PSY8219 : Week 3

Homework 2 Due Today

Homework 3 Due September 19

Readings for Today
   Attaway Chapters 3, 4, and 5

Readings for Next Week
   Attaway Chapter 1
NOTE: the homework assignment due next week has a couple of tricky bits (the class slides will give some clues)
please don't "look ahead" on the Matlab files I distributed before class ...
Mathematical Operations on Arrays

Basic Linear Algebra
Mathematical Operations on Arrays

```matlab
>> clear all
>> data = [1 2 ; 3 4]

add a number to a particular element in an array

>> data(1,2) = data(1,2) + 10
```
**Mathematical Operations on Arrays**

```matlab
>> clear all
>> data = [1 2 ; 3 4]
```

Add or subtract a number to/from EVERY element:
```
>> data = data + 10
>> data = data - 10
```
Mathematical Operations on Arrays

```matlab
>> clear all

>> data = [1 2; 3 4]

multiple or divide by the same number for every element in the array

>> data = data * 10

>> data = data / 10
```
Mathematical Operations on Arrays

>> clear all
>> data = [1 2 ; 3 4]

>> x = repmat(10, 2, 2)
>> x = repmat(10, size(data))

what does repmat do?
Mathematical Operations on Arrays

```matlab
>> clear all
>> data = [1 2 ; 3 4]

>> x = repmat(10, 2, 2)
>> x = repmat(10, size(data))

>> data + 10
>> data + x
```
Mathematical Operations on Arrays

```
>> clear all

>> data = [1 2 ; 3 4]
```

When you add or subtract two arrays, the corresponding elements are added or subtracted.

```
>> data2 = [8 7 ; 6 5]

>> data + data2

>> data - data2
```
Mathematical Operations on Arrays

```matlab
>> clear all
>> data = [1 2 ; 3 4]

What about these?
>> x = repmat(10, 2, 2)
>> data * 10
>> data * x
>> data / 10
>> data / x
```
Arrays vs. Matrices

Arrays are data structures with rows and columns used to organize and use data.

Matrices are mathematical entities used in linear algebra.

Unfortunately, in Matlab, arrays and matrices are defined in exactly the same way. While they are distinct computationally and mathematically, Matlab treats them as the very same type. Watch out.
Mathematical Operations on Arrays

```matlab
>> clear all
>> data = [1 2 ; 3 4]

What about these?
```

```matlab
>> x = repmat(10, 2, 2)
>> data * 10
>> data .* x
>> data / 10
>> data ./ x
```

Element-by-element
Array vs. Matrix multiplication and division

**Best Practices**

Get in the habit, within Matlab, of always using .* or ./ when you multiply or divide, even if you do not need it (unless you really intend matrix operations)
Some more operations on arrays

```plaintext
>> clear all
>> data = [1 2 ; 3 4]
```

**What does this do?**

```plaintext
>> data .^ 2
```

**Versus this?**

```plaintext
>> data ^ 2
```

**Best Practices**

Also always use .^
Some more operations on arrays

```plaintext
>> log10(data)
>> exp(data)
>> sin(data)
```
Some more operations on arrays

```
>> mean(data)
What’s it doing?
How could you find out?
>> help mean
>> doc mean
```
Some more operations on arrays

>>> mean(data)

What’s it doing?
How could you find out?

>>> help mean

>>> doc mean

>>> mean(data,1)

>>> mean(data,2)
Evaluating Mathematical Formulas in Matlab

One thing that Matlab does really well is calculate (and plot) lots of values for some function.
Evaluating Mathematical Formulas in Matlab

One thing that Matlab does really well is calculate (and plot) lots of values for some function

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \]

What is this?
Evaluating Mathematical Formulas in Matlab

One thing that Matlab does really well is calculate (and plot) lots of values for some function

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \]

What is this?

probability density function for a normal distribution
Evaluating Mathematical Formulas in MatLab

How would we code this in MatLab?

\[ f(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp \left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] \]
Evaluating Mathematical Formulas in Matlab

How would we code this in Matlab?

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] \]

sig2 = 1  % variance
mu = 0     % mean
x = 1      % a particular value
f = (1/sqrt(2*pi*sig2)) * ... 
    exp(-((x-mu)^2)/(2*sig2))
Evaluating Mathematical Formulas in Matlab

How would we code this in Matlab?

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] \]

\[ \text{sig2} = 1 \quad \% \text{ variance} \]
\[ \text{mu} = 0 \quad \% \text{ mean} \]
\[ x = 1 \quad \% \text{ a particular value} \]
\[ f = \left( \frac{1}{\sqrt{2\pi\text{sig2}}} \right) \times \ldots \exp\left( -\frac{((x-\mu)^2)}{(2\times\text{sig2})} \right) \]

This calculates only one value at a time.
Evaluating Mathematical Formulas in Matlab

How would we code this in Matlab?

