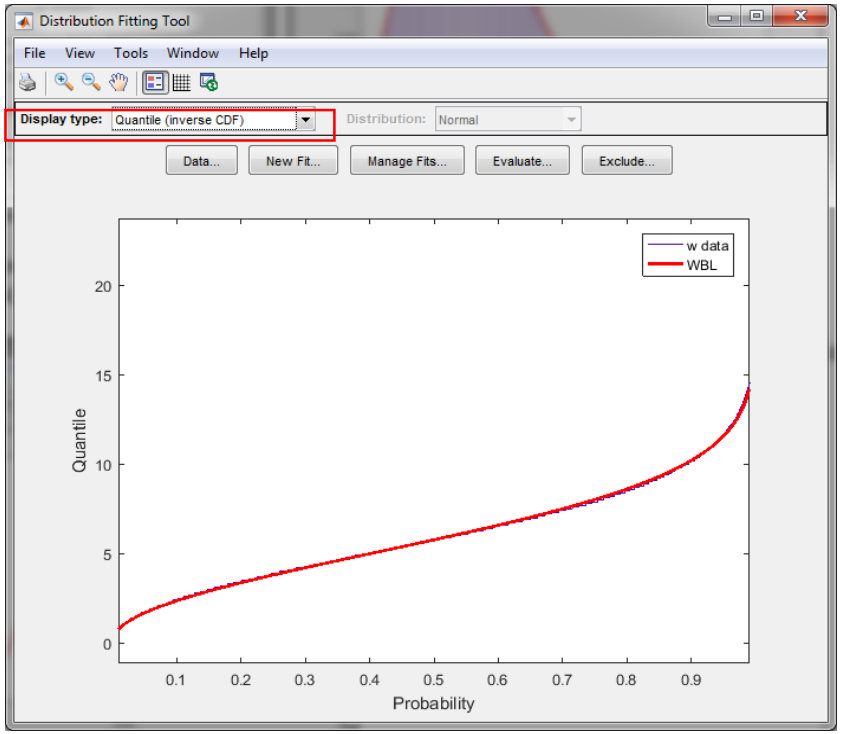
3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

- Display type.

Specify the type of plot displayed: pdf, cdf, quantile, probability plot (exponential, extreme value, logistic, log-logistic, lognormal, normal, Rayleigh, weibull).



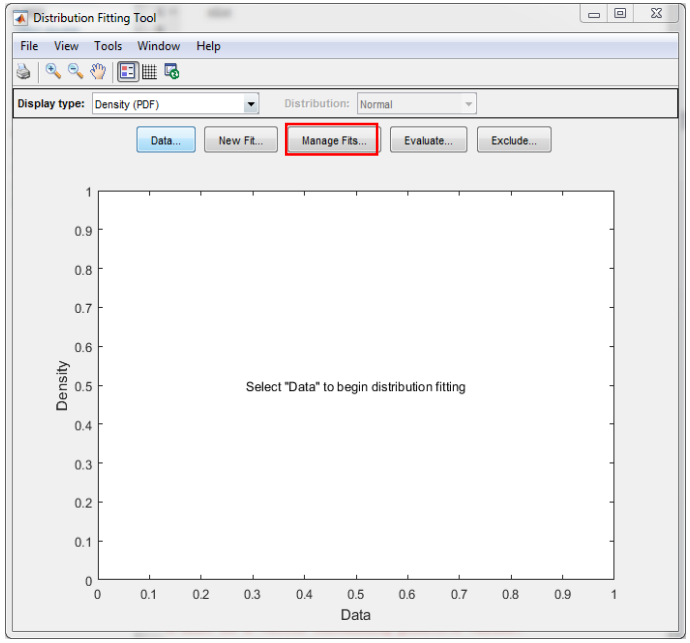


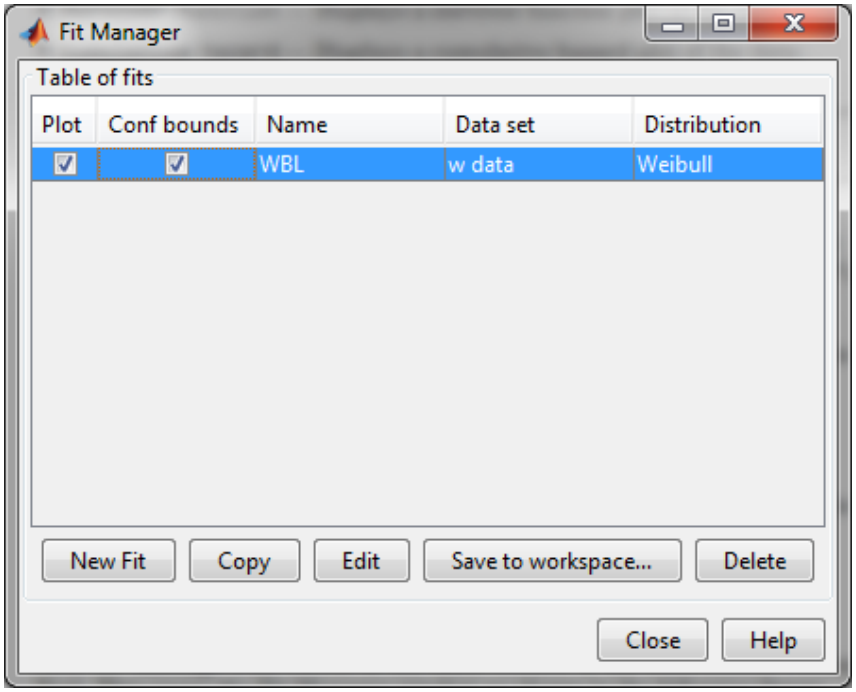
3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

- Managing Fits.

