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?

Basic Structure of Matlab
- Low-level arithmetic tasks and programming language
- Data management and basic description
- OLS and ANOVA
- Optimization (application will be MLE)
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
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
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 accessible 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 accessible 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)
If you are familiar with R; Matlab is very similar in structure
If you are familiar with R; Matlab is very similar in structure

- When opened a basic command-line interface is presented
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)
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
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”.
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.

```matlab
rand('seed',1)
X = rand(100,4);
whos
```
**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 comment your code preceding the line of text with a single ‘%’
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 comment 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’.

Let's 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'. Let's see what we have; create another cell with ‘%%’, type the command ‘whos’ then ‘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 comment 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 comment 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 comment 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 comment 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 precision [-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 precision \([-\text{Inf}, -1.79769e+308, -2.22507e-308, 0, 2.22507e-308, 1.79769e+308, \text{Inf}]\)
- 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 precision [-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 precision [-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’
Semi-colons separate elements along rows, spaces along columns; in a new cell enter `M1=[1 2 3;4 5 6]`.

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,:)`
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`
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,:)’
The for loop syntax is as follows: `for k=start:increment:end
algorithm
end`.

Let's try one. Type the following code in a new cell:

```matlab
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`. 
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:
The for loop syntax is as follows ‘for \( k =\text{start}:\text{increment}:\text{end} \) algorithm end’. Let's try one. Type the following code in a new cell:

```matlab
b = 1;
for i = 1:1:10
    b = (b+i)^1.2;
end
b
```
The for loop syntax is as follows ‘for k=start:increment:end algorithm end’. Let’s try one. Type the following code in a new cell:

```matlab
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**

The while loop syntax is as follows: `while (logical condition) algorithm end`. Let's try one.

Type the following code in a new cell:

```matlab
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'`.
**While**

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

```matlab
b = 1;
i = 1;
while (b < 100000)
    b = (b+i)^1.2;
i = i+1;
end
b
i
```
The while loop syntax is as follows ‘while (logical condition) 
algorithm end’. Lets try one. Type the following code in a new cell:

```matlab
b = 1;
i = 1;
while (b < 100000)
b = (b+i)^1.2;
i = i+1;
end
b
i
```
The while loop syntax is as follows ‘while (logical condition) algorithm end’. Let's try one. Type the following code in a new cell:

```matlab
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 statements and its extensions are used as follows:

