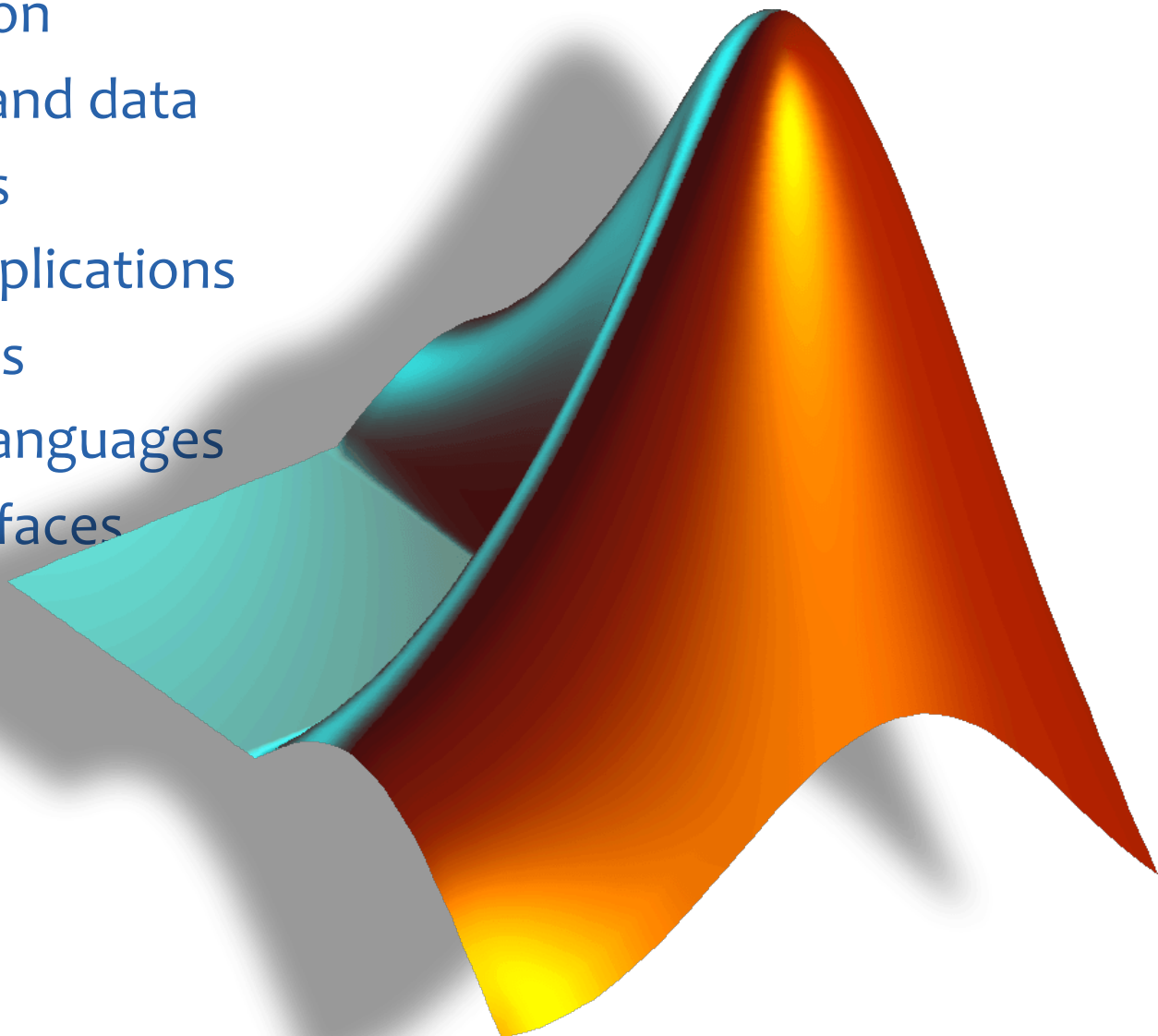
An Introduction to MATLAB

Day 1

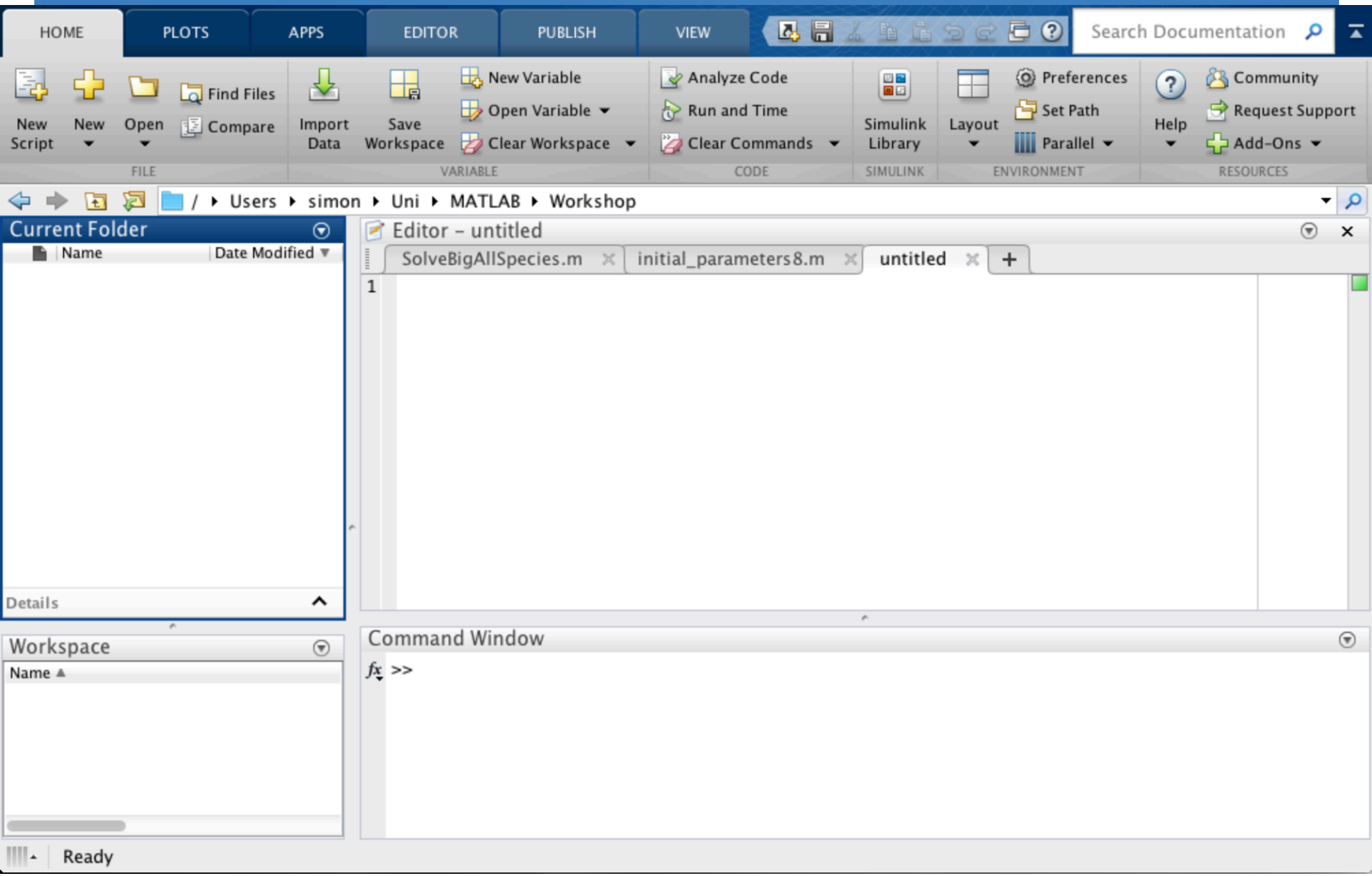
Simon Mitchell

Simon.Mitchell@ucla.edu

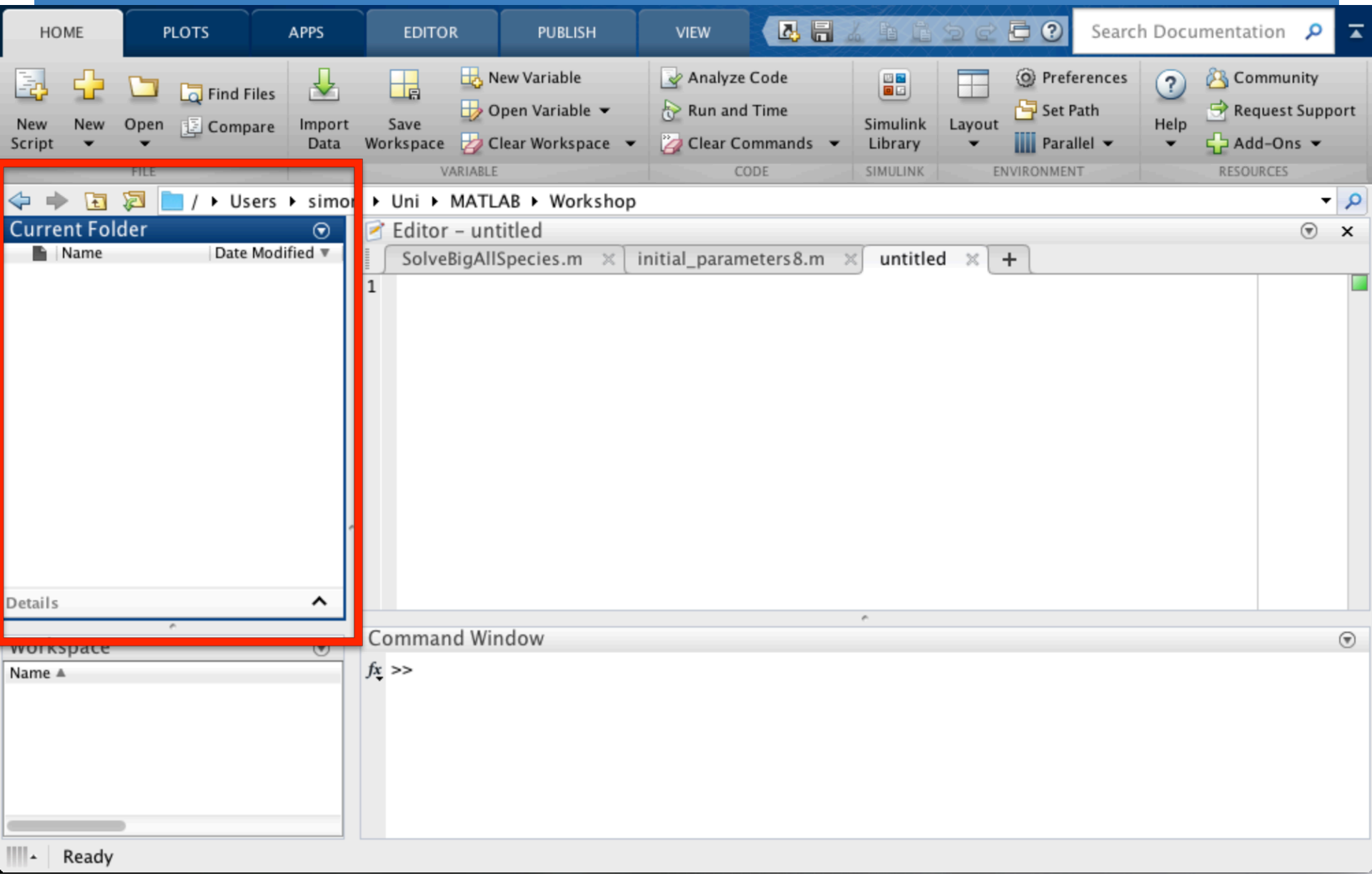
* High level language
* Programing language and development environment
* Built-in development tools
* Numerical manipulation
* Plotting of functions and data
* Implement algorithms
* Create models and applications
* Many built in functions
* Interface with other languages
* Create graphical interfaces