It is possible to opens a new fit window (New fit button), create a copy of the selected fit (Copy button), edit the fit (Edit button), or delete the selected fit (Delete button). Furthermore, it is possible to display bounds for the fit selecting Conf Bounds.

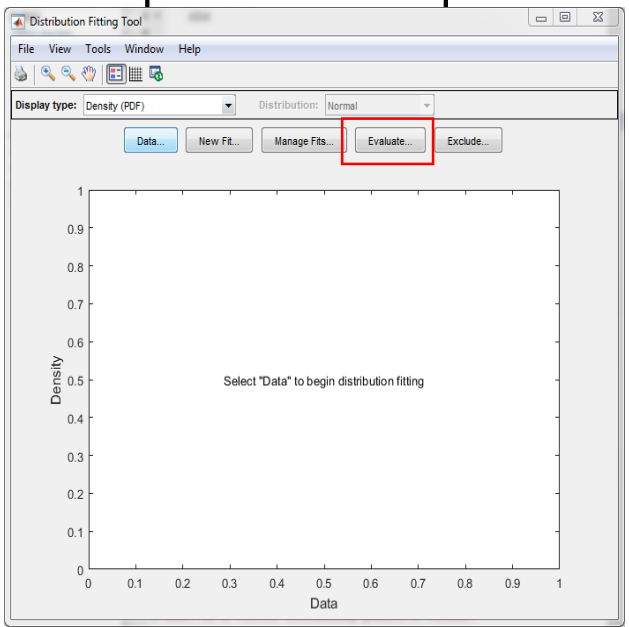


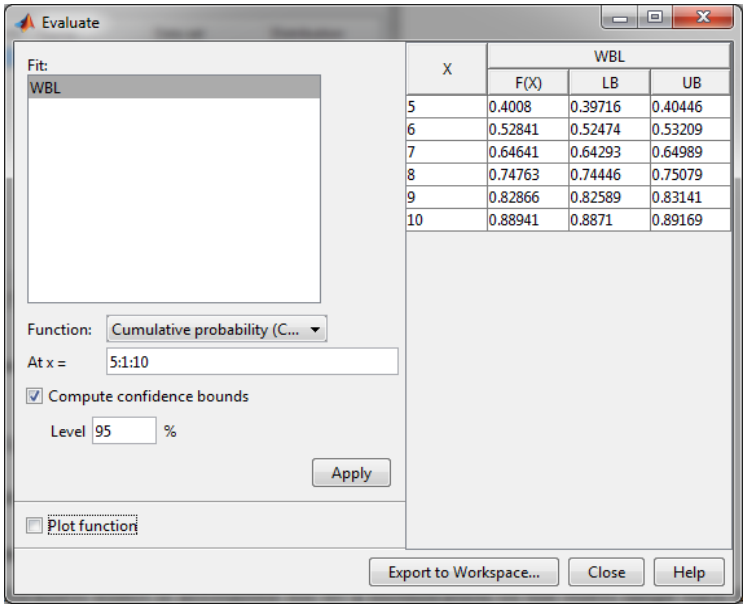


3.3 Common Theoretical Distributions

3.3.1 Graphic User Interfaces

- Evaluating Fits.
- select the fit that you want to evaluate
- select the type of probability function you want to evaluate (pdf, cdf, quantile…)
- enter the vector of points at which you want to evaluate the function
- It is possible to compute the confidence bounds
- Click Apply and the results of the evaluated function appeared on the right
- Click Export to Workspace allow saving results in the Workspace.





3.3 Common Theoretical Distributions

3.3.2 Command Line Functions

The process to represent the theoretical distribution fitted to the data is to estimate the parameter of the function, using the command function *name_of_distribution_fit*. After that, it is possible to represent it with the functions *name_of_distribution_pdf* or *name_of_distribution_cdf*.

Name	fit	pdf	cdf
Normal	normfit	normpdf	normcdf
LogNormal	lognfit	lognpdf	logncdf
Weibull	wblfit	wblpdf	wblcdf
Generalized Extreme value	gevfit	gevpdf	gevcdf

3.3 Common Theoretical Distributions

3.3.2 Command Line Functions

Example 3.2. The wind speed follows a Weibull distribution. Find the fitting parameters and calculate the probability that the wind speed is 22.6 m/s.

```
» parameters=wblfit(w);
» a=parameters(1);b=parameters(2);
» x=22.6;
» p = wblcdf(x,a,b)
```