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

```matlab
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.
If statements and its extensions are used as follows:

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

Note: Compound logical '(A || B)' = A or B and '(A && B)' = A and B.
If statements and its extensions are used as follows:

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

Note: Compound logical `(A || B) = A or B` and `(A && B) = A and B`.
If statements and its extensions are used as follows:

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

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

Note: Compound logical ‘(A || B)’ = A or B and ‘(A && B)’ = A and B.
If statements and its extensions are used as follows:

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

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

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 \pm B

Multiplication
A*B

Transposition
A'
Matrix Operations

**Addition/Subtraction**
A +/− B

**Multiplication**
A*B

**Transposition**
A’

**Inverse**
inv(A)

**Determinant**
det(A)

**Eigenvalues/Vectors**
\[D, V] = eig(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
\[ D = \text{eig}(A) \]
\[ [V, D] = \text{eig}(A) \]
Matrices I

**Addition/Subtraction**
A +/- B

**Multiplication**
A*B

**Transposition**
A'

**Inverse**
inv(A)

**Determinant**
det(A)

**Eigenvalues/Vectors**

\[ D = \text{eig}(A) \]

\[ [V, D] = \text{eig}(A) \]
Matrix Power
$A^p$
Matrices II

Matrix Power
$A^p$

Diagonal
$\text{diag}(A)$
$B = \text{diag}(a_1, a_2, \ldots, a_n)$
Matrices II

Matrix Power
A^p

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

Upper Triangle
triu(A)
Matrix Power
\[ A^p \]

Lower Triangle
\[ \text{tril}(A) \]

Diagonal
\[ \text{diag}(A) \]
\[ B = \text{diag}(a_1, a_2, \ldots, a_n) \]

Upper Triangle
\[ \text{triu}(A) \]
**Matrices II**

**Matrix Power**
\[ A^p \]

**Diagonal**
\[ \text{diag}(A) \]
\[ B = \text{diag}(a_1, a_2, \ldots, a_n) \]

**Upper Triangle**
\[ \text{triu}(A) \]

**Lower Triangle**
\[ \text{tril}(A) \]

**Identity Matrix**
\[ B = \text{eye}(N) \]
Matrices II

Matrix Power
\( A^p \)

Lower Triangle
\( \text{tril}(A) \)

Diagonal
\( \text{diag}(A) \)
\( B = \text{diag}(a_1, a_2, \ldots, a_n) \)

Identity Matrix
\( B = \text{eye}(N) \)

Upper Triangle
\( \text{triu}(A) \)

Ones, Zeros Matrices
\( A = \text{ones}(M, N) \)
\( B = \text{zeros}(M, N) \)
Matrices II

Matrix Power
\[ A^p \]

Diagonal
\[ \text{diag}(A) \]
\[ B = \text{diag}(a_1, a_2, \ldots, a_n) \]

Upper Triangle
\[ \text{triu}(A) \]

Lower Triangle
\[ \text{tril}(A) \]

Identity Matrix
\[ B = \text{eye}(N) \]

Ones, Zeros Matrices
\[ 'A = \text{ones}(M,N)' \]
\[ 'B = \text{zeros}(M,N)' \]
**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 \texttt{function\_name.m}
3. To add directory \texttt{addpath('dir','dir2','dir3' \ldots)} to remove \texttt{rmpath('directory')}
4. Function definition is ‘function \texttt{[Outputs] = function\_name(inputs) Operations’}
5. No “return” functions are used, Matlab simply returns the last assignment of \texttt{[Outputs]}
6. Functions can access Local variables (those defined within), global variables and (inputs)
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

\[
\begin{align*}
\text{function } & [\text{ln,expo}] = \text{exlog}(x) \\
\text{expo} & = 2; \\
\text{ln} & = \log(x); \\
\text{expo} & = \exp(x);
\end{align*}
\]

From your working m-file execute \text{exlog}(3)
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

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

From your working m-file execute `exlog(3)`.
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

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

From your working m-file execute exlog(3)
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
</tbody>
</table>
# Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mean()</code></td>
<td>mean</td>
</tr>
<tr>
<td><code>median()</code></td>
<td>median</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
</tbody>
</table>
## Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
</tbody>
</table>
## Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
<tr>
<td>std()</td>
<td>standard deviation</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
<tr>
<td>std()</td>
<td>standard deviation</td>
</tr>
<tr>
<td>var()</td>
<td>variance</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mean()</code></td>
<td>mean</td>
</tr>
<tr>
<td><code>median()</code></td>
<td>median</td>
</tr>
<tr>
<td><code>mode()</code></td>
<td>mode</td>
</tr>
<tr>
<td><code>max()</code></td>
<td>maximum</td>
</tr>
<tr>
<td><code>min()</code></td>
<td>minimum</td>
</tr>
<tr>
<td><code>range()</code></td>
<td><code>max()-min()</code></td>
</tr>
<tr>
<td><code>std()</code></td>