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] \]

```matlab
sig2 = 1 % variance
mu = 0 % mean
x = -4:.01:4 % a particular value
f = (1/sqrt(2*pi*sig2)) * ... 
    exp(-((x-mu)^2)/(2*sig2))
plot(x,f)
```

What is this doing?
Evaluating Mathematical Formulas in Matlab

How would we code this in Matlab?

\[ f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] \]

sig2 = 1  \hspace{1cm} \text{% variance}
mu = 0  \hspace{1cm} \text{% mean}
x = -4:.01:4  \hspace{1cm} \text{% a particular value}
f = (1/sqrt(2*pi*sig2)) * ... 
    exp(-((x-mu)^2)/(2*sig2))
plot(x,f)

What's wrong with this?
Evaluating Mathematical Formulas in Matlab

How would we code this in Matlab?

\[
f(x) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left[ -\frac{(x-\mu)^2}{2\sigma^2} \right]
\]

\[
sig2 = 1 \quad \% \ \text{variance}
\]

\[
mu = 0 \quad \% \ \text{mean}
\]

\[
x = -4:.01:4 \quad \% \ \text{a particular value}
\]

\[
f = (1/sqrt(2*pi*sig2)) .* ...
    \exp(-((x-mu).^2)/(2*sig2))
\]

plot(x,f)
Vectors, Matrices, and Linear Algebra
Vectors, Matrices, and Linear Algebra

Arrays are merely containers that hold numeric data in an organized way.

Vectors (1-dimensional) and Matrix (2-dimensional) are mathematical entities with mathematical operators that act on them.

In Matlab, a 1-dimensional vector and 1-dimensional array, and a 2-dimensional matrix and a 2-dimensional array are defined exactly the same way, but they need to be thought of differently.
Vectors
Vectors

Vectors have a magnitude and direction.

Terminology can be a bit confusing in that a vector is one-dimensional in one sense

\[ \begin{align*}
  a &= [1 \ 2] \\
  b &= [3 \ 2 \ 1] \\
  c &= [1 \ 3 \ 3 \ 1]
\end{align*} \]
Vectors

Vectors have a magnitude and direction.

But we’ll also illustrate plotting a vector in a multidimensional space

```plaintext
>> a = [1 2]         % vector 2D space
>> b = [3 2 1]       % vector 3D space
>> c = [1 3 3 1]     % vector 4D space
```
Vectors

>>> a = [1 2] % vector 2D space
>>> b = [3 2]
>>> a + b
Vectors

```python
>> a + b
```
Vectors

\[ \text{a} + \text{b} \]
Vectors

```python
>>> a = [1, 2]
>>> b = 2*a
```
Vectors

```
>> a = [1 2]
>> b = 2*a
```
Vectors

```python
>>> a = [1, 2]
>>> b = 2*a
```
Vectors

\[ \text{norm}(b) \quad \% \text{ vector length (norm)} \]

Euclidian norm is found using the Pythagorean Theorem (also called 2-norm or an \( L_2 \) norm)
Vectors

angle between two vectors ...

\[
\begin{align*}
\gg a &= [1 \ 3] \\
\gg b &= [2 \ 1]
\end{align*}
\]
Vectors

angle between two vectors ... Implement in Matlab?

\[ \cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|} \]

“dot product”

norm
Vectors

\[ \theta = \text{radtodeg}(\cos( ... \\
\text{dot}(a,b) / (\text{norm}(a)\times\text{norm}(b)))) \]

\[
\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i
\]

\[
\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}
\]
Matrices
Solving systems of linear algebra

\[
\begin{align*}
x + 2y &= 2 \\
x + y &= 3
\end{align*}
\]

How would you view these equations in Matlab?
Solving systems of linear algebra

\[
\begin{align*}
  x + 2y &= 2 \\
  x + y &= 3
\end{align*}
\]

How would you view these equations in Matlab?

```matlab
x1 = -5:5;
x2 = -5:5;
y1 = -0.5*x1 + 1
y2 = -x2 + 3
plot(x1,y1,x2,y2)
```

What’s the solution to the system of equations?
Solving systems of linear algebra

\[
x + 2y = 2 \\
x + y = 3
\]