P=1

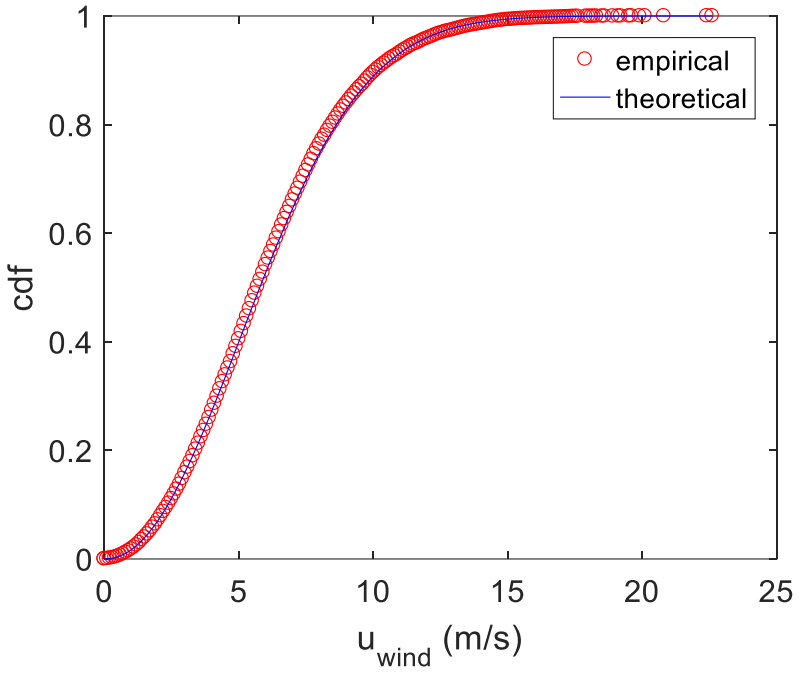
3.3 Common Theoretical Distributions

3.3.2 Command Line Functions

Example 3.3. Represent the empirical cumulative distribution function with the theoretical cumulative distribution function of the wind speed supposed as a Weibull distribution.

```
» [fe,x] = ecdf(w);
» w_sort=sort(w);
» ft=wblcdf(w_sort,a,b);

» figure()
» plot(x,fe,'ro'),hold on
» plot(w_sort,ft,'b-')
» legend('empirical','theoretical')
» xlabel('u_{wind} (m/s)')
» ylabel('cdf')
```



3.4 Exercises

1. Estimate the probability density function and the cumulative distribution function of the significant wave height, considering the relative frequency values. Compare the cdf results with the Matlab function and consider the 95% lower and upper confidence bounds.

2. Find the best distribution function to the significant wave height using the Graphic User Interface. The best fit will have the lowest value of the Log likelihood. After that, evaluate the cumulative distribution function of the median of the data, and the quantile for the cumulative probability 0.5

3. Represent the cumulative distribution function obtained in the Exercise 2, with command line functions.

Outline

1. File Inputs/Outputs

2. Descriptive Statistics

3. Probability Distributions

4. Sampling
- **4.1 Direct Methods**
- **4.2 Inversion Methods**
- **4.3 Exercises**

5. Regression Analysis

4.1 Direct Methods

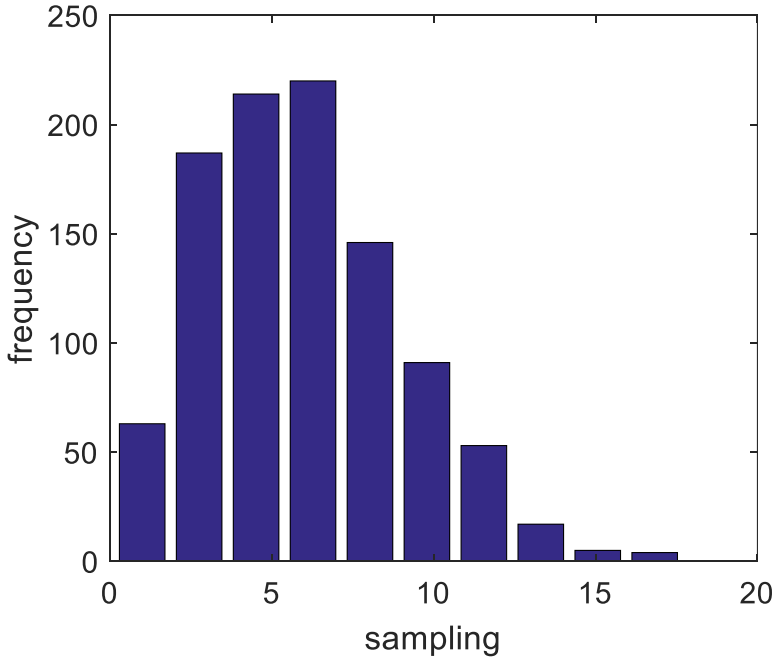
- Direct methods directly use the definition of the distribution, and generate random numbers that are uniformly distributed.

Name	function
Normal	normrnd
LogNormal	lognrnd
Weibull	wblrnd
Generalized Extreme value	gevrnd

4.1 Direct Methods

Example 4.1. Represent the histogram of 1000 uniform random variables obtained from a Weibull distribution with scale parameter a= 6.8717, and shape parameter b=2.1038.

```
» a=6.8717;b=2.1038;
» random=[1 1000]; %1 random sample of size 1000
» R = wblrnd(a,b,random);
»[n,x]=hist(R);
» figure()
» bar(x,n)
» ylabel('frequency')
» xlabel('sampling')
```



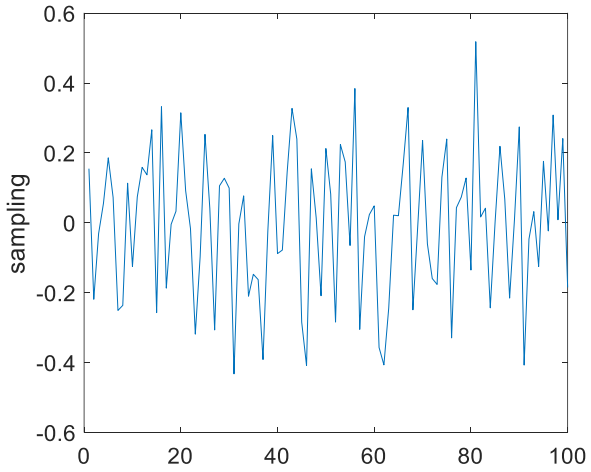
4.1 Direct Methods

Example 4.2. Generate 10 random samples of size 100, following a normal distribution with mean=0 and standard deviation=0.2. Represent the 3rd random sample.

```
» m=10;n=100;
» x=zeros(m,n);
» mu=0;sigma=0.2;
» for i=1:m %generate m random samples of size n
»     x(i,:)=normrnd(mu,sigma,1,n);
» end
```

$$X=\begin{pmatrix} x_{1,1} & x_{1,2} \dots \dots \dots \dots & x_{1,100} \\ & & \\ x_{10,1} & x_{10,2} \dots \dots \dots \dots & x_{10,100} \end{pmatrix}$$

```
» figure()
» plot(x(3,:))
» ylabel('sampling')
```