<td>standard deviation</td>
</tr>
<tr>
<td><code>var()</code></td>
<td>variance</td>
</tr>
<tr>
<td><code>cov()</code></td>
<td>covariance matrix</td>
</tr>
</tbody>
</table>
## Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
<tr>
<td>std()</td>
<td>standard deviation</td>
</tr>
<tr>
<td>var()</td>
<td>variance</td>
</tr>
<tr>
<td>cov()</td>
<td>covariance matrix</td>
</tr>
<tr>
<td>corr()</td>
<td>correlation matrix</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
<tr>
<td>std()</td>
<td>standard deviation</td>
</tr>
<tr>
<td>var()</td>
<td>variance</td>
</tr>
<tr>
<td>cov()</td>
<td>covariance matrix</td>
</tr>
<tr>
<td>corr()</td>
<td>correlation matrix</td>
</tr>
<tr>
<td>cov2corr()</td>
<td>convert cov to corr</td>
</tr>
</tbody>
</table>
### Description and Exploration I

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean()</td>
<td>mean</td>
</tr>
<tr>
<td>median()</td>
<td>median</td>
</tr>
<tr>
<td>mode()</td>
<td>mode</td>
</tr>
<tr>
<td>max()</td>
<td>maximum</td>
</tr>
<tr>
<td>min()</td>
<td>minimum</td>
</tr>
<tr>
<td>range()</td>
<td>max()-min()</td>
</tr>
<tr>
<td>std()</td>
<td>standard deviation</td>
</tr>
<tr>
<td>var()</td>
<td>variance</td>
</tr>
<tr>
<td>cov()</td>
<td>covariance matrix</td>
</tr>
<tr>
<td>corr()</td>
<td>correlation matrix</td>
</tr>
<tr>
<td>cov2corr()</td>
<td>convert cov to corr</td>
</tr>
<tr>
<td>corr2cov()</td>
<td></td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scatter(y,x)</code></td>
<td>scatter plot</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
</tbody>
</table>
### Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scatter(y,x)</code></td>
<td>scatter plot</td>
</tr>
<tr>
<td><code>prctile(x,[10,20..])</code></td>
<td>percentiles</td>
</tr>
<tr>
<td><code>skewness()</code></td>
<td>skewness</td>
</tr>
<tr>
<td><code>kurtosis()</code></td>
<td>kurtosis</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
<tr>
<td>kurtosis()</td>
<td>kurtosis</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scatter(y,x)</code></td>
<td>scatter plot</td>
</tr>
<tr>
<td><code>prctile(x,[10,20..])</code></td>
<td>percentiles</td>
</tr>
<tr>
<td><code>skewness()</code></td>
<td>skewness</td>
</tr>
<tr>
<td><code>kurtosis()</code></td>
<td>kurtosis</td>
</tr>
<tr>
<td><code>hist()</code></td>
<td>histogram</td>
</tr>
<tr>
<td><code>histfit()</code></td>
<td>hist with normal</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
<tr>
<td>kurtosis()</td>
<td>kurtosis</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram</td>
</tr>
<tr>
<td>histfit()</td>
<td>hist with normal</td>
</tr>
<tr>
<td>cdfplot()</td>
<td>empirical cdf</td>
</tr>
<tr>
<td>Function</td>
<td>Statistic</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
<tr>
<td>kurtosis()</td>
<td>kurtosis</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram</td>
</tr>
<tr>
<td>histfit()</td>
<td>hist with normal</td>
</tr>
<tr>
<td>cdfplot()</td>
<td>empirical cdf</td>
</tr>
<tr>
<td>trimmean(x,pct)</td>
<td>trimmed mean</td>
</tr>
</tbody>
</table>
### Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
<tr>
<td>kurtosis()</td>
<td>kurtosis</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram</td>
</tr>
<tr>
<td>histfit()</td>
<td>hist with normal</td>
</tr>
<tr>
<td>cdfplot()</td>
<td>empirical cdf</td>
</tr>
<tr>
<td>trimmean(x,pct)</td>
<td>trimmed mean</td>
</tr>
<tr>
<td>IQR()</td>
<td>Interquartile (75-25) range</td>
</tr>
</tbody>
</table>
### Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>scatter(y,x)</td>
<td>scatter plot</td>
</tr>
<tr>
<td>prctile(x,[10,20..])</td>
<td>percentiles</td>
</tr>
<tr>
<td>skewness()</td>
<td>skewness</td>
</tr>
<tr>
<td>kurtosis()</td>
<td>kurtosis</td>
</tr>
<tr>
<td>hist()</td>
<td>histogram</td>
</tr>
<tr>
<td>histfit()</td>
<td>hist with normal</td>
</tr>
<tr>
<td>cdfplot()</td>
<td>empirical cdf</td>
</tr>
<tr>
<td>trimmean(x,pct)</td>
<td>trimmed mean</td>
</tr>
<tr>
<td>IQR()</td>
<td>Interquartile (75-25) range</td>
</tr>
<tr>
<td>geommean()</td>
<td>geometric mean (stats)</td>
</tr>
</tbody>
</table>
# Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scatter(y,x)</code></td>
<td>scatter plot</td>
</tr>
<tr>
<td><code>prctile(x,[10,20..])</code></td>
<td>percentiles</td>
</tr>
<tr>
<td><code>skewness()</code></td>
<td>skewness</td>
</tr>
<tr>
<td><code>kurtosis()</code></td>
<td>kurtosis</td>
</tr>
<tr>
<td><code>hist()</code></td>
<td>histogram</td>
</tr>
<tr>
<td><code>histfit()</code></td>
<td>hist with normal</td>
</tr>
<tr>
<td><code>cdfplot()</code></td>
<td>empirical cdf</td>
</tr>
<tr>
<td><code>trimmean(x,pct)</code></td>
<td>trimmed mean</td>
</tr>
<tr>
<td><code>IQR()</code></td>
<td>Interquartile (75-25) range</td>
</tr>
<tr>
<td><code>geommean()</code></td>
<td>geometric mean (stats)</td>
</tr>
<tr>
