Statistical Computing in Matlab

Bruce A. Desmarais

Odum Institute for Research in Social Science

February 2, 2009

Welcome!

Who am I?

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Basic Structure of Matlab

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Basic Structure of Matlab
- Low-level arithmetic tasks and programming language

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Basic Structure of Matlab
- Low-level arithmetic tasks and programming language
- Data management and basic description

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Basic Structure of Matlab
- Low-level arithmetic tasks and programming language
- Data management and basic description
- OLS and ANOVA

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Basic Structure of Matlab
- Low-level arithmetic tasks and programming language
- Data management and basic description
- OLS and ANOVA
- Optimization (application will be MLE)

What is Matlab?

What is Matlab?

- Matlab (Matrix laboratory) is an interactive software system for numerical computations and graphics

What is Matlab?

- Matlab (Matrix laboratory) is an interactive software system for numerical computations and graphics
- Designed for numerical (approximate) mathematics

What is Matlab?

- Matlab (Matrix laboratory) is an interactive software system for numerical computations and graphics
- Designed for numerical (approximate) mathematics
- Not (Math Lab); don't confuse with Maple or Mathematica

What is Matlab?

- Matlab (Matrix laboratory) is an interactive software system for numerical computations and graphics
- Designed for numerical (approximate) mathematics
- Not (Math Lab); don't confuse with Maple or Mathematica
- Combines accessable low-level environment with a wealth of well-designed high-level processes

What is Matlab?

- Matlab (Matrix laboratory) is an interactive software system for numerical computations and graphics
- Designed for numerical (approximate) mathematics
- Not (Math Lab); don't confuse with Maple or Mathematica
- Combines accessable low-level environment with a wealth of well-designed high-level processes
- Is free to you!! (or included in tuition; however you want to look at it)

General Structure

- If you are familiar with R; Matlab is very similar in structure

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented
- Everything can be (and probably should be) done from the command line (see caveat in a few bullets)

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented
- Everything can be (and probably should be) done from the command line (see caveat in a few bullets)
- Low-level data-manipulation utilities and numerical algorithms with base load

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented
- Everything can be (and probably should be) done from the command line (see caveat in a few bullets)
- Low-level data-manipulation utilities and numerical algorithms with base load
- Toolboxes can be loaded that contain specialized higher-level algorithms

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented
- Everything can be (and probably should be) done from the command line (see caveat in a few bullets)
- Low-level data-manipulation utilities and numerical algorithms with base load
- Toolboxes can be loaded that contain specialized higher-level algorithms
- Toolboxes cost $$, but the UNC license comes with many including the coveted "Stats".

General Structure

- If you are familiar with R; Matlab is very similar in structure
- When opened a basic command-line interface is presented
- Everything can be (and probably should be) done from the command line (see caveat in a few bullets)
- Low-level data-manipulation utilities and numerical algorithms with base load
- Toolboxes can be loaded that contain specialized higher-level algorithms
- Toolboxes cost $$, but the UNC license comes with many including the coveted "Stats".
- All work should actually be done through edited ".m" files or batches of Matlab code.

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single '%'

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by ‘%%’. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single ‘%’
- To send all of the code in a cell to the Matlab command line, place the cursor in that cell and press ‘Ctrl + Enter’.

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single '%'
- To send all of the code in a cell to the Matlab command line, place the cursor in that cell and press 'Ctrl + Enter'.
- It is not possible to send multiple cells at once without sending the whole file

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single '%'
- To send all of the code in a cell to the Matlab command line, place the cursor in that cell and press 'Ctrl + Enter'.
- It is not possible to send multiple cells at once without sending the whole file
- Lets create our first object, so we're all on the same page first type 'rand('seed',1)' 'Enter' then 'X = rand(100,4);'