4.2 Inversion Methods

It uses the fact that the cumulative distribution function F(U) is uniform (0,1), then we can obtain the desired random variable X from the following relationship:

$$X = F^{-1}(U)$$

The inverse distribution function $F^{-1}(U)$ is obtained with the following functions:

Name	function
Normal	Norminv
LogNormal	Logninv
Weibull	wblinv
Generalized Extreme value	gevinv

The general procedure for the inverse transformation method is:

- Generate a uniform random number of probabilities U, with the Matlab function *rand*.
- Apply the desired inverse distribution function.

4.2 Inversion Methods

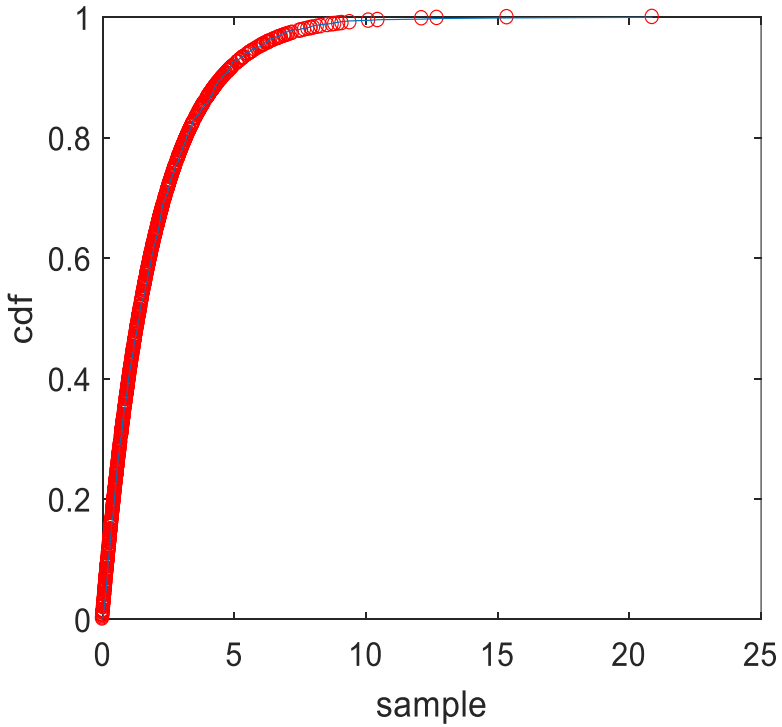
Example 4.3. Generate a random sample of size 1000 from an exponential distribution for $\lambda=2$.

```
» lam=2;
» n=1000;
» U=rand(1,n);
» X=expinv(U,lam);
```

4.2 Inversion Methods

Example 4.4. Plot the empirical cumulative distribution function of the above results and compare it with the exponential distribution function.

```
» X_sort=sort(X);
» cdf_rnd=ecdf(X_sort);
» lambda=expfit(X_sort);
» cdf=expcdf(X_sort,lambda);
» figure()
» plot(X_sort,cdf,'ro')
» hold on
» plot(X_sort,cdf_rnd(2:end));
» ylabel('cdf')
» xlabel('sample')
```



4.3 Exercises

1. Generate a random sample of size 2000 from the distribution function obtained in the Exercise 2 (Probability Distributions). Use direct and inverse methods, and calculate its median value.

2. Generate 100 random samples of size 1000, following a normal distribution with mean=1 and standard deviation=0.5.

Outline

1. File Inputs/Outputs

2. Descriptive Statistics

3. Probability Distributions

4. Sampling

5. Regression Analysis
- **5.1 Simple Linear Regression**
- **5.2 Multilinear Regression**
- **5.3 Non-Linear Regression**
- **5.4 Exercises**

5.1 Simple Linear Regression

- p = polyfit(x,y,n): returns the coefficients for a polynomial *p(x)* of degree *n* that is a best fit (in a least-squares sense) for the data in *y.* The coefficients in *p* are in descending powers, and the length of *p* is *n+1:*

$$p(x)=p_1x^n+p_2x^{n-1}+...+p_nx^n+p_{n+1}.$$

Example 5.1. Generate 10 points equally spaced along a cosine curve in the interval [0,2pi]. Use *polyfit* to fit a 7th-degree polynomial to the points.

```
» x = linspace(0, 2*pi,10);
» y = cos(x);
» p = polyfit(x,y,7)
p =
   0.0000    0.0010   -0.0187    0.1074   -
0.1181   -0.3974   -0.0327    1.0000
```

