MATLAB Built-in Functions for Stat.

Mean, Median, Mode: Let $x = [x_1, x_2, \dots, x_n]^T$ be a vector of data values. Try the followings:

```
>> mean(x)

>> median(x)

>> mode(x)   % use help mode to see the other syntaxes for this function.
```

Note. To find all modes of a data set, we first find the smallest one and then remove it from the data set and follow the procedure for the remaining data values. To remove a specific value from a vector, use the following:

```
>> x=[2 3 4 -1 5 2 3 1 1 0];

>> x1=x(find(x~=5))

>>x(x==5)=[]
```

Measures of variation: Try the following

```
>> var(x); std(x); max(x); min(x);
```

`rand` function in MATLAB. This function generates a sequence of numbers that are uniformly distributed between 0 and 1. A simple representation of its syntax is:

```
>> r = rand(m, n)
```

where `r` = an $m \times n$ matrix of random numbers. To generate random numbers *uniformly distributed* between a and b, use the following.

```
>> r = a + (b-a)*rand(m, n)
```

The following script generates the random values for cd, along with their mean, standard deviation, and a histogram.

```
n = 10000;
cdmin = 0.225;
cdmax = 0.275
r = rand(n,1);
cdrand = cdmin+(cdmax - cdmin)*r;
meancd = mean(cdrand);
stdcd = std(cdrand);
hist(cdrand)   % default value for the n umber of bins =10 other wise use hist(cdrand,M)
title('Distribution of random numbers')
xlabel('bins')
```

`randn` function in MATLAB. This function generates a sequence of numbers that are normally distributed with a mean of 0 and standard deviation of 1 (standard normal distribution). A simple representation of its syntax is

```
>> r = randn(m, n)
```

where `r` = an *m*-by-*n* matrix of random numbers. To generate random numbers *normally distributed* with a mean of `mu` and a standard deviation of `sig`, use the following:

```
>> r = mu + sig*randn(m, n)
```

The following script generates the random values for cd, along with their mean, standard deviation, and a histogram.

```
n = 10000;
cd = 0.25;
stdev = 0.015
r = randn(n,1);
cdrand = cd + stdev*r;
meancd = mean(cdrand);
stdcd = std(cdrand);
hist(cdrand)   % default value for the n umber of bins =10 other wise use hist(cdrand,M)
title('Distribution of random numbers')
xlabel('bins')
```

Linear regression.

The following MATLAB function returns the coefficients of the line of best fit and coefficient of determination.

```
function [a, r2] = linregr(x,y)
% linregr: linear regression curve fitting
% [a, r2] = linregr(x,y): Least squares fit of straight
% line to data by solving the normal equations
% input:
% x = independent variable
% y = dependent variable
% output:
% a = vector of slope, a(1), and intercept, a(2)
% r2 = coefficient of determination
n = length(x);
if length(y)~= n,
    error('x and y must be same length');
end
x = x(:); y = y(:); % convert to column vectors
sx = sum(x);
sy = sum(y);
sx2 = sum(x.*x);
sxy = sum(x.*y);
sy2 = sum(y.*y);
a(1) = (n*sxy - sx*sy)/(n*sx2-sx^2);
a(2) = sy/n - a(1)*sx/n;
r2 = ((n*sxy - sx*sy)/sqrt(n*sx2 - sx^2)/sqrt(n*sy2 - sy^2))^2;
% create plot of data and best fit line
xp = linspace(min(x),max(x),2);
yp = a(1)*xp + a(2);
plot(x,y,'o',xp,yp)
grid on
end
```

MATLAB has a built-in function to return the linear regression model. The function

```
>> mdl = fitlm(x,y)
```

returns a linear regression model of the responses y, fit to the data array x. Try the following:

```
>> x = [10 20 30 40 50 60 70 80];

>> y = [25 70 380 550 610 1220 830 1450];
```

```
>> mdl = fitlm(x,y)
```

In the output:

- The model formula in the display, such as `y ~ 1 + x1 + x2 + x3`, that corresponds to $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$.
- Estimated Coefficient estimates the model's coefficients and provides further information. For example, the estimate for the constant term (intercept) is 47.977.
- SE — Standard error of the coefficients.
- tStat — *t*-statistic for each coefficient to test the null hypothesis that the corresponding coefficient is zero against the alternative that it is different from zero, given the other predictors in the model.
- pValue — *p*-value for the *t*-statistic of the two-sided hypothesis test.

MATLAB Functions: `polyfit` and `polyval`

MATLAB has a built-in function `polyfit` that fits a least-squares *n*th-order polynomial to data. It can be applied as in

```
>> p = polyfit(x, y, n)
```

where x and y are the vectors of the independent and the dependent variables, respectively, and n = the order of the polynomial. The function returns a vector p containing the polynomial's coefficients. We should note that it represents the polynomial using decreasing powers of x as in the following representation:

$$P(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1}$$

```
>> x = [10 20 30 40 50 60 70 80];
>> y = [25 70 380 550 610 1220 830 1450];
>> a = polyfit(x,y,1)
>> y = polyval(a,45)
```