Rewrite in matrix notation:

\[
\begin{bmatrix}
1 & 2 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} =
\begin{bmatrix}
2 \\
3
\end{bmatrix}
\]
Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} & \ldots & a_{1n} \\
  a_{21} & a_{22} & \ldots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \ldots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
  b_{11} & b_{12} & \ldots & b_{1p} \\
  b_{21} & b_{22} & \ldots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \ldots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
  c_{11} & c_{12} & \ldots & c_{1p} \\
  c_{21} & c_{22} & \ldots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \ldots & c_{mp}
\end{bmatrix}
\]

mxn matrix \times \text{nxp matrix} = \text{mxp matrix}
Multiplying Matrices

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1p} \\
b_{21} & b_{22} & \cdots & b_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1p} \\
c_{21} & c_{22} & \cdots & c_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

mxn matrix \times \text{nxp matrix} = \text{mxp matrix}

\[c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}\]
Multiplying Matrices

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
    b_{11} & b_{12} & \cdots & b_{1p} \\
    b_{21} & b_{22} & \cdots & b_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
    c_{11} & c_{12} & \cdots & c_{1p} \\
    c_{21} & c_{22} & \cdots & c_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

mxn matrix \quad nxp matrix \quad mxp matrix

\[c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}\]
Multiplying Matrices

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1p} \\
b_{21} & b_{22} & \cdots & b_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1p} \\
c_{21} & c_{22} & \cdots & c_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

\[c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}\]
Multiplying Matrices

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
    b_{11} & b_{12} & \cdots & b_{1p} \\
    b_{21} & b_{22} & \cdots & b_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= \begin{bmatrix}
    c_{11} & c_{12} & \cdots & c_{1p} \\
    c_{21} & c_{22} & \cdots & c_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    c_{m1} & c_{m2} & \cdots & c_{np}
\end{bmatrix}
\]

mxn matrix  \times  nxp matrix  =  mxp matrix

\[c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}\]
Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1p} \\
  b_{21} & b_{22} & \cdots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= \begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1p} \\
  c_{21} & c_{22} & \cdots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

mxn matrix \times nxp matrix = mxp matrix

\[c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}\]
### Multiplying Matrices

Let's consider multiplying two matrices:

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1p} \\
  b_{21} & b_{22} & \cdots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1p} \\
  c_{21} & c_{22} & \cdots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

The elements of the resulting matrix are calculated as:

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
Multiplying Matrices

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
    b_{11} & b_{12} & \cdots & b_{1p} \\
    b_{21} & b_{22} & \cdots & b_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= \begin{bmatrix}
    c_{11} & c_{12} & \cdots & c_{1p} \\
    c_{21} & c_{22} & \cdots & c_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

\[c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}\]
Multiplying Matrices

\[ \begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix} \times \begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1p} \\
  b_{21} & b_{22} & \cdots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix} = \begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1p} \\
  c_{21} & c_{22} & \cdots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \cdots & c_{np}
\end{bmatrix} \]

\[ c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \]
Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1p} \\
  b_{21} & b_{22} & \cdots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= \begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1p} \\
  c_{21} & c_{22} & \cdots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \cdots & c_{np}
\end{bmatrix}
\]

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
Multiplying Matrices

\[
\begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1p} \\
b_{21} & b_{22} & \cdots & b_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= \begin{bmatrix}
c_{11} & c_{12} & \cdots & c_{1p} \\
c_{21} & c_{22} & \cdots & c_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
Multiplying Matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1p} \\
  b_{21} & b_{22} & \cdots & b_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1p} \\
  c_{21} & c_{22} & \cdots & c_{2p} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
Multiplying Matrices

\[
\begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}
\times
\begin{bmatrix}
    b_{11} & b_{12} & \cdots & b_{1p} \\
    b_{21} & b_{22} & \cdots & b_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    b_{n1} & b_{n2} & \cdots & b_{np}
\end{bmatrix}
= 
\begin{bmatrix}
    c_{11} & c_{12} & \cdots & c_{1p} \\
    c_{21} & c_{22} & \cdots & c_{2p} \\
    \vdots & \vdots & \ddots & \vdots \\
    c_{m1} & c_{m2} & \cdots & c_{mp}
\end{bmatrix}
\]

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
Solving systems of linear algebra

\[
\begin{align*}
  x + 2y & = 2 \\
  x + y & = 3
\end{align*}
\]

Rewrite in matrix notation:

\[
\begin{bmatrix}
  1 & 2 \\
  1 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
=
\begin{bmatrix}
  2 \\
  3
\end{bmatrix}
\]
Solving systems of linear algebra

\[
x + 2y = 2
\]
\[
x + y = 3
\]

Rewrite in matrix notation:

\[
\begin{bmatrix}
1 & 2 \\
1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
\end{bmatrix} =
\begin{bmatrix}
2 \\
3 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
x + 2y \\
x + y \\
\end{bmatrix} =
\begin{bmatrix}
2 \\
3 \\
\end{bmatrix}
\]
Solving systems of linear algebra

\[ \begin{align*}
  x_1 + 2x_2 &= 2 \\
  x_1 + x_2 &= 3
\end{align*} \]

Rewrite:

\[
\begin{bmatrix}
  1 & 2 \\
  1 & 1
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}
= 
\begin{bmatrix}
  2 \\
  3
\end{bmatrix}
\]

\[
A \ x = b
\]

*matrix* \hspace{1cm} \textit{vector} \hspace{1cm} \textit{vector}
Mathematical Operations on Matrices

\[
A + B = B + A
\]
\[
c(A+B) = cA + cB
\]