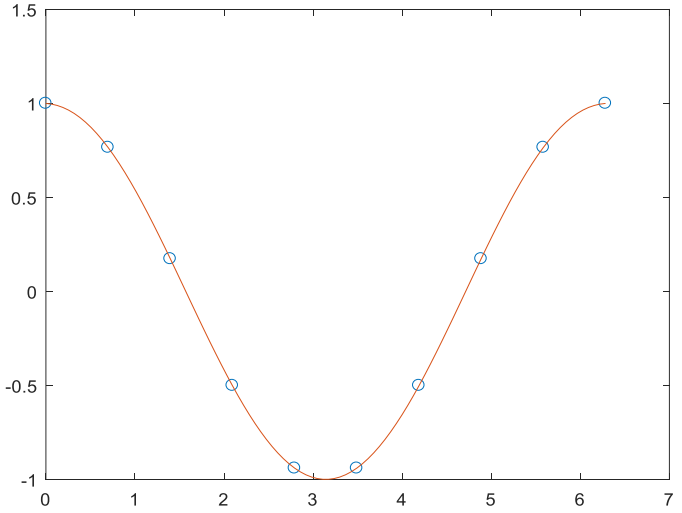
$$p(x)=0x^7+0.001x^6-0.0187x^5+0.1074x^4-0.1181x^3-0.3974x^2-0.0327x+1$$

5.1 Simple Linear Regression

- y = polyval(p,x): returns the value of a polynomial of degree *n* evaluated at *x*. The input argument *p* is a vector of length *n+1* whose elements are the coefficients in descending powers of the polynomial to be evaluated.

Example 5.2. Evaluate the previous polynomial and plot the results.

```
» x1 = linspace(0,2*pi);
» y1 = polyval(p,x1);
» figure()
» plot(x,y,'o')
» hold on
» plot(x1,y1)
```



5.2 Multilinear Regression

- [b,bint,r] = regress(y,X):

- returns a vector *b* of coefficient estimates for a multilinear regression of the responses in *y* on the predictors in *X*. Additionally returns a matrix *bint* of 95% confidence intervals for the coefficient estimates, and a vector *r* of residuals.

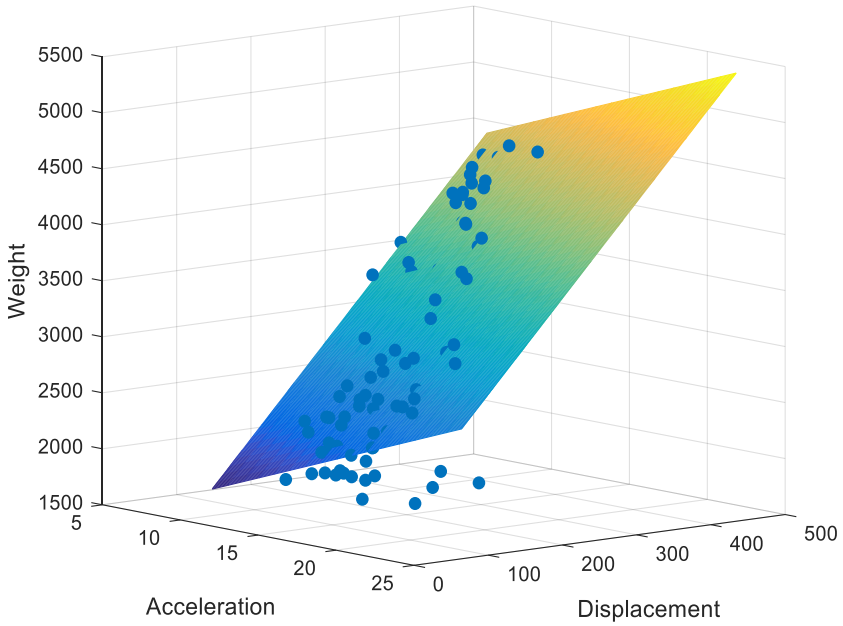
- assume that a matrix *X* will contain a first column of ones that corresponds to an "intercept" parameter *beta.*

- rcoplot(R, Rint): residuals plot. The residuals are the difference between the *observed* values of the response (dependent) variable and the values that a model *predicts*

5.2 Multilinear Regression

Example 5.3. Estimate the coefficients of a multiple linear regression, of acceleration and displacement as predictors, and weight as the response of the sample data: carsmall.

```
» load carsmall
» x1 = Acceleration;
» x2 = Displacement;
» y = Weight;
» X = [ones(size(x1)) x1 x2];
» b = regress(y,X)
» figure()
» scatter3(x1,x2,y,'filled');hold on
» x1fit = min(x1):1:max(x1);
» x2fit = min(x2):1:max(x2);
»[X1FIT,X2FIT] =meshgrid(x1fit,x2fit);
» YFIT = b(1) + b(2)*X1FIT + b(3)*X2FIT;
» mesh(X1FIT,X2FIT,YFIT)
» xlabel('Acceleration');ylabel('Displacement')
» zlabel('Weight')
```



5.3 Non-Linear Regression

- Transform the variables to make the relationship linear.

N	Function to Fit	Variable Substitution/Parameter Restoration
1	$y = a/x + b$	$x' = 1/x$
2	$y = b/(x + a)$	$y' = 1/y,\ a = b'/a',\ b = 1/a'$
3	$y = ab^x$	$y' = \ln y,\ a = e^{b'},\ b = e^{a'}$
4	$y = be^{ax}$	$y' = \ln y,\ b = e^{b'}$
5	$y = c - be^{-ax}$	$y' = \ln(c - y)$
6	$y = ax^b$	$y' = \ln y,\ x' = \ln x$
7	$y = axe^{bx}$	$y' = \ln(y/x)$
8	$y = c/(1 + be^{ax})$	$y' = \ln(c/y - 1),\ b = e^{b'}$
9	$y = a \ln x + b$	$x' = \ln x$

Example 5.4. It is known that the data tabulated below can be modelled by the following equation: y=a*x^b . Use the linear regression (polyfit) to determinate the parameters *a* and *b*. Data:x = [10 20 30 40 50 60 70 80];y = [25 70 380 550 610 1220 830 1450];

```
» Ly=log10(y);
» Lx=log10(x);
» p  = polyfit(Lx,Ly,1);
» b=p(1)= 1.9842
» a=10^p(2)= 0.2741
```

Transform:

$Ly = p_{n+1} + p_1 Lx$.