<td><code>harmmean()</code></td>
<td>harmonic mean (stats)</td>
</tr>
</tbody>
</table>
## Description and Exploration II

<table>
<thead>
<tr>
<th>Function</th>
<th>Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scatter(y,x)</code></td>
<td>scatter plot</td>
</tr>
<tr>
<td><code>prctile(x, [10, 20..])</code></td>
<td>percentiles</td>
</tr>
<tr>
<td><code>skewness()</code></td>
<td>skewness</td>
</tr>
<tr>
<td><code>kurtosis()</code></td>
<td>kurtosis</td>
</tr>
<tr>
<td><code>hist()</code></td>
<td>histogram</td>
</tr>
<tr>
<td><code>histfit()</code></td>
<td>hist with normal</td>
</tr>
<tr>
<td><code>cdfplot()</code></td>
<td>empirical cdf</td>
</tr>
<tr>
<td><code>trimmean(x, pct)</code></td>
<td>trimmed mean</td>
</tr>
<tr>
<td><code>IQR()</code></td>
<td>Interquartile (75-25) range</td>
</tr>
<tr>
<td><code>geommean()</code></td>
<td>geometric mean (stats)</td>
</tr>
<tr>
<td><code>harmmean()</code></td>
<td>harmonic mean (stats)</td>
</tr>
<tr>
<td><code>moment(x, order)</code></td>
<td>central moments (stats)</td>
</tr>
</tbody>
</table>
‘\( P = \text{anova1}(X, \text{GROUP}) \)’ returns the p-value for the null hypothesis that the means of the groups are equal.
‘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**

- ‘\( P = \text{anova1}(X, \text{GROUP}) \)’ 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 ‘\( Xc=\text{nominal}(X) \)’
- ‘\([P, \text{ANOVATAB}] = \text{anova1}(\ldots)\)’ 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,ANOVATAK] = anova1(…)' returns the ANOVA table values as the cell array ANOVATAK.
- ‘[P,ANOVATAK,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.
Statistical Computing in Matlab

Statistics Utilities

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 occupy each cell (must be constant). X must be $P \times REPSxK$ where $P$ is the number of categories in group 1 and $K$ is the number in group 2. Cell entries should be Y values.
Anova2: Two Way Anova

- ‘$P = \text{anova2}(X, \text{REPS})$’ 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 occupy each cell (must be constant). $X$ must be $P \times \text{REPS} \times K$ where $P$ is the number of categories in group 1 and $K$ is the number in group 2. Cell entries should be $Y$ values.
- ‘$P=\text{ANOVAN}(Y, \text{GROUP},'PARAM1',\text{val1},'PARAM2',\text{val2},...)$’
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 occupy each cell (must be constant). X must be \( P \times REPS \times K \) where \( P \) is the number of categories in group 1 and \( K \) is the number in group 2. Cell entries should be Y values.
- ‘P=ANOVAN(Y,GROUP,’PARAM1’,val1,’PARAM2’,val2,...)’
- Let’s try it ‘randn(‘seed’,1)’ ‘XA2=randn(1000,100)’ then run a two-way ANOVA with REPS = 10.
‘P = anovan(X,GROUP)’ returns the p-values for the null hypotheses of no group effects in a multi-way anova.
**Anovan: N Way Anova**

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

‘X’ is a vector of outcome values

‘GROUP’ is awkward. 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 multi-way anova.
- ‘X’ is a vector of outcome values
- ‘GROUP’ is awkward. 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...
‘B=regrss(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 coefficients and 95% Confidence Intervals

Generate ‘X = randn(100,1);’ and ‘Y = randn(100,1) + X;’ now regress Y on X and see what you get
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 coefficients 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.
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 coefficients 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 coefficients 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.
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 coefficients 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
Practitioners often want additional information. The ‘regstats’ function provides this.
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:
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:
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>'covb'</td>
<td>Covariance of regression coefficients</td>
</tr>
<tr>
<td>'yhat'</td>
<td>Fitted values of the response data</td>
</tr>
<tr>
<td>'adjrsquare'</td>
<td>Adjusted R-square statistic</td>
</tr>
<tr>
<td>'dfbetas'</td>
<td>Scaled change in regression coefficients</td>
</tr>
<tr>
<td>'cookd'</td>
<td>Cook’s distance</td>
</tr>
<tr>
<td>'tstat'</td>
<td>t statistics for coefficients</td>
</tr>
</tbody>
</table>
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
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'.
- `[B,DEV,STATS] = GLMFIT(...)` returns many useful quantities including the covariance matrix.
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’)
‘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
First we need to generate some data