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single '%'
- To send all of the code in a cell to the Matlab command line, place the cursor in that cell and press 'Ctrl + Enter'.
- It is not possible to send multiple cells at once without sending the whole file
- Lets create our first object, so we're all on the same page first type 'rand('seed',1)' 'Enter' then 'X = rand(100,4);'
- Now send it to the command line using 'Ctrl + Enter'

Using The Editor

First open the Matlab editor

- The matlab editor is organized into cells separated by '%%'. Place code in cells that you want to run all at once.
- You may commment your code preceding the line of text with a single '%'
- To send all of the code in a cell to the Matlab command line, place the cursor in that cell and press 'Ctrl + Enter'.
- It is not possible to send multiple cells at once without sending the whole file
- Lets create our first object, so we're all on the same page first type 'rand('seed',1)' 'Enter' then 'X = rand(100,4);'
- Now send it to the command line using 'Ctrl + Enter'
- Lets see what we have; create another cell with '%%', type the command 'whos' then 'Ctrl + Enter'

Data Types

- Numeric classes include double precision, single precision, Integer, Complex Numbers, Infinity and NaN.

Data Types

- Numeric classes include double precision, single precision, Integer, Complex Numbers, Infinity and NaN.
- Matlab 64-bit storage limits for double precison [-Inf, -1.79769e+308, -2.22507e-308, 0, 2.22507e-308, 1.79769e+308, Inf]

Data Types

- Numeric classes include double precision, single precision, Integer, Complex Numbers, Infinity and NaN.
- Matlab 64-bit storage limits for double precison [-Inf, -1.79769e+308, -2.22507e-308, 0, 2.22507e-308, 1.79769e+308, Inf]
- Mathematica and Maple Toolboxes are available for higher precision for $$.

Data Types

- Numeric classes include double precision, single precision, Integer, Complex Numbers, Infinity and NaN.
- Matlab 64-bit storage limits for double precison [-Inf, -1.79769e+308, -2.22507e-308, 0, 2.22507e-308, 1.79769e+308, Inf]
- Mathematica and Maple Toolboxes are available for higher precision for $$.
- Matlab evaluates a logical statement as (true =1 or false = 0) unlike other languages, these values are directly available for numerical manipulation

Data Types

- Numeric classes include double precision, single precision, Integer, Complex Numbers, Infinity and NaN.
- Matlab 64-bit storage limits for double precison [-Inf, -1.79769e+308, -2.22507e-308, 0, 2.22507e-308, 1.79769e+308, Inf]
- Mathematica and Maple Toolboxes are available for higher precision for $$.
- Matlab evaluates a logical statement as (true =1 or false = 0) unlike other languages, these values are directly available for numerical manipulation
- The basic logical operators are (==,∼=,>,<,>=,<=), create another cell and type '((3>2)+(1/3==2/6))∧2'

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter 'M1=[1 2 3;4 5 6]'.

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter 'M1=[1 2 3;4 5 6]'.
- High - Dimensional arrays are created by concatenating 'cat()' lower dimensional arrays along higher dimensions 'cat(d,a1,a2,a3..an)' where d is dimension and 'a's are arrays.

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter ‘M1=[1 2 3;4 5 6]’.
- High - Dimensional arrays are created by concatenating ‘cat()’ lower dimensional arrays along higher dimensions ‘cat(d,a1,a2,a3..an)’ where d is dimension and ‘a’s are arrays.
- Arrays can be of arbitrary dimension; great for looping through matrices (e.g. data sets)

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter ‘M1=[1 2 3;4 5 6]’.
- High - Dimensional arrays are created by concatenating ‘cat()’ lower dimensional arrays along higher dimensions ‘cat(d,a1,a2,a3..an)’ where d is dimension and ‘a’s are arrays.
- Arrays can be of arbitrary dimension; great for looping through matrices (e.g. data sets)
- Define ‘M2=[7 8 9;10 11 12]’ now ‘A=cat(3,M1,M2)’ now ‘whos’

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter 'M1=[1 2 3;4 5 6]'.
- High - Dimensional arrays are created by concatenating 'cat()' lower dimensional arrays along higher dimensions 'cat(d,a1,a2,a3..an)' where d is dimension and 'a's are arrays.
- Arrays can be of arbitrary dimension; great for looping through matrices (e.g. data sets)
- Define 'M2=[7 8 9;10 11 12]' now 'A=cat(3,M1,M2)' now 'whos'
- To reference array elements 'A(d1,d2,d3...)' to reference a single element, substitute a ':' for the 'di' to reference all elements along a dimension and 'dia:dib' to reference a subset along a dimension.