Ly=log(y);Lx=log(x)

$a = 10^{p_{n+1}}$

$b = p_1$

5.3 Non-Linear Regression

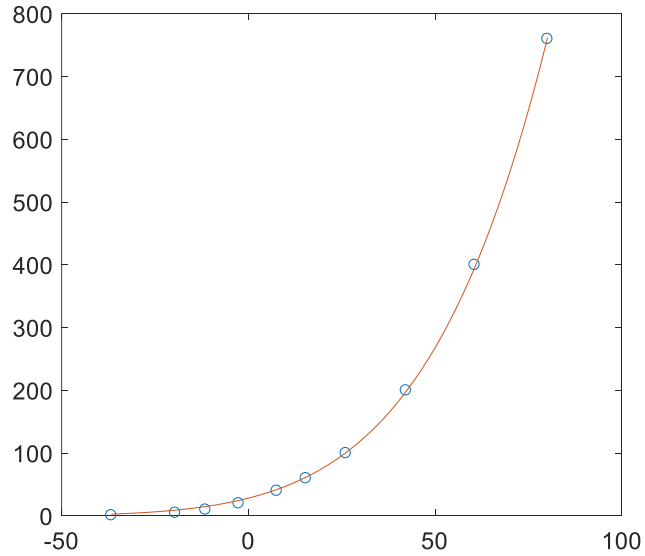
- Alternatively, try to fit a nonlinear function directly using either the Statistics and Machine Learning Toolbox™ *nlinfit* function, the Optimization Toolbox™ *lsqcurvefit* function, or by applying functions in the Curve Fitting Toolbox.

- x = lsqcurvefit(fun,x0,xdata,ydata): starts at *x0* and finds coefficients *x* to best fit the nonlinear function *fun(x,xdata)* to the data *ydata* (in the least-squares sense)

5.3 Non-Linear Regression

Example 5.5. Find the best fit to the data: x=[-36.7 -19.6 -11.5 -2.6 7.6 15.4 26.1 42.2 60.6 80.1]; y=[1 5 10 20 40 60 100 200 400 760]; with the equation log(Y)=A-(B/x+273.15).Represent the results.

```
» function F=myfun(par,x)
» F=10.^(par(1)-(par(2)./(x+273.15)));
» end
» y=[1 5 10 20 40 60 100 200 400 760];
» x=[-36.7 -19.6 -11.5 -2.6 7.6 15.4 26.1 42.2 60.6 80.1];
» x0=[5;500];
»[par, resnorm, residual]=lsqcurvefit(@myfun,x0,x,y)
» figure(),plot(x,y,'o'),
» xx=(min(x):0.01:max(x));
» F=myfun(par,xx);
» hold on
» plot(xx,F)
```



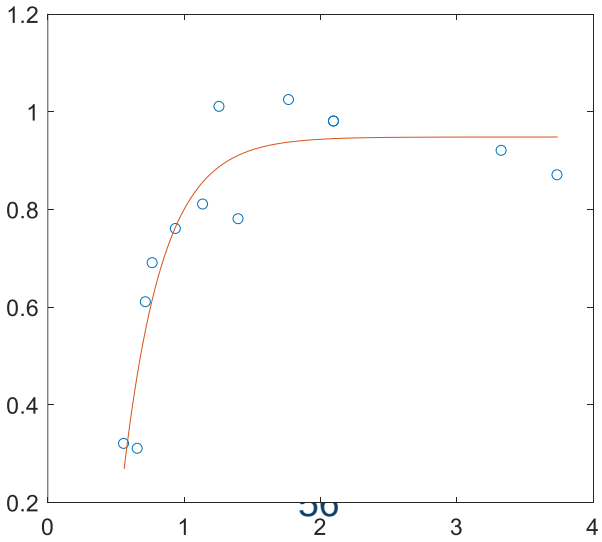
5.3 Non-Linear Regression

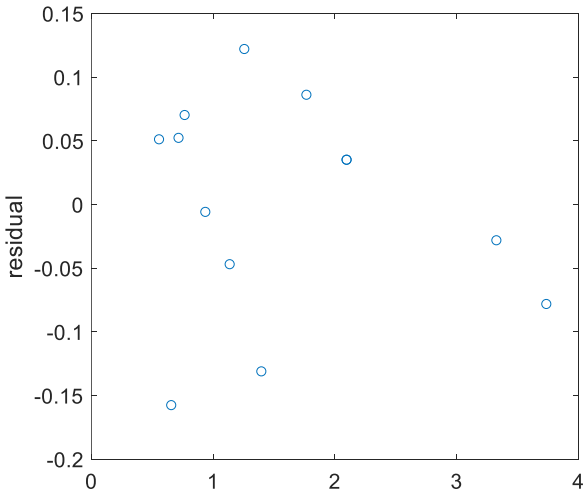
- [beta,R] = nlinfit(X,Y,modelfun,beta0): returns a vector *beta* of estimated coefficients for the nonlinear regression of the responses in *Y* on the predictors in *X* using the model specified by *modelfun*. The coefficients are estimated using iterative least squares estimation, with initial values specified by *beta0*. Additionally returns the residuals, *R*.

