An Introduction to MATLAB

Day 1
Simon Mitchell
Simon.Mitchell@ucla.edu
* High level language
* Programing language and development environment
* Built-in development tools
* Numerical manipulation
* Plotting of functions and data
* Implement algorithms
* Create models and applications
* Many built in functions
* Interface with other languages
* Create graphical interfaces
The MATLAB Environment

![MATLAB Environment](image-url)
Command Window Basics

```
>> 2+2
ans =
    4
>> 3^2
ans =
    9
>> sin(pi/2)
ans =
    1
>> ans
ans =
    1
```

```
>> 7/0
ans =
    Inf
>> (10-3)/(12-(6*2))
ans =
    Inf
>> 1e04*1e06
ans =
    1.0000e+10
```
Common Arithmetic Operators

+ Addition
- Subtraction
* Multiplication
/ Division
^ Exponential
() Order operations
Semicolons in MATLAB

Suppress the output from a MATLAB expression

Workspace

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

```matlab
>> 3^2
ans =
9
>> 3^2;
```
Suppress the output from a MATLAB expression
Variables

Variable name = variable value

Be careful of i,j

ans =

0.0000 + 1.0000i
MATLAB is CASE SENSITIVE

```matlab
>> x
x =
    9

>> X
Undefined function or variable 'X'.
Did you mean:
>> x
x =
    9

>>
```
Close MATLAB

Reopen MATLAB
Initially Workspace is empty
Load your workspace
Variables to result of expression

If an expression is not stored as a variable it will be stored as ‘ans’
Multiple assignments and ‘who’

```matlab
>> a=4; b=2; c=a*b

c =

     8

>> who

Your variables are:
a   ans   b   c   x   y   z
fx
```
whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>ans</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
</tbody>
</table>

>> clear a
>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ans</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
</tbody>
</table>
```matlab
>> x = pi
x =
    3.1416

>> format long
>> x = pi
x =
    3.141592653589793

>> format short
>> x
x =
    3.1416

>> format rat
>> pi
ans =
    355/113

>> format bank
>> pi
ans =
    3.14
```

‘format short’
to get back to normal
Numbers are actually 1x1 Matrices

```matlab
>> format short
>> x
x =
    3.1416
>> size(x)
an =
     1   1
```

```matlab
>> rowVector = [1,2,3,4,5]
rowVector =
    1    2    3    4    5
>> rowVector = [1:5]
rowVector =
    1    2    3    4    5
>> size(rowVector)
ans =
     1    1
>> size(rowVector)
ans =
     1    5
```
Operations apply to matrices

```matlab
>> rowVector
rowVector =
    1    2    3    4    5

>> secondVector = [4,3,2,1,0]
secondVector =
    4    3    2    1    0

>> rowVector+secondVector
ans =
    5    5    5    5    5    5
```
Operations apply to matrices

```plaintext
>> rowVector+secondVector
ans =
    5   5   5   5   5   5

>> columnVector = [1;2;3;4;5]
columnVector =
    1
    2
    3
    4
    5

>> columnVector + rowVector
Error using +
Matrix dimensions must agree.
```
Put . before an operator to make it element-wise. 

\[ \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \]

\[ \begin{pmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{pmatrix} \]

\[
\begin{align*}
\text{myMatrix} + \text{mySecondMatrix} &= \\
\text{ans} &= \\
\text{myMatrix} \times \text{mySecondMatrix} &= \\
\text{disp(ans)} &= \\
\end{align*}
\]

\[
\begin{pmatrix} 6 & 10 & 12 \\ 12 & 10 & 6 \end{pmatrix}
\]

disp(variable) displays the contents of a variable.
fprintf – print something to command window

>> numberOfDays = 3;
>> nameOfInstructor = 'Simon';
>> fprintf('We are learning MATLAB with %s over %d days',nameOfInstructor, numberOfDays)
We are learning MATLAB with Simon over 3 days
>> fprintf('We are learning MATLAB with %s over %d days\n',nameOfInstructor, numberOfDays)
We are learning MATLAB with Simon over 3 days

%s   Format as a string.
%d   Format as an integer.
%f   Format as a floating point value.
%e   Format as a floating point value in scientific notation.
%g   Format in the most compact form: %f or %e.
\n   Insert a new line in the output string.
\t   Insert a tab in the output string.
Concatenating matrices

```matlab
>> myMatrix = [0,1,2,3];
>> myOtherMatrix=[4,5,6,7];
>> cat(1,myMatrix,myOtherMatrix)

ans =

   0    1    2    3
   4    5    6    7

>> cat(2,myMatrix,myOtherMatrix)

ans =

    0    1    2    3    4    5    6    7

>> [myMatrix,myOtherMatrix]

ans =

    0    1    2    3    4    5    6    7

>> [myMatrix;myOtherMatrix]

ans =

    0    1    2    3
   4    5    6    7

>> find(ans)

ans =

    2
    3
    4
    5
    6
    7
    8
```
Concatenating matrices