Convince yourself in Matlab

```
>> A = [2 1 ; 2 3]
>> B = [1 2 ; 0 1]
>> c = 2
>> (A + B) - (B + A)
>> c.*(A+B) - (c.*A + c.*B)
```
Mathematical Operations on Matrices

How about this? Try it in Matlab.

\[ AB = BA \]
Mathematical Operations on Matrices

How about this? Try it in Matlab.

\[ AB \neq BA \]

Matrix multiplication
\[ \gg A*B - B*A \]

vs.
Array multiplication
\[ \gg A.*B - B.*A \]
Mathematical Operations on Matrices

How about this? Try it in Matlab.

\[ \mathbf{AB} \neq \mathbf{BA} \]

Matrix multiplication
\[
\text{>> } \mathbf{A*B} - \mathbf{B*A}
\]

vs.

Array multiplication
\[
\text{>> } \mathbf{A.*B} - \mathbf{B.*A}
\]

Imagine \( \mathbf{A} \) is 2x3 and \( \mathbf{B} \) is 3x4.

Then \( \mathbf{AB} \) is defined but \( \mathbf{BA} \) isn’t.
What do we need to do to solve for \( x \)?
If we can move $A$ to the other side, we will have a solution for $x$. 

$$\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$
Mathematical Operations on Matrices

Ax = b

how would you solve this if A, x, and b were all scalar values?
Mathematical Operations on Matrices

Ax = b

imagine we have some matrix W such that
WAx = Wb

x = Wb
Mathematical Operations on Matrices

\[ Ax = b \]

imagine we have some matrix \( W \) such that

\[ WAx = Wb \]

\[ x = Wb \]
Mathematical Operations on Matrices

W is the inverse of matrix A, or $A^{-1}$

$W Ax = Wb$

$A^{-1} Ax = A^{-1} b$

$x = A^{-1} b$
Mathematical Operations on Matrices

$A^{-1}$ is not the same as a matrix of the inverse of the elements ...

Show they are not the same in Homework 3.
Mathematical Operations on Matrices

If $\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{x}$,

what does $\mathbf{A}^{-1}\mathbf{A}$ need to equal?

what shape is it?

what values does it have within it?
Back to this

\[
\begin{bmatrix}
1 & 2 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= \begin{bmatrix}
2 \\
3
\end{bmatrix}
\]

\[
A \ x = b
\]

\[
A^{-1} \ A \ x = A^{-1} \ b
\]

\[
I \ x = A^{-1} \ b
\]
Back to this

$$\text{>> } x = \text{inv}(A) \times b$$
Back to this

```matlab
>> x = inv(A) * b
```

This also allowed in Matlab. It uses a different algorithm.

```matlab
>> x = A \ b
```

“Matrix left division”
Larger systems of linear equations
(mathematically the same matrix operation)

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{bmatrix}
= 
\begin{bmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{bmatrix}
\]

solving for \( x \)
Linear Regression

\[
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n \\
y
\end{bmatrix} =
\begin{bmatrix}
1 & x_{11} & \cdots & x_{1k} \\
1 & x_{12} & \cdots & x_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_{n2} & \cdots & x_{nk}
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_k
\end{bmatrix} +
\begin{bmatrix}
\varepsilon_1 \\
\varepsilon_2 \\
\vdots \\
\varepsilon_n
\end{bmatrix}
\]

solving for \( \mathbf{b} \)
Simple Regression

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n \\
\end{bmatrix}
= 
\begin{bmatrix}
  1 & x_{11} \\
  1 & x_{12} \\
  \vdots \\
  1 & x_{1n} \\
\end{bmatrix}
\begin{bmatrix}
  b_0 \\
  b_1 \\
  \vdots \\
  \varepsilon_n \\
\end{bmatrix}
+ 
\begin{bmatrix}
  \varepsilon_1 \\
  \varepsilon_2 \\
  \vdots \\
  \varepsilon_n \\
\end{bmatrix}
\]

\[y = Xb\]

solving for \(b\)
Simple Regression

\[
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} =
\begin{bmatrix}
1 & x_{11} \\
1 & x_{12} \\
\vdots & \\
1 & x_{1n}
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1
\end{bmatrix}
+ \begin{bmatrix}
\varepsilon_1 \\
\varepsilon_2 \\
\vdots \\
\varepsilon_n
\end{bmatrix}
\]

\( y = Xb \)

can we do the same trick as before?
Simple MATLAB example

```matlab
>> X = [1 60 ; 1 61; 1 62; 1 63; 1 65]
>> y = [3.1 ; 3.6 ; 3.8 ; 4 ; 4.1]
```
Simple Regression

\[ y = X \ b \]

Can we just do this?

\[ X^{-1} \ y = X^{-1} \ X \ b \]

Try is.
Simple Regression

\[ y = X b \]

Can we just do this?

\[ X^{-1} y = X^{-1} X b \]

Try is.

