Computer Applications Lab
Lab 9
Probability, Statistics, Interpolation, and Calculus

Chapter 7

Dr. Iyad Jafar

Adapted from the publisher slides

Outline

- **Statistics and Probability**
- **Histograms**
- **Normal and Uniform Distributions**
- **Random Number Generation**
- **Interpolation**

Common Statistical Functions

- MATALB provides several functions to compute statistical properties of any set of numerical data

Function	Description
mean (x)	Computes the average value for the elements of vector x.
median(x)	Computes the median value for the elements in vector x.
std(x)	Computes the standard deviation of the elements in vector x.
cumsum(x)	Returns a vector the same size of x containing the cumulative sum of the elements in x.

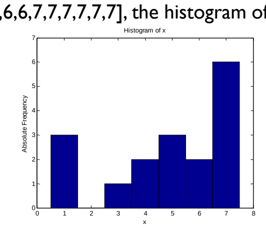
- For the first three commands, if x is a matrix, Matlab returns a raw vector in which each element is the result of applying these function to its columns. cumsum returns a matrix that contains cumulative sums of the columns.

Examples

```
>> x =[ones(1,5)*3,ones(1,4),ones(1,6)*(2)];
>> mean(x)
     ans = 31
>> median(x)
     ans = 2
>> std(x)
     ans = 0.7988
>> cumsum(x)
ans =  3     6     9    12    15    16    17
  18    19    21    23    25    27    29
  31
```

Histograms

- Given a set of data values stored in vector x, the histogram of x is a display of statistical information that uses rectangles to show the frequency of data items in successive numerical intervals of equal size.
- Histograms are useful in studying data properties and distribution as they can be used to approximate the probability function.
- Suppose x = [1,1,1,3,4,4,5,5,5,6,6,7,7,7,7,7,7], the histogram of these values is shown below.



Histograms

- To generate histograms in MATALB, use the
  - ***hist(x)*** *function or its variants (discussed later)*
  - ***bar(x,y),*** where x is a vector containing the centers of the bars (counts) and y is a vector containing the frequencies of the values in each bar.
- Example

```
>> x = [1,1,1,3,4,4,5,5,5,6,6,7,7,7,7,7,7];
>> hist(x,[0:8]); % here we specified the bin centers. If not provided, Matlab default is 10 bins between the min and max of input data
```

- OR

```
>> x = [0:8] ;  % specify bar centers
>> y = [0,3,0,1,2,3,2,6,0];  % absolute frequencies manually computed
>> bar(x,y);
```

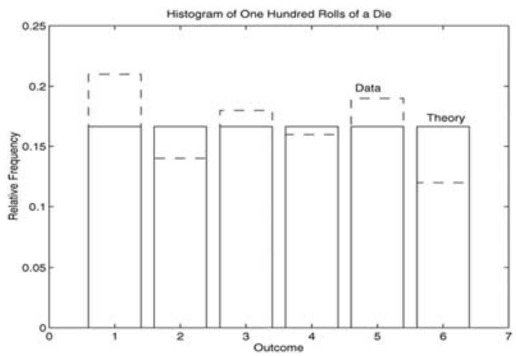

Histograms

Command	Description
bar(x,y)	Creates a bar chart of y versus x.
hist(y)	Aggregates the data in the vector y into 10 bins evenly spaced between the minimum and maximum values in y.
hist(y,n)	Aggregates the data in the vector y into n bins evenly spaced between the minimum and maximum values in y.
hist(y,x)	Aggregates the data in the vector y into bins whose center locations are specified by the vector x. The bin widths are the distances between the centers.
[z,x] = hist(y)	Same as hist(y) but returns two vectors z and x that contain the frequency count and the bin locations.
[z,x] = hist(y,n)	Same as hist(y,n) but returns two vectors z and x that contain the frequency count and the bin locations.
[z,x] = hist(y,x)	Same as hist(y,x) but returns two vectors z and x that contain the frequency count and the bin locations. The returned vector x is the same as the user-supplied vector x

Probability

- In any experiment or phenomena, the probability of an event is a number between 0 and 1 that indicates the likelihood of that event to occur if the experiment is repeated infinitely
- **Example:** Rolling a dice. The probability of obtaining any of the number on the dice is 1/6.
- Random variable is often the term used to express the outcome of an experiment. In rolling a dice, the random variable may take any of the values 1,2,3,4,5,6.
- Based on the experiment, these random variables could be continuous or discrete.