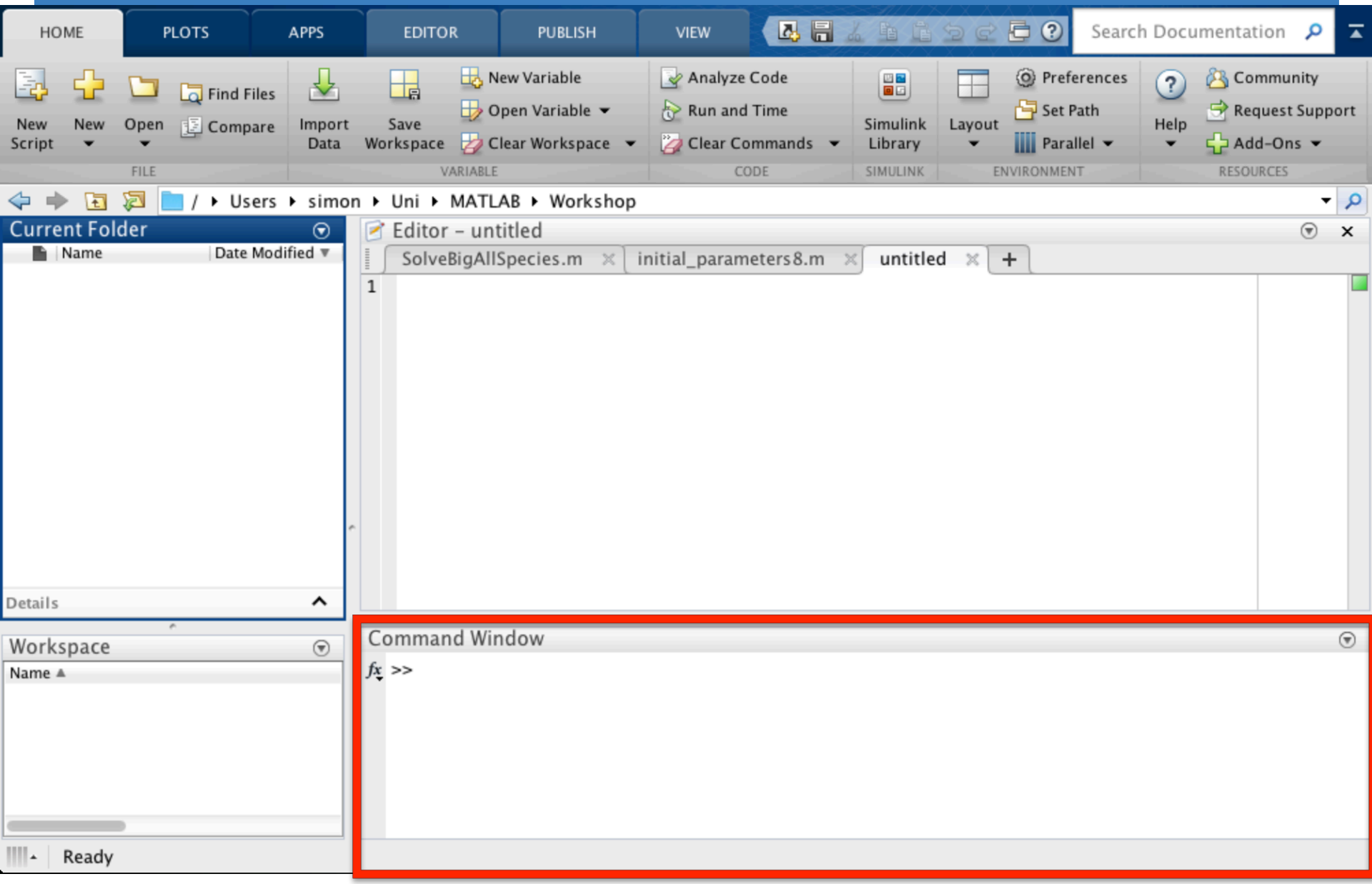
The MATLAB Environment



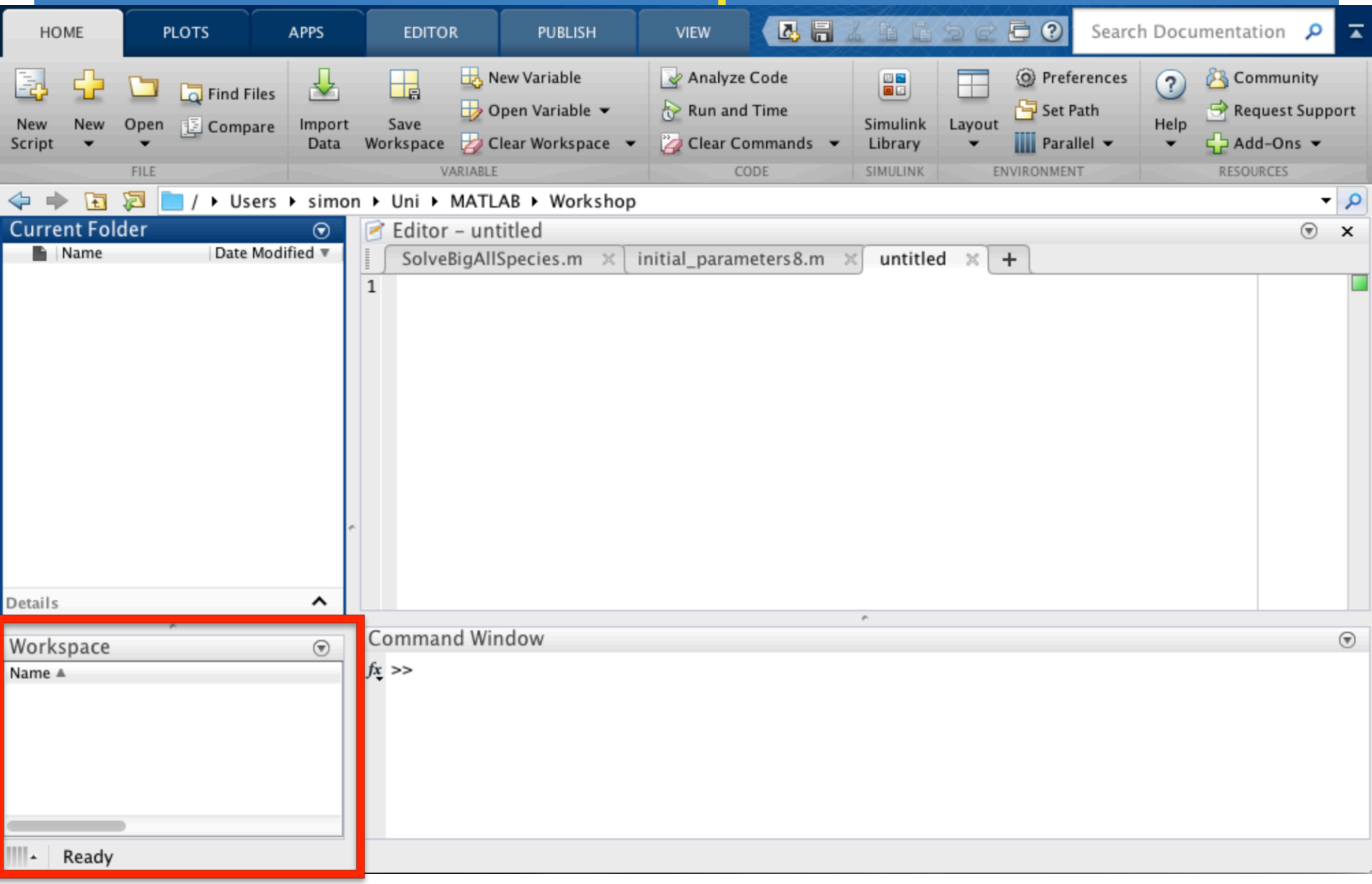
Current Folder



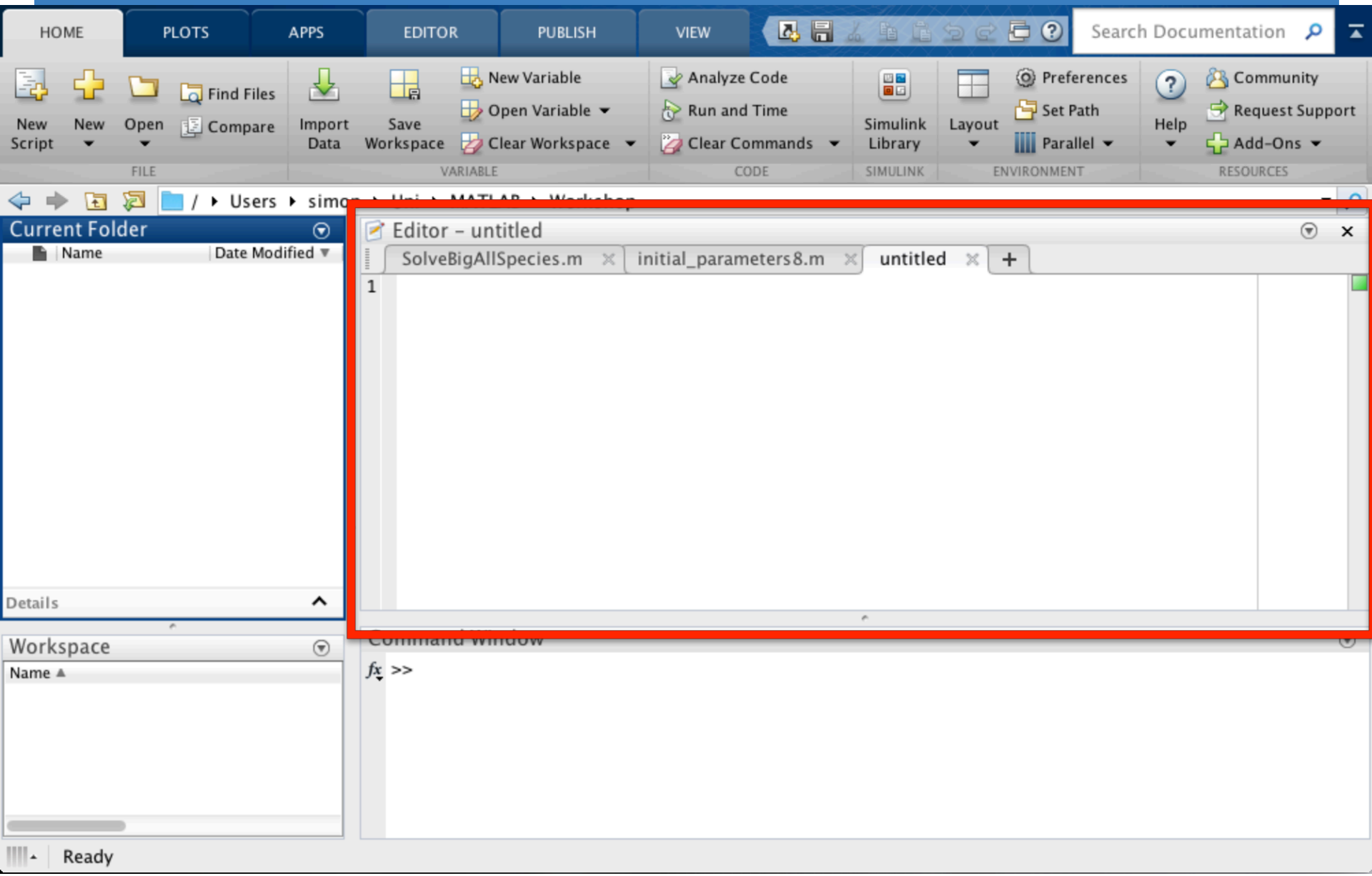
Command Window >>



Workspace


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
/ ▸ Users ▸ simon ▸ Uni ▸ MATLAB ▸ Workshop
Current Folder
Name
Date Modified
Details
Editor - untitled
SolveBigAllSpecies.m
initial_parameters8.m
untitled
Workspace
Name
Command Window
>>
Ready

Editor


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
Users
Current Folder
Name
Date Modified
Details
Editor - untitled
SolveBigAllSpecies.m
initial_parameters8.m
untitled
Workspace
Name
>>
Ready

Command Window Basics

Command Window

```
>> 2+2

ans =

     4

>> 3 ^ 2

ans =

     9

>> sin(pi/2)

ans =

     1

>> ans

ans =

     1

>>
```