Vectors, Matrices and HD Arrays

- Semi-colons separate elements along rows, spaces along columns; in a new cell enter 'M1=[1 2 3;4 5 6]'.
- High - Dimensional arrays are created by concatenating 'cat()' lower dimensional arrays along higher dimensions 'cat(d,a1,a2,a3..an)' where d is dimension and 'a's are arrays.
- Arrays can be of arbitrary dimension; great for looping through matrices (e.g. data sets)
- Define 'M2=[7 8 9;10 11 12]' now 'A=cat(3,M1,M2)' now 'whos'
- To reference array elements 'A(d1,d2,d3...)' to reference a single element, substitute a ':' for the 'di' to reference all elements along a dimension and 'dia:dib' to reference a subset along a dimension.
- Lets try it all; type 'A(1,1:3,:)'

For

For

The for loop syntax is as follows 'for k=start:increment:end *algorithm* end'. Lets try one. Type the following code in a new cell:

For

The for loop syntax is as follows 'for k=start:increment:end *algorithm* end'. Lets try one. Type the following code in a new cell:

```
b = 1;
for i = 1:1:10
b = (b+i)^1.2;
end
b
```

For

The for loop syntax is as follows 'for k=start:increment:end *algorithm* end'. Lets try one. Type the following code in a new cell:

```
b = 1;
for i = 1:1:10
b = (b+i)^1.2;
end
b
```

Now send this to the command line and see if you get $6.0089e + 004$.

While

While

The while loop syntax is as follows 'while (logical condition) *algorithm* end'. Lets try one. Type the following code in a new cell:

While

The while loop syntax is as follows 'while (logical condition) *algorithm* end'. Lets try one. Type the following code in a new cell:

```
b = 1;
i =1;
while (b < 100000)
b = (b+i)^1.2;
i = i+1;
end
b
i
```

While

The while loop syntax is as follows 'while (logical condition) *algorithm* end'. Lets try one. Type the following code in a new cell:

```
b = 1;
i =1;
while (b < 100000)
b = (b+i)^1.2;
i = i+1;
end
b
i
```

Now send this to the command line and see if you get '5.4281e+005' and '12'.

If

If

If statements and its extensions are used as follows:

If

If statements and its extensions are used as follows:

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

If

If statements and its extensions are used as follows:

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

```
if (1 > 2)
     a = 2;
elseif (2 > 2)
    a = 2;
else
    a = 1;
end
a
```

If

If statements and its extensions are used as follows:

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

```
if (1 > 2)
     a = 2;
elseif (2 > 2)
    a = 2;
else
    a = 1;
end
a
```

Note: Compound logical '(A || B)' = A or B and '(A && B)' = A and B.

Matrices I

Addition/Subtraction

A +/- B

Matrices I

Addition/Subtraction

A +/- B

Multiplication

A*B

Matrices I

Addition/Subtraction

A +/- B

Multiplication

A*B

Transposition

A'

Matrices I

Addition/Subtraction
A +/- B

Multiplication
A*B

Transposition
A'

Inverse
inv(A)

Matrices I

Addition/Subtraction
A +/- B

Multiplication
A*B

Transposition
A'

Inverse
inv(A)

Determinant
det(A)

Matrices I

Addition/Subtraction
A +/- B

Multiplication
A*B

Transposition
A'

Inverse
inv(A)