No. Matrix must be square.
Simple Regression

\[ y = X b \]

We can do this:

\[ X^T y = X^T X b \]

What will \( X^T X \) give you? What size is it?
Simple Regression

\[ y = X \, b \]

We can do this:

\[ X^T \, y = X^T \, X \, b \]

What could we do now?
Simple Regression

\[ y = X \cdot b \]

We can do this:

\[ X^T \cdot y = X^T \cdot X \cdot b \]

What could we do now?

\[ (X^T \cdot X)^{-1} \cdot X^T \cdot y = (X^T \cdot X)^{-1} \cdot X^T \cdot X \cdot b \]

What will this give you?
Simple Regression

\[ y = X \ b \]

We can do this:

\[ X^T \ y = X^T \ X \ b \]

What could we do now?

\[ (X^T \ X)^{-1} \ X^T \ y = (X^T \ X)^{-1} \ X^T \ X \ b \]

What will this give you?

\[ (X^T \ X)^{-1} \ X^T \ y = b \]
Simple MATLAB example

```matlab
>> X = [1 60 ; 1 61; 1 62; 1 63; 1 65]
>> y = [3.1 ; 3.6 ; 3.8 ; 4 ; 4.1]
>> b = inv(X'*X) * X' * y

>> plot(X(:,2), y, ...
       X(:,2), b(1)+b(2)*X(:,2))
```
Cell Arrays

Cell Arrays

Unlike arrays, the elements of cell arrays can all be different types.

```matlab
>> clear all
>> a = {2 'house' true ; ...
       'dog' pi [2 3 ; 7 8]}
```

What does this look like?
Using Cell Arrays

Referencing a Cell Array

```matlab
>> b = a{1,3}
>> whos b

>> c = a(1,3)
>> whos c
```
Using Cell Arrays

Changing a value

```
>> a{1,3} = 'bob'
```

But what about this?

```
>> a(3,3) = 23
```
Using Cell Arrays

Add new elements

\[ a\{3,3\} = 23 \]

But what about this?

\[ a(4,4) = 23 \]
Using Cell Arrays

Cell arrays can contain cell arrays

```matlab
>> x = 6
>> a{3,1} = {x 'house'; 23 false}
```

Reference cell arrays within cell arrays

```matlab
>> a{3,1}{1,2}
```
Using Cell Arrays

Reference arrays within cell arrays

```matlab
>> a{4,1} = [2 1 4; 9 8 6]
>> a
>> a{4,1}(2,2)
```
Using Cell Arrays

Reference arrays within cell arrays

```matlab
>> a{4,1} = [2 1 4; 9 8 6]
>> a
This?
>> a(4,1)(2,2)
```
Using Cell Arrays

What would this give you?

```matlab
>> a{4,3} = {'house' 27 'shoe'}
```

```matlab
>> a
```

```matlab
>> a{4,3}{1,3}(2)
```
Using Cell Arrays

Extract part of a cell array

```matlab
>> a{1:2, 2:3}
```

vs

```matlab
>> a(1:2, 2:3)
```
Using Cell Arrays

Extract part of a cell array

```matlab
>> a{1:2, 2:3}
```

vs

```matlab
>> a(1:2, 2:3)
```

What does this do?

```matlab
>> {a{1:2, 2:3}}
```
Using Cell Arrays

Create multidimensional cell arrays

```matlab
>> a{1,2,3} = 'house'
```

Preallocate

```matlab
>> b = cell(4,3)
```

```matlab
>> c = cell(2,5,5)
```
Removing elements from cell arrays

You can do this

```plaintext
>> a{2,2,1} = [2 3 ; 5 6]
>> a{2,2,1} = []
```
Accessing Cell Arrays

data{1,1} = 'Jim';
data{1,2} = [98 99 97];
data{2,1} = 'Frank';
data{2,2} = [88 87 90];
data{3,1} = 'Joe';
data{3,2} = [76 77 71];

a = data{:,2}
b = data(:,2)
Accessing Cell Arrays

\[ b = \text{data}(::,2) \]

convert from cell array to array
\[ c = \text{cell2mat}(b) \]
\[ \text{whos} \ c \]
Imagine a between-subject experiment

How could we create a data structure using a cell array?

One B/S variable with 3 levels
  10 subjects in level 1
  12 subjects in level 2
  8 subjects in level 3

One W/S variable with 2 levels

One W/S variable with 5 levels
Imagine a between-subject experiment

clear all

NBet = 3
NWith1 = 2
NWith2 = 5
Nsubj = zeros(1,NBet)
Nsubj(1) = 10
Nsubj(2) = 12
Nsubj(3) = 8

data{1} = zeros(Nsubj(1),NWith1,NWith2)
data{2} = zeros(Nsubj(2),NWith1,NWith2)
data{3} = zeros(Nsubj(3),NWith1,NWith2)
Homework 3: For Q4, learn about and use structures

Structures


Structures

Structures are another kind of data structure that can take heterogeneous data.