Command Window

```
>> 7/0

ans =

   Inf

>> (10-3)/(12-(6*2))

ans =

   Inf

>> 1e+04*1e+06

ans =

   1.0000e+10

>>
```

Common Arithmetic Operators

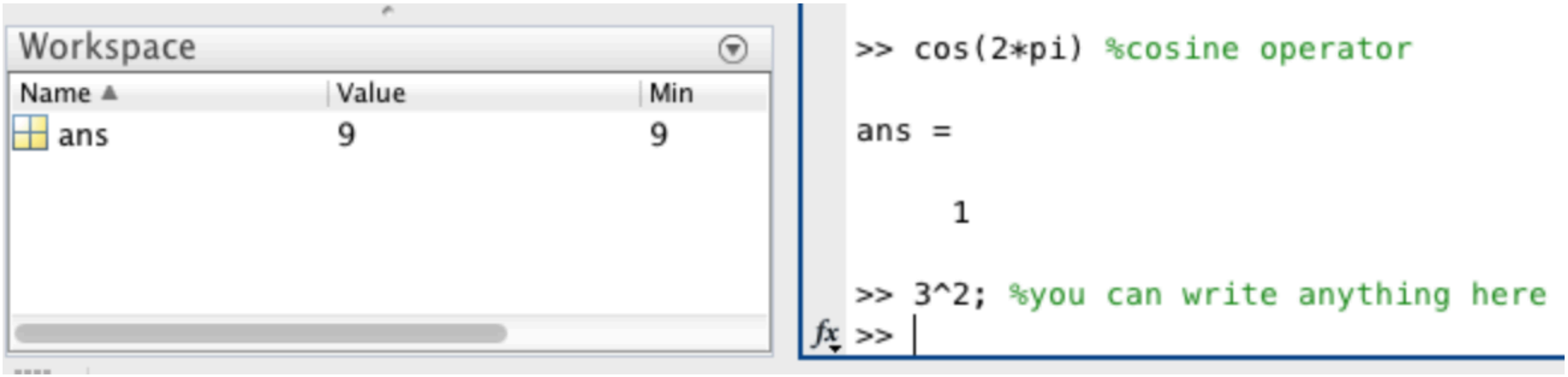
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponential
()	Order operations

Semicolons in MATLAB

Suppress the output from a MATLAB expression

Comments %

Suppress the output from a MATLAB expression



Variables

Variable name = variable value

Workspace

Name	Value	Min
ans	9	9
x	9	9
y	3	3
z	27	27

```
ans =

     1

>> 3^2; %you can write anything here
>> x=ans;
>> y=3;
>> z=x*y;
>>
```

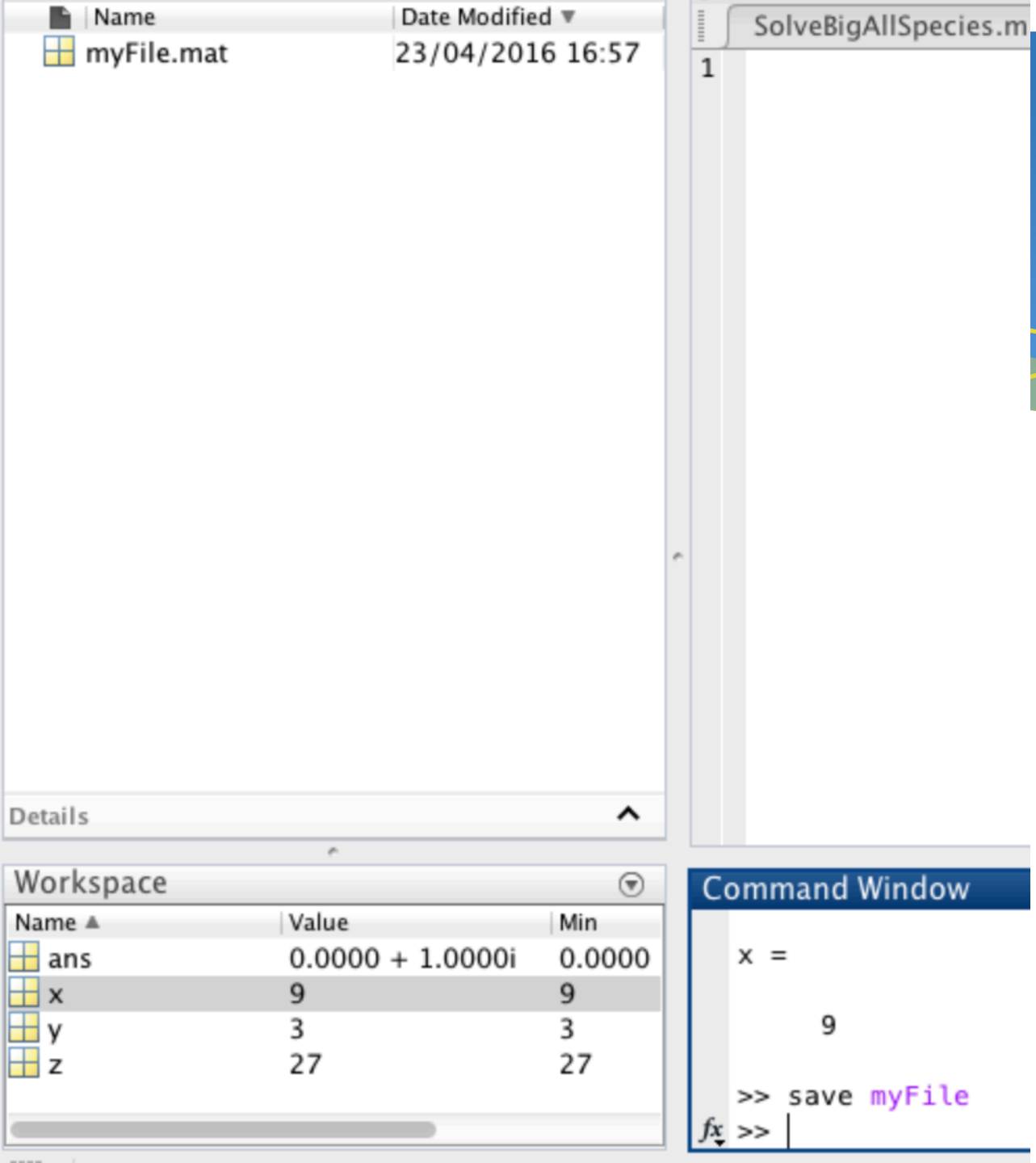
Be careful of i,j

```
>> i

ans =

   0.0000 + 1.0000i
```