Determinant
det(A)

Eigenvalues/Vectors
$D =$ eig(A)
$[V, D] =$ eig(A)

Matrices I

Addition/Subtraction
A +/- B

Multiplication
A*B

Transposition
A'

Inverse
inv(A)

Determinant
det(A)

Eigenvalues/Vectors
$D =$ eig(A)
$[V, D] =$ eig(A)

Matrices II

Matrix Power

A∧p

Matrices II

Matrix Power

A$\wedge$p

Diagonal

diag(A)

B = diag(a1,a2..an)

Matrices II

Matrix Power

A$\wedge$p

Diagonal

diag(A)

B = diag(a1,a2..an)

Upper Triangle

triu(A)

Matrices II

Matrix Power
A∧p

Lower Triangle
tril(A)

Diagonal
diag(A)
B = diag(a1,a2..an)

Upper Triangle
triu(A)

Matrices II

Matrix Power
A∧p

Diagonal
diag(A)
B = diag(a1,a2..an)

Upper Triangle
triu(A)

Lower Triangle
tril(A)

Identity Matrix
B = eye(N)

Matrices II

Matrix Power
A∧p

Diagonal
diag(A)
B = diag(a1,a2..an)

Upper Triangle
triu(A)

Lower Triangle
tril(A)

Identity Matrix
B = eye(N)

Ones, Zeros Matrices
'A = ones(M,N)'
'B=zeros(M,N)'

Matrices II

Matrix Power
A∧p

Diagonal
diag(A)
B = diag(a1,a2..an)

Upper Triangle
triu(A)

Lower Triangle
tril(A)

Identity Matrix
B = eye(N)

Ones, Zeros Matrices
‘A = ones(M,N)’
‘B=zeros(M,N)’

Function Composition

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path
2. File should be named *function_name*.m

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path
2. File should be named *function_name*.m
3. To add directory addpath('dir','dir2','dir3' ...) to remove rmpath('directory')

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path
2. File should be named *function_name*.m
3. To add directory addpath('dir','dir2','dir3' ...) to remove rmpath('directory')
4. Function definition is ‘function [Outputs] = function_name(inputs) Operations’

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path
2. File should be named *function_name*.m
3. To add directory addpath('dir','dir2','dir3' ...) to remove rmpath('directory')
4. Function definition is 'function [Outputs] = function_name(inputs) Operations'
5. No "return" functions are used, Matlab simply returns the last assignment of [Outputs]

Function Composition

1. Save a function file to a directory (folder) and add directory to Matlab Path
2. File should be named *function_name*.m
3. To add directory addpath('dir','dir2','dir3' ...) to remove rmpath('directory')
4. Function definition is 'function [Outputs] = function_name(inputs) Operations'
5. No "return" functions are used, Matlab simply returns the last assignment of [Outputs]
6. Functions can access Local variables (those defined within), global variables and (inputs)

Example Function

Example Function

1. Open a new Matlab Editor

Example Function

1. Open a new Matlab Editor
2. Save it as exlog.m to whatever directory you'd like

Example Function

1. Open a new Matlab Editor
2. Save it as exlog.m to whatever directory you'd like
3. Add directory to the search path

```
function [ln,expo] = exlog(x)
expo = 2;
ln = log(x);
expo = exp(x);
```

From your working m-file execute exlog(3)

Example Function II

There's something wrong with exlog()'s returned values.
In Matlab '[a1,a2...an] = f(x1,x2..xn)' assigns outputs of f() to a1,a2..an
To return both outputs to a single vector from exlog():

```
function [lnexpo] = exlog(x)
lnexpo = [log(x) exp(x)];
```

Now try it

Saving and Loading

To save X Y Z...to *filename*

save *filename* X Y Z...

```
save C:\Users\Owner\Documents\a.mat A
```

Saving and Loading

To save X Y Z...to *filename*

save *filename* X Y Z...

```
save C:\Users\Owner\Documents\a.mat A
```

To load all variables from *filename*

load *filename*

Saving and Loading

To save X Y Z...to *filename*

save *filename* X Y Z...

```
save C:\Users\Owner\Documents\a.mat A
```

To load all variables from *filename*

load *filename*

Load only X Y Z..

load *filename* X Y Z..