Probability - Example



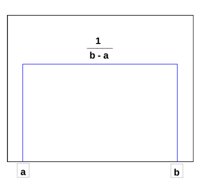
Probability – Probability Functions

- Random variables are usually associated with functions or distributions to characterize the experiment they come from.
- Such functions are called *probability density functions (pdf)*.
- If available, these distributions can be used to calculate the probability of the random variable to take certain value(s).
- The area under pdf is always equal to 1. this is based on the fact that we will obtain any of the possible outcomes if we run the experiment.

Common Distributions

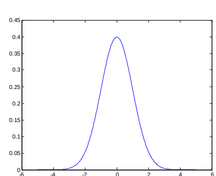
- Uniform Distribution

$$\mathrm{pdf}(x) = \begin{cases} \frac{1}{b-a} & , a \le x \le b \\ 0 & , \text{otherwise} \end{cases}$$



- Normal Distribution

$$\mathrm{pdf}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Computing Probability of an Event

- The probability that a random variable x is less than or equal than some value b is computed from the pdf by

$$P(x \le b) = \int_{-\infty}^{b} \mathrm{pdf}(x)dx = \mathrm{CDF}(x)$$

Continuous RV

$$P(x \le x_k) = \sum_{i=-\infty}^{k} \mathrm{pdf}(x_i)$$

Discrete RV

- This is usually referred as the cumulative distribution function CDF.
- For continuous random variables, the probability P(x=b) = 0.
- For discrete random variables, the probability $P(x=x_k)$ = $\mathrm{pdf}(x_k)$

Histograms as pdf

- For practical experiments, it is common to use the normalized histogram to represent the pdf of the experiment.
- The normalized histogram is defined as the absolute frequency histogram divided by total number of frequencies.

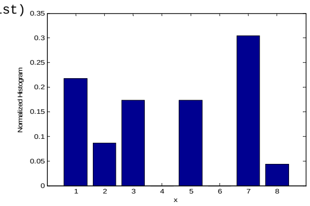
Example

- Suppose that x = [1,1,1,1,1,2,2,3,3,3,3,5,5,5,5,7,7,7,7,7,7,7,8]
- The following commands generates the normalized histogram of x with bin centers at 1 through 8.

```
>> [y,binLocations] = hist(x,[1:8]) ;
>> normhist = y / sum(y) ;  % or could use normhist = y / length(x)
>> label('x')
>> ylabel('Normalized Histogram')
```

- The probability of obtaining a number less than or equal to 3 can be found by

```
>> cdf = cumsum(normhist)
>> prob = cdf(3)
```



Sums and Differences of Random Variables

It can be proved that the mean of the sum (or difference) of two independent normally distributed random variables equals the sum (or difference) of their means, but the variance is always the sum of the two variances. That is, if x and y are normally distributed with means μ_x and μ_y, and variances σ_x and σ_y, and if $u = x + y$ and $\upsilon = x - y$, then

$$\mu_u = \mu_x + \mu_y$$

$$\mu_\upsilon = \mu_x - \mu_y$$

$$\sigma_u^2 = \sigma_\upsilon^2 = \sigma_x^2 + \sigma_y^2$$

Random Numbers

- Random numbers are extensively used in simulation and game programming.
- Matlab provides several functions to generate random numbers based on a given distribution.

Random Numbers

Command	Description
rand	Generates a single uniformly distributed random number between 0 and 1.
rand(n)	Generates an n x n matrix containing uniformly distributed random numbers between 0 and 1.
rand(m,n)	Generates an m x n matrix containing uniformly distributed random numbers between 0 and 1.
s = rand('state')	Returns a 35-element vector s containing the current state of the uniformly distributed generator.
rand('state',s)	Sets the state of the uniformly distributed generator to s.
rand('state',0)	Resets the uniformly distributed generator to its initial state.
rand('state',j)	Resets the uniformly distributed generator to state j, for integer j.
rand('state',sum(100*clock))	Resets the uniformly distributed generator to a different state each time it is executed.

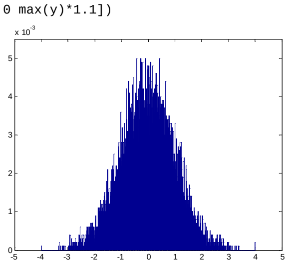
Random Numbers

Command	Description
randn	Generates a single normally distributed random number having a mean of 0 and a standard deviation of 1.
randn(n)	Generates an *n* ´ *n* matrix containing normally distributed random numbers having a mean of 0 and a standard deviation of 1.
randn(m,n)	Generates an *m* ´ *n* matrix containing normally distributed random numbers having a mean of 0 and a standard deviation of 1.
s = randn('state')	Like rand('state') but for the normally distributed generator
randn('state',s)	Like rand('state',s) but for the normally distributed generator.
randn('state',0)	Like rand('state',0) but for the normally distributed generator.
randn('state',j)	Like rand('state',j) but for the normally distributed generator.
randn('state',sum(100*clock))	Like rand('state',sum(100*clock)) but for the normally distributed generator.