Whereas cell arrays reference by an index number, structures reference by name.
Structures

Imagine we have subject records like this

```python
>> subject.name = 'John Smith'
>> subject.age = 27
>> subject.consent = true
>> subject.data = [1.2, 3.2; 4.3, 1.1]
```

```python
>> subject
>> subject.age
>> subject.data(1,2)
```
Structures

Now do this

>> size(subject)
Structures

Now do this

```matlab
>> size(subject)
```

Matlab assumes these are arrays of structures.

So you can do this

```matlab
>> subject(1)
>> subject(1,1)
```
Structures

And add new subjects

```python
>> subject(2).name = 'Frank Jones'
>> subject(2).age = 21
>> subject(2).consent = false
>> subject(2)
```

It fills in the missing fields and assigns them to [ ]
Structures

Imagine we do this now, adding a new field:

```plaintext
>> subject(2).ses = 'high'
```

```plaintext
>> subject(2)
```
Imagine we do this now, adding a new field:

```plaintext
>> subject(2).ses = 'high'
```

The new field is added to subject 1 as well.
Structures

You can also do this

```matlab
>> subject(3) = struct('name', 
    'Andy Stone', 'age', 31, ...
    'consent', true, 'data', [],
    'ses', 'low')
```

But 😞 you can’t leave any fields unspecified

```matlab
>> subject(4) = struct('name', 
    'Joe Russo', 'age', 22, ...
    'consent', false, 'data', [])
```
Structures

You can eliminate an entry from the array of structures

```matlab
>> subject(3) = [ ]
```
Structures

You can have structures within structures

```python
>> subject(1).experiment1.date = "09-May-2012"
>> subject(1).experiment1.tester = "Mary Blake"
>> subject(1).experiment1.data = []

>> subject(1)
>> subject(1).experiment1
```
Structures

Imagine you want the data from all the subjects. What does this give you?

```plaintext
>> subject(:).data
```
Structures

Imagine you want the data from all the subjects. What does this give you?

```bash
>> subject(:).data
```

Does this seem odd?

What is the code doing?
Structures

Imagine you want the data from all the subjects. What does this give you?

```ruby
>> subject(:).data
```

Does this seem odd?
What is the code doing?

What does this do?

```ruby
>> {subject(:).data}
```
Structures

>> {subject(:).data}

Could you turn this into an array?

>> cell2mat({subject(:).data})

But how is this organized?
Structures

>> {subject(:).data}
Could you turn this into an array?

>> cell2mat({subject(:).data})

But how is this organized?

What is this doing?

>> cell2mat({subject(:).data}')

NOTE: this is useful for the homework
Tables
(new to Matlab 2013b)
Some documentation and videos on table types in Matlab


<table>
<thead>
<tr>
<th>SubjectNum</th>
<th>Name</th>
<th>Age</th>
<th>Expertise</th>
<th>Memory</th>
<th>Perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob Smith</td>
<td>43</td>
<td>Novice</td>
<td>0.65</td>
<td>0.76</td>
</tr>
<tr>
<td>4</td>
<td>Mary Jones</td>
<td>53</td>
<td>Expert</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>5</td>
<td>Tom Heard</td>
<td>23</td>
<td>Expert</td>
<td>0.79</td>
<td>0.71</td>
</tr>
<tr>
<td>8</td>
<td>Frank Rizzo</td>
<td>65</td>
<td>Expert</td>
<td>0.91</td>
<td>0.84</td>
</tr>
<tr>
<td>19</td>
<td>Maria Smythe</td>
<td>47</td>
<td>Novice</td>
<td>0.70</td>
<td>0.65</td>
</tr>
<tr>
<td>7</td>
<td>Henry Williams</td>
<td>35</td>
<td>Novice</td>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>24</td>
<td>Laura Franklin</td>
<td>42</td>
<td>Novice</td>
<td>0.75</td>
<td>0.57</td>
</tr>
<tr>
<td>9</td>
<td>Molly Patterson</td>
<td>28</td>
<td>Expert</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>51</td>
<td>Stanley Teal</td>
<td>21</td>
<td>Expert</td>
<td>0.85</td>
<td>0.79</td>
</tr>
<tr>
<td>79</td>
<td>Lolly Band</td>
<td>32</td>
<td>Novice</td>
<td>0.61</td>
<td>0.53</td>
</tr>
<tr>
<td>SubjectNum</td>
<td>Name</td>
<td>Age</td>
<td>Expertise</td>
<td>Memory</td>
<td>Perception</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>-----</td>
<td>-----------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>1</td>
<td>Bob Smith</td>
<td>43</td>
<td>Novice</td>
<td>0.65</td>
<td>0.76</td>
</tr>
<tr>
<td>4</td>
<td>Mary Jones</td>
<td>53</td>
<td>Expert</td>
<td>0.87</td>
<td>0.86</td>
</tr>
<tr>
<td>5</td>
<td>Tom Heard</td>
<td>23</td>
<td>Expert</td>
<td>0.79</td>
<td>0.71</td>
</tr>
<tr>
<td>8</td>
<td>Frank Rizzo</td>
<td>65</td>
<td>Expert</td>
<td>0.91</td>
<td>0.84</td>
</tr>
<tr>
<td>19</td>
<td>Maria Smythe</td>
<td>47</td>
<td>Novice</td>
<td>0.70</td>
<td>0.65</td>
</tr>
<tr>
<td>12</td>
<td>Henry Williams</td>
<td>35</td>
<td>Novice</td>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>24</td>
<td>Laura Franklin</td>
<td>42</td>
<td>Novice</td>
<td>0.75</td>
<td>0.57</td>
</tr>
<tr>
<td>32</td>
<td>Molly Patterson</td>
<td>28</td>
<td>Expert</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>51</td>
<td>Stanley Teal</td>
<td>21</td>
<td>Expert</td>
<td>0.85</td>
<td>0.79</td>
</tr>
<tr>
<td>79</td>
<td>Lolly Band</td>
<td>32</td>
<td>Novice</td>
<td>0.61</td>
<td>0.53</td>
</tr>
</tbody>
</table>
reading in a table