Description and Exploration I

Function	Statistic
mean()	mean

Description and Exploration I

Function	Statistic
mean()	mean
median()	median

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation
var()	variance

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation
var()	variance
cov()	covariance matrix

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation
var()	variance
cov()	covariance matrix
corr()	correlation matrix

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation
var()	variance
cov()	covariance matrix
corr()	correlation matrix
cov2corr()	convert cov to corr

Description and Exploration I

Function	Statistic
mean()	mean
median()	median
mode()	mode
max()	maximum
min()	minimum
range()	max()-min()
std()	standard deviation
var()	variance
cov()	covariance matrix
corr()	correlation matrix
cov2corr()	convert cov to corr
corr2cov()	" "

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf
trimmean(x,pct)	trimmed mean

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf
trimmean(x,pct)	trimmed mean
IQR()	Interquartile (75-25) range

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf
trimmean(x,pct)	trimmed mean
IQR()	Interquartile (75-25) range
geommean()	geometric mean (stats)

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf
trimmean(x,pct)	trimmed mean
IQR()	Interquartile (75-25) range
geommean()	geometric mean (stats)
harmmean()	harmonic mean (stats)

Description and Exploration II

Function	Statistic
scatter(y,x)	scatter plot
prctile(x,[10,20..])	percentiles
skewness()	skewness
kurtosis()	kurtosis
hist()	histogram
histfit()	hist with normal
cdfplot()	empirical cdf
trimmean(x,pct)	trimmed mean
IQR()	Interquartile (75-25) range
geommean()	geometric mean (stats)
harmmean()	harmonic mean (stats)
moment(x,order)	central moments (stats)

Anova1: One Way Anova

- 'P = anova1(X,GROUP)' returns the p-value for the null hypothesis that the means of the groups are equal.

Anova1: One Way Anova

- 'P = anova1(X,GROUP)' returns the p-value for the null hypothesis that the means of the groups are equal.
- X must be categorical. If it isn't use 'Xc=nominal(X)'

Anova1: One Way Anova

- 'P = anova1(X,GROUP)' returns the p-value for the null hypothesis that the means of the groups are equal.
- X must be categorical. If it isn't use 'Xc=nominal(X)'
- '[P,ANOVATAB] = anova1(...)' returns the ANOVA table values as the cell array ANOVATAB.

Anova1: One Way Anova

- 'P = anova1(X,GROUP)' returns the p-value for the null hypothesis that the means of the groups are equal.
- X must be categorical. If it isn't use 'Xc=nominal(X)'
- '[P,ANOVATAB] = anova1(...)' returns the ANOVA table values as the cell array ANOVATAB.
- '[P,ANOVATAB,STATS] = anova1(...)' returns the additional 'stats' which is used with the MULTCOMPARE function.

Anova1: One Way Anova

- 'P = anova1(X,GROUP)' returns the p-value for the null hypothesis that the means of the groups are equal.
- X must be categorical. If it isn't use 'Xc=nominal(X)'
- '[P,ANOVATAB] = anova1(...)' returns the ANOVA table values as the cell array ANOVATAB.
- '[P,ANOVATAB,STATS] = anova1(...)' returns the additional 'stats' which is used with the MULTCOMPARE function.
- Lets try it 'rand('seed',1)' then 'randn('seed',1)' 'x=floor(rand(100,1))' then 'y= x + randn(100,1)' now run the most extensive anova of y on x.

Anova2: Two Way Anova

- 'P = anova2(X,REPS)' returns the p-value for the null hypotheses (group1, group2, interaction) in a *balanced* two-way anova.