5.3 Non-Linear Regression

Example 5.6. Fit the nonlinear model y=a-b*exp(-c*x) with *nlinfit* and represent the fitting results and the residuals for the data:
x=[1.40,1.77,0.94,2.10,1.26,0.72,3.74,0.56,1.14,0.77,0.66,3.33,2.10];
Y=[0.78,1.024,0.76,0.98,1.01,0.61,0.87,0.32,0.81,0.69,0.31,0.92,0.98];

```
» fun = @(b,x)(b(1)-b(2)*exp(-b(3)*x));
» beta0 = [2;2;2];
»[beta, R] = nlinfit(x,y,fun,beta0)
» figure(),plot(x,y,'o'),
» xx=(min(x):0.01:max(x));
» F=fun(beta,xx);
» hold on
» plot(xx,F)
» figure(),plot(x,R,'o')
```





5.4 Exercises

1. Use *polyfit* to fit a 1th-degree polynomial to the vectors x = [10 20 30 40 50 60 70 80], y = [25 70 380 550 610 1220 830 1450]. Represent the fit, and the residuals. Based on your analysis predict y at x=50.

2. It is known that the data tabulated below can be modelled by the following equation: y=a*x^b . Use non-linear regression (*lsqcurvefit* or *nlinf* function) to determinate the parameters *a* and *b*.
   x = [10 20 30 40 50 60 70 80]; y = [25 70 380 550 610 1220 830 1450];

5.4 Exercises

3. A material is tested for cyclic fatigue failure whereby a stress, in MPa, is applied to the material and the number of cycles needed to cause failure is measured. The results are below. Use nonlinear regression to fit a model (log N=A+B log S) to this data. N=[1 10 100 1000 10000 100000 1000000]; S=[1100 1000 925 800 625 550 420].
4. The following data were collected for the steady flow of water in a concrete circular pipe:

Experiment	1	2	3	4	5	6	7	8	9
Diameter	0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
Slope	0.001	0.001	0.001	0.01	0.01	0.01	0.05	0.05	0.05
Flow	0.04	0.24	0.69	0.13	0.82	2.38	0.31	1.95	5.66

Use multiple linear regression to fit the following model to this data:

$$Q = \alpha D^a S^b$$

where Q=flow, D=diameter, and S=slope. Represent the observed and predicted data.

SOLUTIONS

DESCRIPTIVE STATISTICS

1. Read the data in wana.txt. These data include sea state parameters recorded every three hours from 1995 to 2011. Represent the histogram of the significant wave height (column 5), considering a number of bins specified by N=(1+3.3*log(Nt)), where Nt is the total size of the data. Save the absolute frequency in binary MAT.file, in ASCII file, and in Excel spreadsheed file.

Verify that the sum of the absolute frequencies is equal to the total size of the data.

```
data=dlmread('wana.txt','',1,0);
H=data(:,5);
Nt=length(H);
Nbins=floor(1+3.3*log10(Nt));
hist=histogram(H,Nbins);
values=hist.Values;
sum=sum(values)
Nt==sum

values=hist.Values;
save Fabsmat.mat values
save('Fabstxt.txt','values','-ascii')
xlswrite('Fabsxls',values,'Sheet1');
```

2. Calculate the mean, median, mode, the quantile for the cumulative probability 0.5, and the 50th percentile of the significant wave height.

```
mean(H)
mode(H)
median(H)
quantile(H,0.5)
prctile(H,50)
```

3. Calculate the standard deviation with the variance, and verify that it is the same calculated with Matlab function *std*.

```
S=sqrt(var(H))
S_matlab=std(H)
```

4. Represent the annual variability of the significant wave height with the boxplot. The data of years correspond to the first column of the wana.txt.

```
year=data(:,1);
boxplot(H,year)
```

PROBABILITY DISTRIBUTIONS

1. Estimate the probability density function and the cumulative distribution function of the significant wave height, considering the relative frequency values. Compare the cdf results with the Matlab function and consider the 95% lower and upper confidence bounds.

```
data=dlmread('wana.txt','',1,0);
H=data(:,5);
N=length(H);
histo = histogram(H);
n=histo.Values;
dx=histo.BinWidth;
frel=n/N;
pdf = frel./(dx);
b= histo.BinEdges;
xc=(b(2:end)+b(1:end-1))./2;

figure()
bar(xc,pdf);
xlabel('Hs (m)')
ylabel('pdf')
cdf=cumsum(frel);

figure()
plot(xc,cdf,'o-')
xlabel('Hs (m)')
ylabel('cdf')
[f,x,flo,fup] = ecdf(H);
hold on,
plot(x,f,'r'),
hold on
plot(x,flo,'g--'),hold on
plot(x,fup,'b--'),
legend('Empirical with frel','Empirical with ecdf',...
    'lower confidence bound', 'upper confidence bound')
```

2. Find the best distribution function to the significant wave height using the Graphic User Interface. The best fit will have the lowest value of the Log likelihood. After that, evaluate the cumulative distribution function of the median of the data, and the quantile for the cumulative probability 0.5

The best fit= Lognormal (Log likelihood=-45469.7)

The median of the data is H=1 m (calculated in Exercise 2 of DESCRIPTIVE STATISTICS). The cumulative distribution function at H=1 m is 0.48517

Quantile at p=0.5 is H=1.02356

3. Represent the cumulative distribution function obtained in the Exercise 2, with command line functions.

The best fit is the Lognormal. We find the parameters of the distribution model, and we calculate the cumulative distribution function

```
[fe,x] = ecdf(H);
h_sort=sort(H);
```

```
par=lognfit(h_sort);
ft=logncdf(h_sort,par(1),par(2));

figure()
plot(x,fe,'ro'),hold on
plot(h_sort,ft,'b-')
legend('empirical','theoretical')
xlabel('H (m)')
ylabel('cdf')
```

SAMPLING

1. Generate a random sample of size 2000 from the distribution function obtained in the Exercise 2 (Probability Distributions). Use direct and inverse methods, and calculate its median value.