Variables 2

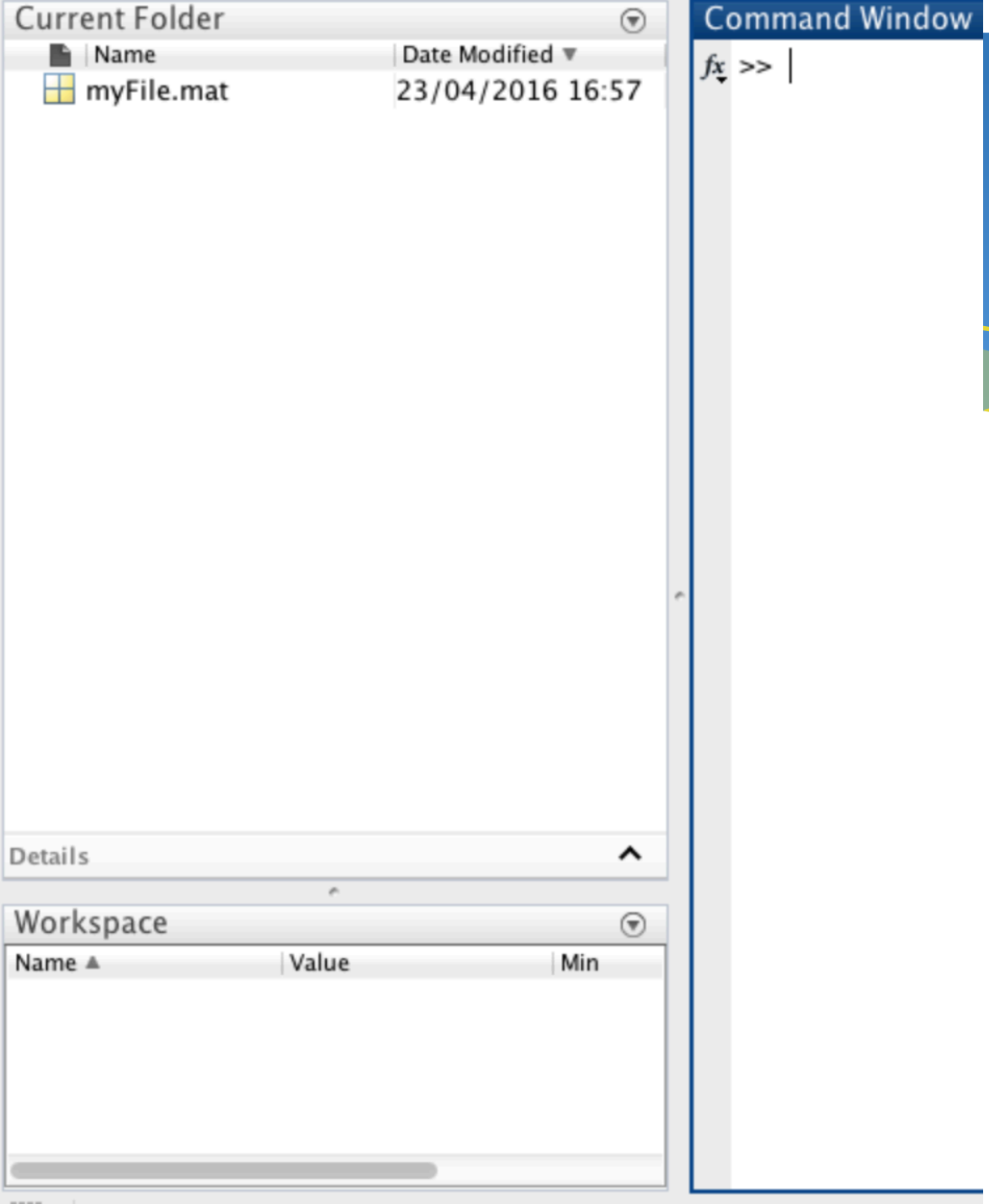
MATLAB is CASE SENSITIVE

```
>> x

x =

     9

>> X
Undefined function or variable 'X'.

Did you mean:
>> x

x =

     9

>>
```