Anova2: Two Way Anova

- ‘P = anova2(X,REPS)’ returns the p-value for the null hypotheses (group1, group2, interaction) in a *balanced* two-way anova.
- ‘X’ is a matrix where the columns indicate group 1 membership and the rows indicate group 2

Anova2: Two Way Anova

- ‘P = anova2(X,REPS)’ returns the p-value for the null hypotheses (group1, group2, interaction) in a *balanced* two-way anova.
- ‘X’ is a matrix where the columns indicate group 1 membership and the rows indicate group 2
- ‘REPS’ is the number of observations that occuppy each cell (must be constant). X must be $P * REPSxK$ where P is the number of categories in group 1and K is the number in group 2. Cell entries should be Y values.

Anova2: Two Way Anova

- 'P = anova2(X,REPS)' returns the p-value for the null hypotheses (group1, group2, interaction) in a *balanced* two-way anova.
- 'X' is a matrix where the columns indicate group 1 membership and the rows indicate group 2
- 'REPS' is the number of observations that occuppy each cell (must be constant). X must be $P * REPSxK$ where P is the number of categories in group 1and K is the number in group 2. Cell entries should be Y values.
- 'P=ANOVAN(Y,GROUP,'PARAM1',val1,'PARAM2',val2,...)'

Anova2: Two Way Anova

- ‘P = anova2(X,REPS)’ returns the p-value for the null hypotheses (group1, group2, interaction) in a *balanced* two-way anova.
- ‘X’ is a matrix where the columns indicate group 1 membership and the rows indicate group 2
- ‘REPS’ is the number of observations that occuppy each cell (must be constant). X must be $P * REPSxK$ where P is the number of categories in group 1and K is the number in group 2. Cell entries should be Y values.
- ‘P=ANOVAN(Y,GROUP,’PARAM1’,val1,’PARAM2’,val2,...)’
- Lets try it ‘randn(‘seed’,1)’ ‘XA2=randn(1000,100)’ then run a two-way ANOVA with REPS = 10.

Anovan: N Way Anova

- 'P = anovan(X,GROUP)' returns the p-values for the null hypotheses of no group effects in a muli-way anova.

Anovan: N Way Anova

- ‘P = anovan(X,GROUP)’ returns the p-values for the null hypotheses of no group effects in a muli-way anova.
- ‘X’ is a vector of outcome values

Anovan: N Way Anova

- 'P = anovan(X,GROUP)' returns the p-values for the null hypotheses of no group effects in a muli-way anova.
- 'X' is a vector of outcome values
- 'GROUP' is akward..It is a cell array constructed as 'C = G1 G2 G3..GN' where each G# is a vector of group indicators the same length of X

Anovan: N Way Anova

- 'P = anovan(X,GROUP)' returns the p-values for the null hypotheses of no group effects in a muli-way anova.
- 'X' is a vector of outcome values
- 'GROUP' is akward..It is a cell array constructed as 'C = G1 G2 G3..GN' where each G# is a vector of group indicators the same length of X
- 'P=ANOVAN(Y,GROUP,'PARAM1',val1,'PARAM2',val2,...)' Offers an extensive number of options. Check 'help anovan' for parameter values.

Anovan: N Way Anova

- ‘P = anovan(X,GROUP)’ returns the p-values for the null hypotheses of no group effects in a muli-way anova.
- ‘X’ is a vector of outcome values
- ‘GROUP’ is akward..It is a cell array constructed as ‘C = G1 G2 G3..GN’ where each G# is a vector of group indicators the same length of X
- ‘P=ANOVAN(Y,GROUP,'PARAM1',val1,'PARAM2',val2,...)’ Offers an extensive number of options. Check ‘help anovan’ for parameter values.
- Lets try this. First generate a cell array ‘C={(randn(100,1)>0) (randn(100,1)>0) (randn(100,1)>0)};’ then ‘x = randn(100,1);’ then...