Random Numbers

Example: check the operation of randn

```
>> x = randn(1,10000);
>> [y,bins] = hist(x,[-4:0.01:4]));
>> y = y / length(x)
>> Bar(bins,y)
>> axis([-5 5 0 max(y)*1.1])
```



Random Numbers

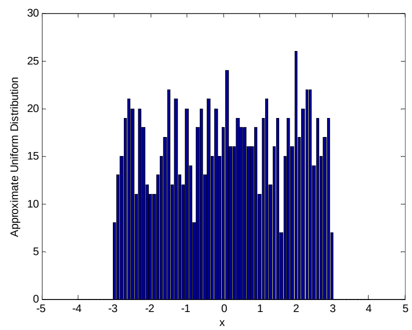
- To generate uniformly distributed random numbers in the interval [a,b], use

  y = (b-a) * x + a

  Where x is a set of uniformly distributed numbers generated by *rand* function.
- **Example :** the following session generates 1000 uniformly distributed numbers in the interval [-3,3].

```
>> x = rand(1,1000) ;
>> y = (3+3) * x - 3 ;
>> [yhist,bins] = hist(y,[-4:0.1:4]);
>> bar(bins,yhist);
```

Random Numbers



21

Random Numbers

- To generate normally distributed random numbers with a mean μ and standard deviation σ

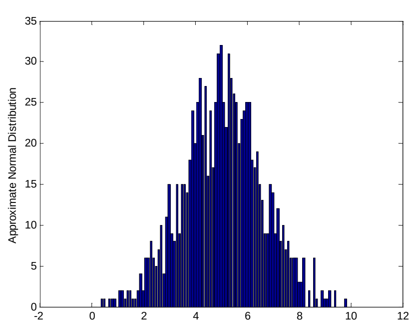
  $y = \sigma * x + \mu$

  Where x is a set of normally distributed numbers generated by *randn* function.
- **Example :** the following session generates 1000 normally distributed numbers with a mean of 5 and deviation of 1.5

```
>> x = randn(1,1000) ;
>> y = 1.5* x  + 5 ;
>> [yhist,bins] = hist(x,[0:0.1:10];
>> bar(bins,yhist);
```

22

Random Numbers



23

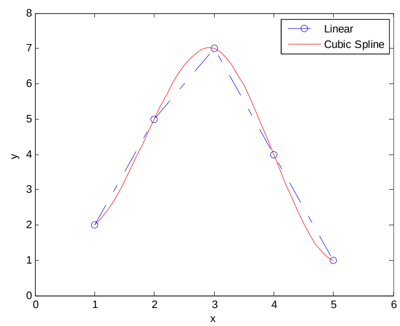
Interpolation

- Engineering problems often require the analysis of data pairs; cause and effect, or input output relationships.
- It is sometimes required to estimate variable's value between the given data point. This process is known as interpolation.
- Plotting the data greatly aids interpolation and extrapolation.

24

Interpolation

- Linear and cubic spline interpolation



Interpolation

Function	Description
`Y_int = interp1(x,y,x_int)`	Used to linearly interpolate a function of one variable: *y* = *f* (*x*). Returns a linearly interpolated vector `y_int` at the specified value `x_int`, using data stored in `x` and `y`.
`Z_int = interp2(x,y,z,x_int,y_int)`	Used to linearly interpolate a function of two variables: *y* = *f* (*x*, *y*). Returns a linearly interpolated vector `z_int` at the specified values `x_int` and `y_int`, using data stored in `x`, `y`, and `z`.
`Y_int = spline(x,y,x_int)`	Computes a cubic-spline interpolation where `x` and `y` are vectors containing the data and `x_int` is a vector containing the values of the independent variable *x* at which we wish to estimate the dependent variable *y*. The result `Y_int` is a vector the same size as `x_int` containing the interpolated values of *y* that correspond to `x_int`.

Interpolation

- **When using interp1 or spline functions, note that x should be sorted in ascending order and x_int values should be within the range of x.**
- **Example: assume that x = [1,2,3,4,5] and y = [2,5,8,10,6]. Estimate the value of y when x_int = [2.5,3.5].**

% using linear

```
>> y_int = interp1(x,y,x_int)
y_int =  6.5000    9.0000
```

% using cubic spline

```
>> y_int = spline(x,y,x_int)
y_int = 6.5000    9.3750
```