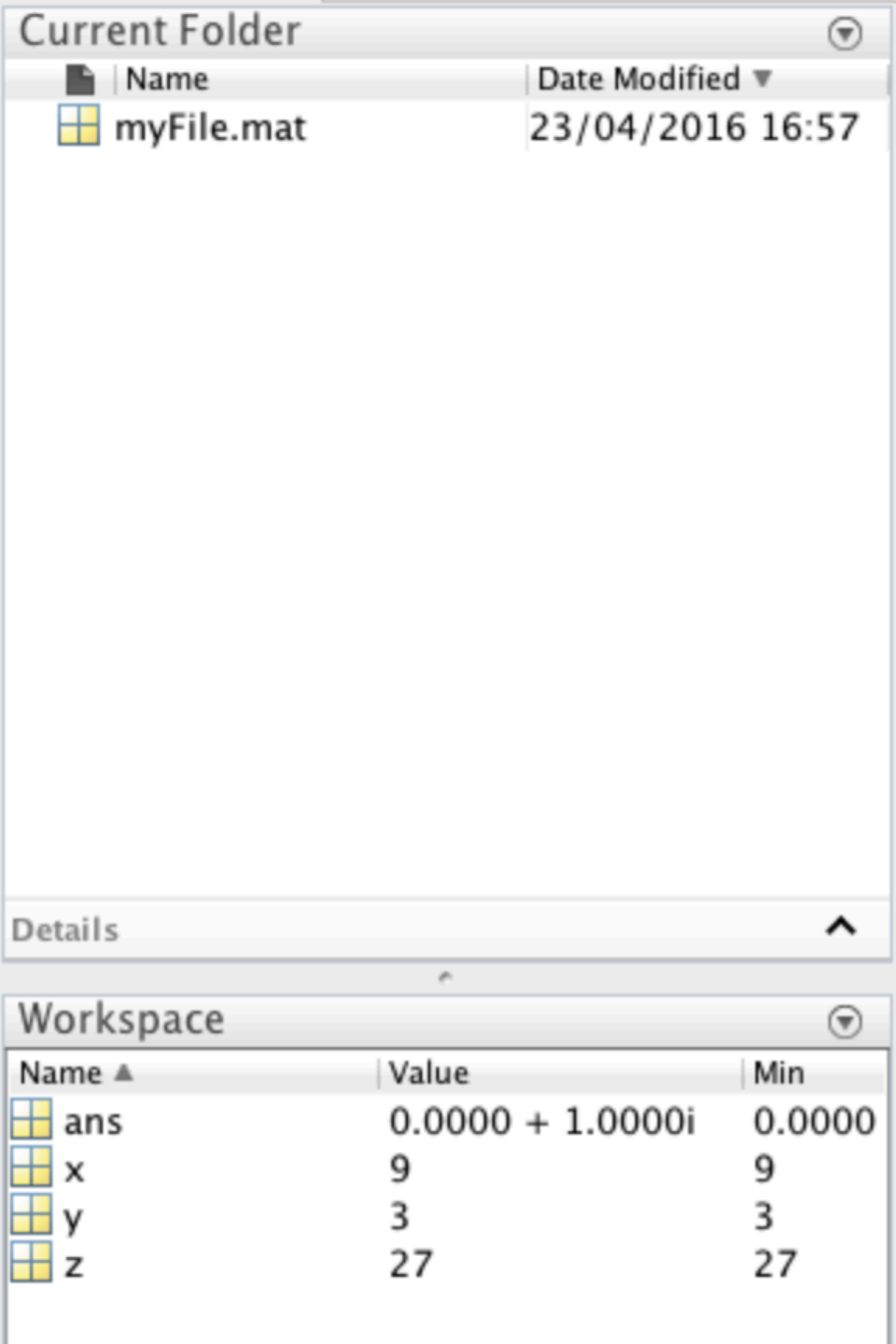


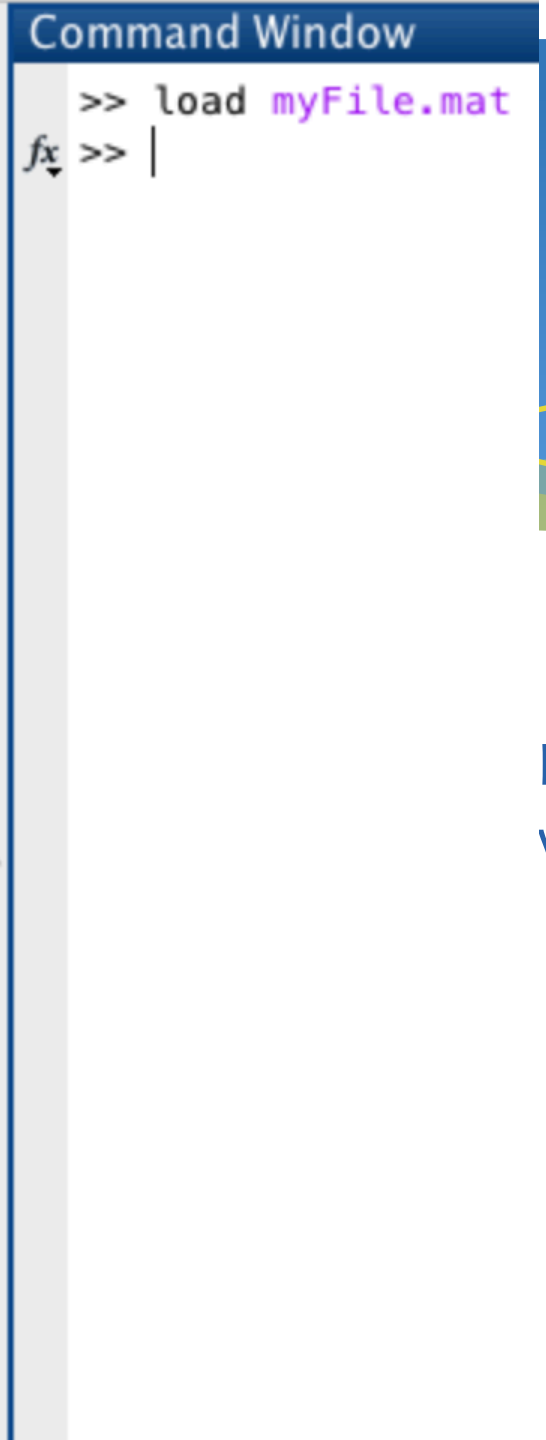
Close MATLAB

Reopen MATLAB



Initially Workspace is empty





Load your workspace

Variables to result of expression

Details

Workspace

Name	Value	Min
ans	5	5
x	4	4
y	3	3
z	27	27

Command Window

```
>> x=sqrt(16)

x =

     4

>> sqrt(25)

ans =

     5

>>
```

If an expression is not stored as a variable it will be stores as ‘ans’

Multiple assignments and ‘who’

Command Window

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

c =

     8

>> who

Your variables are:

a    ans  b    c    x    y    z

>>
```

whos

Command Window

```
>> whos
  Name      Size            Bytes  Class     Attributes

  a         1x1                 8  double
  ans       1x1                 8  double
  b         1x1                 8  double
  c         1x1                 8  double
  x         1x1                 8  double
  y         1x1                 8  double
  z         1x1                 8  double

>> clear a
>> whos
  Name      Size            Bytes  Class     Attributes

  ans       1x1                 8  double
  b         1x1                 8  double
  c         1x1                 8  double
  x         1x1                 8  double
  y         1x1                 8  double
  z         1x1                 8  double

>> clear
>> whos
```

format

Command Window

```
>> x=pi

x =

    3.1416

>> format long
>> x=pi

x =

   3.141592653589793

>> format short
>> x

x =

    3.1416

>> format bank
>> pi

ans =

          3.14
```

```
>> format rat
>> pi

ans =

     355/113

>>
```

'format short'

to get back to normal

Numbers are actually 1x1 Matrices

```
>> format short
>> x

x =

    3.1416

>> size(x)

ans =

     1     1
```

```
>> 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     5
```

Operations apply to matrices

```
>> 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
```

Operations apply to matrices

```
>> rowVector+secondVector

ans =

     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

Command Window

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

myMatrix =

     1     2     3
     4     5     6

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

mySecondMatrix =

     6     5     4
     3     2     1

>> myMatrix + mySecondMatrix

ans =

     7     7     7
     7     7     7

>> myMatrix * mySecondMatrix
Error using  *
Inner matrix dimensions must agree.