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X
- '[B, BINT] = regress(Y,X)' returns coefficeints and 95% Confidence Intervals

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X
- '[B, BINT] = regress(Y,X)' returns coefficeints and 95% Confidence Intervals
- '[B,BINT,R] = regress(Y,X)' returns residuals as well

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X
- '[B, BINT] = regress(Y,X)' returns coefficeints and 95% Confidence Intervals
- '[B,BINT,R] = regress(Y,X)' returns residuals as well
- '[B,BINT,R,RINT] = regress(Y,X)' returns 95% CI's for residuals (no zero means outlier)

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X
- '[B, BINT] = regress(Y,X)' returns coefficeints and 95% Confidence Intervals
- '[B,BINT,R] = regress(Y,X)' returns residuals as well
- '[B,BINT,R,RINT] = regress(Y,X)' returns 95% CI's for residuals (no zero means outlier)
- '[B,BINT,R,RINT,STATS] = regress(Y,X)' adds 'stats' which contains R^2, F-Stat and P-value

Ordinary Least Squares

- 'B=regress(Y,X)' returns the vector of coefficients from regressing Y on the matrix X
- '[B, BINT] = regress(Y,X)' returns coefficeints and 95% Confidence Intervals
- '[B,BINT,R] = regress(Y,X)' returns residuals as well
- '[B,BINT,R,RINT] = regress(Y,X)' returns 95% CI's for residuals (no zero means outlier)
- '[B,BINT,R,RINT,STATS] = regress(Y,X)' adds 'stats' which contains R^2, F-Stat and P-value
- Generate 'X=randn(100,1);' and 'Y=randn(100,1)+X;' now regress Y on X and see what you get

More OLS

- Practitioners often want additional information. The ‘regstats’ function provides this.

More OLS

- Practitioners often want additional information. The 'regstats' function provides this.
- Usage is 'regstats(Y,X,{stat1 stat2...statn})' note spaces (another cell array)

More OLS

- Practitioners often want additional information. The 'regstats' function provides this.
- Usage is 'regstats(Y,X,{stat1 stat2...statn})' note spaces (another cell array)
- The list of optional stats is extensive; just to name a few:

More OLS

- Practitioners often want additional information. The ‘regstats’ function provides this.
- Usage is ‘regstats(Y,X,{stat1 stat2...statn})’ note spaces (another cell array)
- The list of optional stats is extensive; just to name a few:

More OLS

- Practitioners often want additional information. The 'regstats' function provides this.
- Usage is 'regstats(Y,X,{stat1 stat2...statn})' note spaces (another cell array)
- The list of optional stats is extensive; just to name a few:

Name	Meaning
'covb'	Covariance of regression coefficients
'yhat'	Fitted values of the response data
'adjrsquare'	Adjusted R-square statistic
'dfbetas'	Scaled change in regression coefficients
'cookd'	Cook's distance
'tstat'	t statistics for coefficients

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models
- Options for DISTR: 'normal', 'binomial', 'poisson', 'gamma', and 'inverse gaussian'

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models
- Options for DISTR: 'normal', 'binomial', 'poisson', 'gamma', and 'inverse gaussian'
- PARAM's include link with options: 'identity', 'log', 'logit', 'probit', 'comploglog', 'reciprocal', 'loglog'

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models
- Options for DISTR: 'normal', 'binomial', 'poisson', 'gamma', and 'inverse gaussian'
- PARAM's include link with options: 'identity', 'log', 'logit', 'probit', 'comploglog', 'reciprocal', 'loglog'
- Also, for the binomial and Poisson the option ‘estdisp’ can be set to estimate a dispersion parameter (‘on’ or ‘off’)

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models
- Options for DISTR: 'normal', 'binomial', 'poisson', 'gamma', and 'inverse gaussian'
- PARAM's include link with options: 'identity', 'log', 'logit', 'probit', 'comploglog', 'reciprocal', 'loglog'
- Also, for the binomial and Poisson the option ‘estdisp’ can be set to estimate a dispersion parameter (‘on’ or ‘off’)
- ‘constant’ is another PARAM that can be set to estimate a constant