from an excel sheet

```matlab
>> dataxls = readtable('TestDataSheet.xlsx');
```

from a comma-delimited text file

```matlab
>> datatxt = readtable('TestDataSheet.dat');
```
accessing data

from a column
>> dataxls.Name

a subset of rows/columns
>> dataxls{1:3, 'Name'}

more complex indexing
>> dataxls{dataxls.Memory>.7, 'Name'};
sorting rows

based on a particular column

>> sortrows(dataxls, 'Memory', 'ascend')
concatenation

extracting subset of columns of the same type

```matlab
>> d = dataxls{:,{'Memory','Perception'}}
>> whos d
```

but this will not work (incompatible types)

```matlab
>> d = dataxls{:,{'Expertise','Perception'}}
```
create a new subtable from an existing table

>> d = dataxls(:, {'Expertise', 'Memory', ...
                     'Perception'})

>> whos d
don't confuse { and (  

```matlab
>> d = dataxls{:,{'Memory','Perception'}}
```

vs

```matlab
>> d = dataxls(:, {'Expertise', 'Memory', 'Perception'})
```
create a new field (column)

```python
dataxls.Score = 2*dataxls.Memory + ...  
               3*dataxls.Perception
```
join two tables

<table>
<thead>
<tr>
<th>SubjectNum</th>
<th>NewMem</th>
<th>NewPerc</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.92</td>
<td>0.80</td>
</tr>
<tr>
<td>32</td>
<td>0.89</td>
<td>0.85</td>
</tr>
<tr>
<td>51</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>12</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td>4</td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>24</td>
<td>0.88</td>
<td>0.86</td>
</tr>
<tr>
<td>79</td>
<td>0.72</td>
<td>0.58</td>
</tr>
<tr>
<td>1</td>
<td>0.69</td>
<td>0.77</td>
</tr>
<tr>
<td>19</td>
<td>0.71</td>
<td>0.70</td>
</tr>
<tr>
<td>5</td>
<td>0.77</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Different order of SubjectNum
join two tables

>> newdata = readtable('NewTestDataSet.xlsx');

>> combined = join(dataxls, newdata);
creating a table from existing data

LastName = {'Smith';'Johnson';'Williams'; ...
  'Jones';'Brown'};
Age = [38;43;38;40;49];
Height = [71;69;64;67;64];
Weight = [176;163;131;133;119];
BloodPressure = [124 93; 109 77; 125 83; ... 117 75; 122 80];

ewtable = table(LastName, Age, Height,
  Weight, BloodPressure)
save a table

write a table to txt format

```matlab
>> writetable(newtable, 'NewTable.txt');
```

on a PC, but not a Mac, you can also write to xls

```matlab
>> writetable(newtable, 'NewTable.xlsx');
```
Other Kinds of Data Structures

categorical arrays (ordinal and nominal data)
sql databases
linked lists
stacks
queues
trees
heaps
graphs
hash tables
dictionaries
objects
Programming Methods and Control Flow
Control Flow

Control flow statements in a programming language control whether and how often other statements in that language are executed.