The distribution function is the Lognormal, so we define one random sample of size 2000 and we generate the values from the Lognormal distribution using the parameters of the fitting calculated in the Exercise 3:

```
random=[1 2000];

%direct
R = lognrnd(par(1),par(2),random);
median(R)

%indirect
I1=rand(1,2000);
I2=logninv(I1,par(1),par(2));
median(I2)
```

2. Generate 100 random samples of size 1000, following a normal distribution with mean=1 and standard deviation=0.5.

We define an array of ceros with m rows and n column, where m=number of random samples; and n= size of the variables.

For each row or random sample, we are going to generate n values following a normal distribution.

```
m=100;n=1000;
x=zeros(m,n);
mu=1;sigma=0.5;
for i=1:m
    x(i,:)=normrnd(mu,sigma,1,n);
end
```

REGRESSION ANALYSIS

1. Use *polyfit* to fit a 1th-degree polynomial to the vectors x = [10 20 30 40 50 60 70 80], y = [25 70 380 550 610 1220 830 1450]. Represent the fit, and the residuals. Based on your analysis predict y at x=50.

```
x = [10 20 30 40 50 60 70 80];
y = [25 70 380 550 610 1220 830 1450];

p  = polyfit(x,y,1);


figure(),
plot(x,y,'o')
hold on
xx=(0:0.1:max(x));yy=p(2)+p(1).*xx;
plot(xx,yy);


yfit=polyval(p,x);
yres=y-yfit;
figure(),plot(x,yres)
legend('residuals')


ypredict = polyval(p,50);
```

2. It is known that the data tabulated below can be modelled by the following equation: y=a*x^b . Use non-linear regression (*lsqcurvefit* or *nlinf* function) to determinate the parameters *a* and *b*. x = [10 20 30 40 50 60 70 80]; y = [25 70 380 550 610 1220 830 1450];

We define the function:

```
function F=myfun2(par,x)
F=par(1)*x.^par(2);
end
```

We apply lsqcurvefit:

```
x = [10 20 30 40 50 60 70 80];
y = [25 70 380 550 610 1220 830 1450];
x0=[5;2];
[par, resnorm, residual]=lsqcurvefit(@myfun2,x0,x,y)
```

a=par(1)

b=par(2)

We use nlinf

```
clear all
fun=@(par,x) par(1)*x.^par(2);
x0=[2;2];
x = [10 20 30 40 50 60 70 80];
y = [25 70 380 550 610 1220 830 1450];
```

```
[beta, R] = nlinfit(x,y,fun,x0)
```
a=beta(1)

b=beta(2)

3. A material is tested for cyclic fatigue failure whereby a stress, in MPa, is applied to the material and the number of cycles needed to cause failure is measured. The results are below. Use nonlinear regression to fit a model (log N=A+B log S) to this data. N=[1 10 100 1000 10000 100000 1000000]; S=[1100 1000 925 800 625 550 420].

```
stres=[1100 1000 925 800 625 550 420];
cycle=[1 10 100 1000 10000 100000 1000000];

figure()
loglog(cycle,stres,'o')

S=@(a,x) (10^a(1)./x).^(1/a(2));
beta0=[13,2];
beta = nlinfit(cycle,stres,S,beta0);

hold on
N=[min(cycle):100:max(cycle)];
St=S(beta,N);
loglog(N,St)
grid on
```

4. The following data were collected for the steady flow of water in a concrete circular pipe:

Experiment	1	2	3	4	5	6	7	8	9
Diameter	0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
Slope	0.001	0.001	0.001	0.01	0.01	0.01	0.05	0.05	0.05
Flow	0.04	0.24	0.69	0.13	0.82	2.38	0.31	1.95	5.66

Use multiple linear regression to fit the following model to this data:

$$\mathrm{Q} = \alpha \mathrm{D}^{\mathrm{a}}\, \mathrm{S}^{\mathrm{b}}$$

where Q=flow, D=diameter, and S=slope. Represent the observed and predicted data.

We apply this expression:

$$\log Q = \log \alpha + \mathrm{a} \log \mathrm{D} + b \log S$$

```
D=[0.3 0.6 0.9 0.3 0.6 0.9 0.3 0.6 0.9];
S=[0.001 0.001 0.001 0.01 0.01 0.01 0.05 0.05 0.05];
Q=[0.04 0.24 0.69 0.13 0.82 2.38 0.31 1.95 5.66];
```

```
LD=log10(D);
LS=log10(S);
LQ=log10(Q);

X = [ones(length(LD),1) LD' LS'];
par = regress(LQ',X)
figure()
scatter3(D,S,Q,'filled')
hold on
x1fit = min(D):0.0001:max(D);
x2fit = min(S):0.0001:max(S);
[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);
alfa=10^par(1);
a= par(2);
b=par(3);
YFIT = alfa*X1FIT.^a.*X2FIT.^b ;
mesh(X1FIT,X2FIT,YFIT)
xlabel('Diameter')
ylabel('Slope')
zlabel('Flow')
```

References

- Martinez, W. L., & Martinez, A. R. (2001). *Computational statistics handbook with MATLAB*. Chapman and Hall/CRC.

- MathWorks, Inc. (2002). *Curve Fitting Toolbox: For Use with MATLAB®: User's Guide*. MathWorks.

- Jones, B. (1997). *MATLAB®: the language of technical computing; computing, visualization, programming. Statistics toolbox: user's guide*. Math Works Incorporated.