```matlab
x = randn(1000,1);
y = (x+randn(1000,1)<0);
```

This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on x.

Now we estimate the model:

```
[B,DEV,STATS] = GLMFIT(x,y,'binomial','link','probit')
```

Did we do well?

Now use

```
YHAT = glmval(B,X,LINK)
```

to produce predicted means.
First we need to generate some data

'x = randn(1000,1);' and 'y = (x+randn(1000,1)<0);'
First we need to generate some data

\[
x = \text{randn}(1000,1);
\]

\[
y = (x + \text{randn}(1000,1) \leq 0);
\]

This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on \(x\)
First we need to generate some data:

1. `x = randn(1000,1);` and `y = (x+randn(1000,1)>0);`

This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on \( x \).

Now we estimate the model:

4. `[B,DEV,STATS] = GLMFIT(x,y,'binomial','link','probit')`

Did we do well?

Now use `YHAT = glmval(B,X,LINK)` to produce predicted means.
First we need to generate some data

'\texttt{x = randn(1000,1);}’ and ‘\texttt{y = (x+randn(1000,1)<0);}’

This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on $x$

Now we estimate the model ‘\texttt{[B,DEV,STATS] = GLMFIT(x,y,’binomial’,’link’,’probit’)}’

Did we do well?
First we need to generate some data

‘\( x = \text{randn}(1000,1); \)’ and ‘\( y = (x + \text{randn}(1000,1)) \mid 0); \)’

This DGP leads to probit as the correct specification with an intercept of 0 and a coefficient of one on \( x \)

Now we estimate the model ‘\([B, \text{DEV}, \text{STATS}] = \text{GLMFIT}(x, y, \text{‘binomial’}, \text{‘link’}, \text{‘probit’})\)’

Did we do well?

Now use ‘\( \text{YHAT} = \text{glmval}(B, X, \text{LINK}) \)’ to produce predicted means
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'
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**

Did we do well?

Now compare se's with and without dispersion 'STATS.se'
Poisson Regression in Matlab

1. Generate Data Again

   \[ x = \text{randn}(1000,1); \] then \[ u = \text{randn}(1000,1); \] and \[ \text{mean} = \exp(2-x+u); \] lastly \[ y = \text{poissrnd}(	ext{mean}); \]

2. This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on \( x \)

3. Now we estimate the model \([B,DEV,STATS] = \text{GLMFIT}(x,y,'poisson','estdisp','on')\)
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?
Generate Data Again

1. \(x = \text{randn}(1000,1);\) then \(u = \text{randn}(1000,1);\) and \(\text{mean} = \exp(2-x+u);\) lastly \(y = \text{poissrnd}(	ext{mean});\)

This DGP leads to the Poisson with overdispersion as the correct specification with an intercept of 0 and a coefficient of one on \(x\)

Now we estimate the model \([B,DEV,STATS] = \text{GLMFIT}(x,y,'\text{poisson'},'\text{estdisp'},'\text{on}')\)

Did we do well?

Now compare se’s with and without dispersion \('\text{STATS.se}'\)