If-Then-Else  For Loops

Switch Statements  While Loops
If-Then-Else Statements

controls whether or not a particular statement will or will not be executed based on the results of some logical test

```plaintext
if condition
  action
end
```
If-Then-Else Statements

\[ x = 12; \]

\[ \text{if } x > 10 \]
\[ \quad x = x - 10; \]
\[ \text{end} \]

\[ x \]
If-Then-Else Statements

\[ x = 12; \]
\[ \text{flag} = (x < 12) \]
\[ \text{if } \text{flag} \]
\[ \quad x = x - 10; \]
\[ \text{end} \]

\[ x \]
If-Then-Else Statements

x = 12;
if x > 10
   x = x-10;
else
   x = 1
end

x
If-Then-Else Statements

```plaintext
x = 12;
if x > 10
    x = x-10;
elseif x < 5
    x = 0
else
    x = 1
end

x
```
If-Then-Else Statements

$x = 7;$
if $x > 10$
    $x = x - 10;$
elseif $x < 5$
    $x = 0$
else
    $x = 1$
end

$x$
If-Then-Else Statements

```plaintext
x = 12;
if x > 10
    x = x-10;
elseif x == 12
    x = 0
else
    x = 1
end

x
```
x = 2;
if x == 1
    txt = 'option 1';
elseif x == 2
    txt = 'option 2';
elseif x == 3
    txt = 'option 3';
elseif x == 4
    txt = 'option 4';
end
Switch Statements

x = 2;
switch x
    case 1
        txt = 'option 1';
    case 2
        txt = 'option 2';
    case 3
        txt = 'option 3';
    case 4
        txt = 'option 4';
    otherwise
        txt = 'invalid option';
end
name = 'Bob';
switch name
  case 'Bob'
    txt = 'Robert';
  case 'Tom'
    txt = 'Thomas';
  case 'Rick'
    txt = 'Richard';
  case 'Tony'
    txt = 'Anthony';
  otherwise
    txt = 'invalid option';
end
Switch Statements

basically just an extended sequence of if statements in a structured format
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf('%2d : %4.2f\n', i, data(i));
end
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf('%2d : %4.2f\n', i, data(i));
end

Key words delimit the beginning and end of a for loop
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf('%2d : %4.2f\n', i, data(i));
end
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf('%2d : %4.2f
', i, data(i));
end

Defines the ranges of the for loop
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf(' %2d : %4.2f
', i, data(i));
end

A different sense of “=”

for i ∈ 1:length(data)
    fprintf('%2d : %4.2f
', i, data(i));
end
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = 1:length(data)
    fprintf('%2d : %4.2f\n', i, data(i));
end

What does this get translated to?

for i = 1:length(data)
    fprintf('%2d : %4.2f\n', i, data(i));
end
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

for i = [1 2 3 4 5]
    fprintf('%2d : %4.2f\n', i, data(i));
end
For Loops

data = [2.3 3.1 4.5 1.1 6.3];

i does not get set equal to the array entirely, but gets set equal to each element of the array each time the loop repeats itself

for i = [1 2 3 4 5]
    fprintf('%2d : %4.2f\n', i, data(i));
end
For Loops

You could do this. What happens?
```
data = [1.3 2.4 3.1 4.5];
for d = data
    d
end
```
For Loops

But what about this?
data = [1.3 2.4 3.1 ; 1.1 5.3 4.5];
for d = data
    d
end
For Loops

How would we print out every data point?

```
data = [1.3 2.4 3.1 ; 1.1 5.3 4.5];
```
For Loops

How would we print out every data point?

```matlab
data = [1.3 2.4 3.1 ; 1.1 5.3 4.5];
for i = 1:size(data,1)
    for j = 1:size(data,2)
        fprintf('%(d,%d) : %f
', i, j, data(i,j));
    end
end
```
For Loops

What does this do?
data = [1.3 2.4 3.1 ; 1.1 5.3 4.5];
sum = 0;
for i = 1:size(data,1)
    for j = 1:size(data,2)
        sum = sum + data(i,j);
    end
end
sum = sum / numel(data)

Not efficient. There are other ways to do this.
While Loops

while condition
  actions
end
While Loops

while condition

  actions

end

How would we replicate a “for loop” with a while loop?
for i=1:10
  i
end
While Loops

N = 10;
i = 1;
while i<=N
    i = i + 1;
end
While Loops

\[ N = 10; \]
\[ i = 1; \]
\[ \text{while } i \leq N \]
\[ \quad i \]
\[ \quad i = i + 1; \]
\[ \text{end} \]

A \textbf{for} loop loops a set number of times.  
A \textbf{while} loop can loop a variable number of times.
Example

what will this do?

a = 1
while a
  a = a/2;
end
Example

data = [420 500 130 540 800 120 930 100 340];

How could we extract only the RTs greater than 150ms and less than 750ms and put them in an array?

First, let’s do it “the long way”. Imagine you weren’t working in a language like Matlab with tons of built-in functions.
Example

data = [420 500 130 540 800 120 930 100 340];

How could we extract only the RTs greater than 150ms and less than 750ms and put them in an array?

idx = 1
for i=1:length(data)
    if ((data(i) > 150) & (data(i) < 750))
        data2(idx) = data(i);
        idx = idx+1;
    end
end