# Classes 2-3: "Entering the Matrix Laboratory & Optimization"

Damian Clarke

January 31, 2020

Research Methods II
MRes. in Economics

# Today's Plan

The command line

Functions

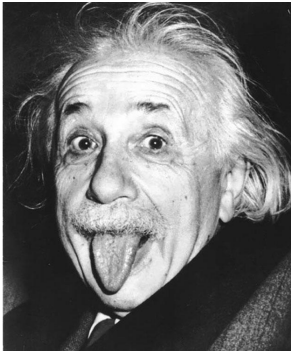
Optimisation

Simulating model solutions

# Our approach...

"It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements **as simple and as few as possible** without having to surrender the **adequate representation** of a single datum of experience."

**Einstein**



# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

```
>> 1 + 1
ans =
     2
```

```
>> x = [1, 2; 3, 4; 5, 6]
x =
    1   2
    3   4
    5   6

>> 2*x
ans =
     2   4
     6   8
    10  12
```

```
>> who
>> whos
```

```
. sysuse auto
(1978 Automobile Data)
. reg mpg price weight, noheader
. outsheet mpg price weight using auto.csv, ///
    nonames comma
```

```
. sysuse auto
(1978 Automobile Data)

. reg mpg price weight, noheader
```

| mpg | Coef. | Std. Err. | t | P>\|t\| | [95% Conf. | Interval] |
|---|---|---|---|---|---|---|
| price | -.0000935 | .0001627 | -0.57 | 0.567 | -.000418 | .0002309 |
| weight | -.0058175 | .0006175 | -9.42 | 0.000 | -.0070489 | -.0045862 |
| _cons | 39.43966 | 1.621563 | 24.32 | 0.000 | 36.20635 | 42.67296 |

```
.
```

```
>> DataIn   = dlmread('auto.csv');
>> X        = DataIn(:, 2:3);
>> size(X)
ans =
    74     2

>> X        = [X, ones(74,1)];
>> y        = DataIn(:,1);
```

```
>> XX=X'*X;
>> Xy=X'*y;
>> beta=inv(XX)*Xy
```

We have used MATLAB to recreate Stata's **point estimates** in a regression function. Can you now generate the same **standard errors**?

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

```
function y = doubleit(x)

  y = 2 * x;

return
```

**Challenge:** Write a function that (i) accepts $\boldsymbol{y}$ and $\boldsymbol{X}$, and (ii) returns $\hat{\boldsymbol{\beta}}$ and its standard error.

**Challenge:** Write a function to calculate:

$$u(x_1, x_2) = x_1^{1/2} \cdot x_2^{1/2}$$

Suppose that, for some reason, we want to find utility for $x_1 = 5$ and $x_2 \in \{1, \ldots, 10\}$...

```
x1 = [1:10]';
x2 = 5;
```

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Log Likelihood Function for the Linear Model with Normal Errors

Suppose that we have:

$$y_i = \boldsymbol{\beta} \cdot \boldsymbol{x}_i + \varepsilon_i$$
$$\varepsilon_i \,|\, \boldsymbol{x}_i \sim \mathcal{N}(0, \sigma^2)$$

$$L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y} \,|\, \boldsymbol{x}) = -\left(\frac{N}{2}\right) \ln 2\pi - \left(\frac{N}{2}\right) \ln \sigma^2 - \left(\frac{1}{2\sigma^2}\right) (\boldsymbol{y} - \boldsymbol{\beta x})'(\boldsymbol{y} - \boldsymbol{\beta x})$$

**Write a function** to calculate $L(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y} \,|\, \boldsymbol{x})$.

```
help fmincon
```

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}$$

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$\mathbf{C} \cdot \mathbf{x} \leq \mathbf{d}$$

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$g(\mathbf{x}) = 0$$

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$h(\mathbf{x}) \leq 0$$

$$\min_{x \in \mathbb{R}^k} f(\mathbf{x})$$

subject to

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}$$
$$\mathbf{A} \cdot \mathbf{x} = b$$
$$\mathbf{C} \cdot \mathbf{x} \leq d$$
$$g(\mathbf{x}) = 0$$
$$h(\mathbf{x}) \leq 0$$

# Adding More Arguments in fmincon...

```
>> lb  = [-1000, -1000, -1000, 0];
>> ub  = [1000, 1000, 1000, 100];
>> theta0  = [0, 0, 0, 1];
>> opt = optimset('TolFun',1E-20,'TolX',1E-20,'MaxFunEvals',1000);

>> fmincon(@(theta)normalML(theta,y,X), theta0, [], ...
                            [], [], [], lb, ub, [], opt);
```

# Adding More Arguments in fmincon...

```
>> opt = optimset('TolFun',1E-20,'TolX',1E-20,'MaxFunEvals',1000, ...
                  'PlotFcns','optimplotfval');
>> fmincon(@(theta)normalML(theta,y,X), theta0, [], ...
                           [], [], [], lb, ub, [], opt);
```

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Outline

1. The command line
2. Functions
3. Optimisation
4. Simulating model solutions

# Loops and Conditional Statements in MATLAB

There are a number of general programming constructs which may be of use for us in MATLAB. Loops let you repeat tasks for a predetermined series of values.

1. **For Loops**: Repeat a task for each value listed. Can take a number of forms:

```
for i=1:10
    2*i
end
```

```
for i=2:2:10
    2*i
end
```

```
for i=[1,3,5,9]
    2*i
end
```

- Here `for` and `end` are obligatory, the counter variable (`i` here) can be called anything
- Remember that MATLAB is built for vectors/matrices, so where possible, prefer vectorized code rather than loops
- MATLAB has an amazingly simple `parfor` loop which is exactly the same, but can speed up work considerably where tasks are "embarrassingly parallel"

# Loops and Conditional Statements in MATLAB

There are a number of general programming constructs which may be of use for us in MATLAB. Loops let you repeat tasks for a predetermined series of values.

**2. While Loops**: Repeat a task as long as some condition is still met.

```
i = 0
while i<10
    i = i+1
end
```

- Here `while` and `end` are obligatory, the counter variable (`i` here) can be called anything
- Must be careful that logic makes sense and avoid infinite loops. If caught in a loop, we can force a break holding down ctrl+c.

# Loops and Conditional Statements in MATLAB

There are a number of general programming constructs which may be of use for us in MATLAB. Conditional statements can be handled using if/else.

**3. if, elseif, else**: Differentially execute statements depending on conditions. For example:

```
t  = 1.96;
df = 1000;
if t>0
    p = 2*(1-tcdf(t,df))
elseif t<=0
    p = 2*tcdf(t,df)
else
    p = NaN
end
```

# Activity 1: Simulating Consumption

$$max_{x_1,x_2} x_1^{1/2} \cdot x_2^{1/2} \qquad \text{subject to} \qquad I = p_1 x_1 + p_2 x_2$$

$$p_1 \sim U(100, 150)$$

# Activity 2: Bootstrap!

**Challenge**: Replicate the following in MATLAB:

```
bootstrap, reps(10000): reg mpg price weight, noheader
```

**Hint:**

```
>> s = [101, 102, 103, 104, 105]
s =
   101   102   103   104   105

>> s([1, 1, 1, 4])
ans =
   101   101   101   104
```