>> [myMatrix;myOtherMatrix]
ans =
     0   1   2   3
     4   5   6   7

>> length(ans)
ans =
     4

>> size(ans)
ans =
     1   1

>> size([myMatrix;myOtherMatrix])
ans =
     2   4

Why is this 1,1?
Many Functions work on columns

```
myBigMatrix =

    0   1   2   3
    4   5   6   7

>> max(myBigMatrix)
ans =

    4   5   6   7

>> min(myBigMatrix)
ans =

    0   1   2   3

>> sum(myBigMatrix)
ans =

    4   6   8  10
```
Working with m-files

* 3 Ways of running code
  * Command Window
  * Scripts
  * Functions
New Script

>> edit
>> edit myNewFile.m
Our First Script

1 - genesExp1 = 260;
2 - genesExp2 = 58;
3 - genesExp3 = 79;
4 - totalGenes = genesExp1 + genesExp2 + genesExp3;
5 - avgGenes = totalGenes / 3;
6 - disp(avgGenes);
Scripts can create variables in the workspace.

```matlab
1 - genesExp1 = 260;
2 - genesExp2 = 58;
3 - genesExp3 = 79;
4 - totalGenes = genesExp1+genesExp2+genesExp3;
5 - avgGenes=totalGenes/3;
6 - disp(avgGenes);
```
Data Types

* Type declarations are not necessary in MATLAB
* MATLAB automatically decides data type
**Data Types**

- **single** - single precision numerical data
- **double** - double precision numerical data
- **logical** - logical values of 1 or 0, represent true and false respectively
- **char** - character data (strings are stored as vector of characters)
- **cell array** - array of indexed cells, each capable of storing an array of a different dimension and data type
- **structure** - named fields capable of storing an array of a different dimension and data type
- **function handle** - pointer to a function
- **user classes** - objects constructed from a user-defined class

Int8 uint8 int16 uint16 int32 uint32 int64 uint64 – don’t worry about these
Data Types

```matlab
>> edit dataTypes.m
```

```matlab
1 - geneName = 'nfbkbia';
2 - disp(geneName);
3 - geneExp = 367.54323;
4 - doubleVal = double(geneExp);
5 - intVal = uint32(geneExp);
6 - disp(doubleVal);
7 - disp(intVal);
8 - disp(isinteger(doubleVal));
9 - disp(isinteger(intVal));
```

```
Command Window

>> dataTypes

nfbkbia
367.5432

368

0

1
```

fx >>
Testing Data Types

```
x = [1 2 3]
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
```

```
x =

1  2  3

ans =

0

ans =

1

ans =

1

ans =

0
```
Relational Operators

<   Less than
>   Greater than
>=  Less than or equal to
>=  Greater than or equal to
==  Equal to
~=  Not equal to

Useful for if statements!
If Statement

```
1 - exp1 = 400;
2 - exp2 = 500;
3 - if (exp1 >= exp2)
    max = exp1
4 - end
```

Never run
If else Statement

```
1 - exp1 = 400;
2 - exp2 = 500;
3 - if (exp1 >= exp2)
4 -     max = exp1
5 - else
6 -     max = exp2
7 - end
```

Command Window

```
>> edit ifStatement.m
>> ifStatement

max =

500
```

$f_x$ >>
If elseif Statement

```matlab
ifStatement.m

1 - exp1 = 400;
2 - exp2 = 400;
3 - if (exp1 > exp2)
4 -     max = exp1
5 - elseif (exp1==exp2)
6 -     max = 'Both are equal'
7 - else
8 -     max = exp2
9 - end

Command Window

max =
Both are equal
```
Nested if Statement

```matlab
exp1 = 440;
exp2 = 400;
if (exp1 > exp2)
    if ((exp1-exp2)>50)
        fprintf('exp1 is much bigger')
    else
        fprintf('exp1 is slightly bigger')
    end
else
    fprintf('exp1 is slightly bigger')
end
max = 'exp1'
elseif (exp1==exp2)
    max = 'Both are equal'
else
    max = exp2
end
```

Command Window:
```
>> ifStatement
exp1 is slightly bigger
max =
exp1
```
For loops

```matlab
for a = 10:20
    fprintf('value of a: %d\n', a);
end
```

Command Window:
```matlab
>> edit forLoops.m
>> edit forLoopExample.m
>> forLoopExample
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```
Write a script that takes a list of gene names and gene expression values and outputs only those gene names over a threshold.

PSEUDO CODE:
For each gene in a list:
  if its expression value is over the threshold
    print the gene name and the expression value
Combining What We’ve Learned

PSUEDO CODE:
For each gene in a list:
  if its expression value is over the threshold
  print the gene name and the expression value
Combining What We’ve Learned

```matlab
exp1Genes = {'IL1A', 'NFKBIA', 'BCL2', 'STAT5A', 'CCL5', 'MYC', 'IRF1'};
exp1Results = [4, 56, 21, 12, 39, 21, 3];
threshold = 20;
for i = 1:length(exp1Results)
    if (exp1Results(i) > threshold)
        fprintf('Gene %s expression %d\n', exp1Genes{i}, exp1Results(i));
    end
end
```