Generalized Linear Models

- ‘B = GLMFIT(X,Y,DISTR,'PARAM1',val1,'PARAM2',val2,...)’ fits generalized linear models
- Options for DISTR: 'normal', 'binomial', 'poisson', 'gamma', and 'inverse gaussian'
- PARAM's include link with options: 'identity', 'log', 'logit', 'probit', 'comploglog', 'reciprocal', 'loglog'
- Also, for the binomial and Poisson the option ‘estdisp’ can be set to estimate a dispersion parameter (‘on’ or ‘off’)
- ‘constant’ is another PARAM that can be set to estimate a constant
- ‘[B,DEV,STATS] = GLMFIT(...)’ returns many useful quantities including the covariance matrix

Probit in Matlab

1. First we need to generate some data

Probit in Matlab

1. First we need to generate some data
2. ‘x = randn(1000,1);’ and ‘y = (x+randn(1000,1)¿0);’

Probit in Matlab

1. First we need to generate some data
2. 'x = randn(1000,1);' and 'y = (x+randn(1000,1)¿0);'
3. This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on x

Probit in Matlab

1. First we need to generate some data
2. ‘x = randn(1000,1);’ and ‘y = (x+randn(1000,1)¿0);’
3. This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model ‘[B,DEV,STATS] = GLMFIT(x,y,‘binomial’,‘link’,‘probit’)’

Probit in Matlab

1. First we need to generate some data
2. ‘x = randn(1000,1);’ and ‘y = (x+randn(1000,1)¿0);’
3. This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model ‘[B,DEV,STATS] = GLMFIT(x,y,‘binomial’,‘link’,‘probit’)’
5. Did we do well?

Probit in Matlab

1. First we need to generate some data
2. ‘x = randn(1000,1);’ and ‘y = (x+randn(1000,1)¿0);’
3. This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model ‘[B,DEV,STATS] = GLMFIT(x,y,‘binomial’,‘link’,‘probit’)’
5. Did we do well?
6. Now use ‘YHAT = glmval(B,X,LINK)’ to produce predicted means

Poisson Regression in Matlab

1. Generate Data Again

Poisson Regression in Matlab

1. Generate Data Again
2. ‘x = randn(1000,1);’ then ‘u=randn(1000,1);’ and ‘mean=exp(2-x+u);’ lastly ‘y=poissrnd(mean);’

Poisson Regression in Matlab

1. Generate Data Again
2. ‘x = randn(1000,1);’ then ‘u=randn(1000,1);’ and ‘mean=exp(2-x+u);’ lastly ‘y=poissrnd(mean);’
3. This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on x

Poisson Regression in Matlab

1. Generate Data Again
2. 'x = randn(1000,1);' then 'u=randn(1000,1);' and 'mean=exp(2-x+u);' lastly 'y=poissrnd(mean);'
3. This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model '[B,DEV,STATS] = GLMFIT(x,y,'poisson','estdisp','on')'

Poisson Regression in Matlab

1. Generate Data Again
2. ‘x = randn(1000,1);’ then ‘u=randn(1000,1);’ and ‘mean=exp(2-x+u);’ lastly ‘y=poissrnd(mean);’
3. This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model ‘[B,DEV,STATS] = GLMFIT(x,y,’poisson’,’estdisp’,’on’)’
5. Did we do well?

Poisson Regression in Matlab

1. Generate Data Again
2. 'x = randn(1000,1);' then 'u=randn(1000,1);' and 'mean=exp(2-x+u);' lastly 'y=poissrnd(mean);'
3. This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on x
4. Now we estimate the model '[B,DEV,STATS] = GLMFIT(x,y,'poisson','estdisp','on')'
5. Did we do well?
6. Now compare se's with and without dispersion 'STATS.se'