>> myMatrix .* mySecondMatrix

ans =

     6    10    12
    12    10     6
```

```
>> ans

ans =

     6    10    12
    12    10     6

>> disp(ans)
     6    10    12
    12    10     6
```

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

```
>> 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
```

```
ans =

     0     1     2     3
     4     5     6     7

>> find(ans)

ans =

     2
     3
     4
     5
     6
     7
     8
```

Concatenating matrices

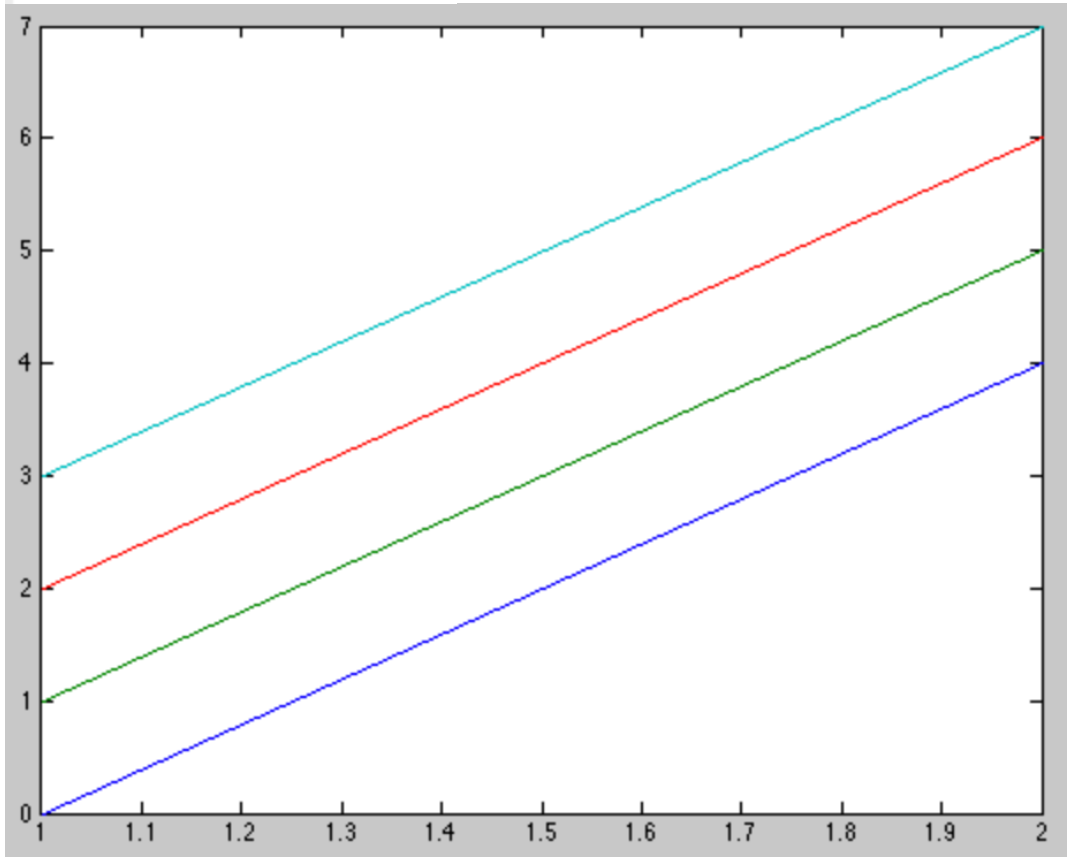
```
>> [myMatrix;myOtherMatrix]

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
```

```
>> plot(myBigMatrix)
```

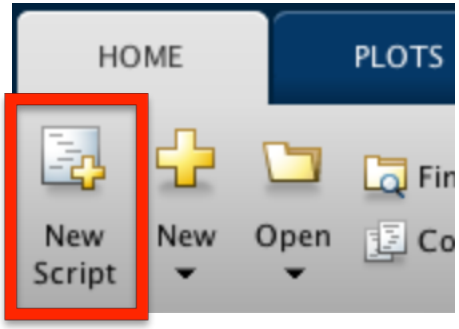


Working with m-files

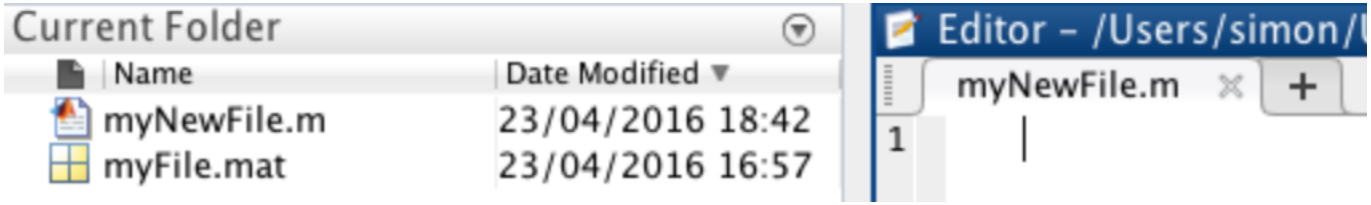
* 3 Ways of running code
  * Command Window ✓
  * Scripts
  * Functions

  } **m-files**

New Script




```
>> edit
>> edit myNewFile.m
```



Our First Script

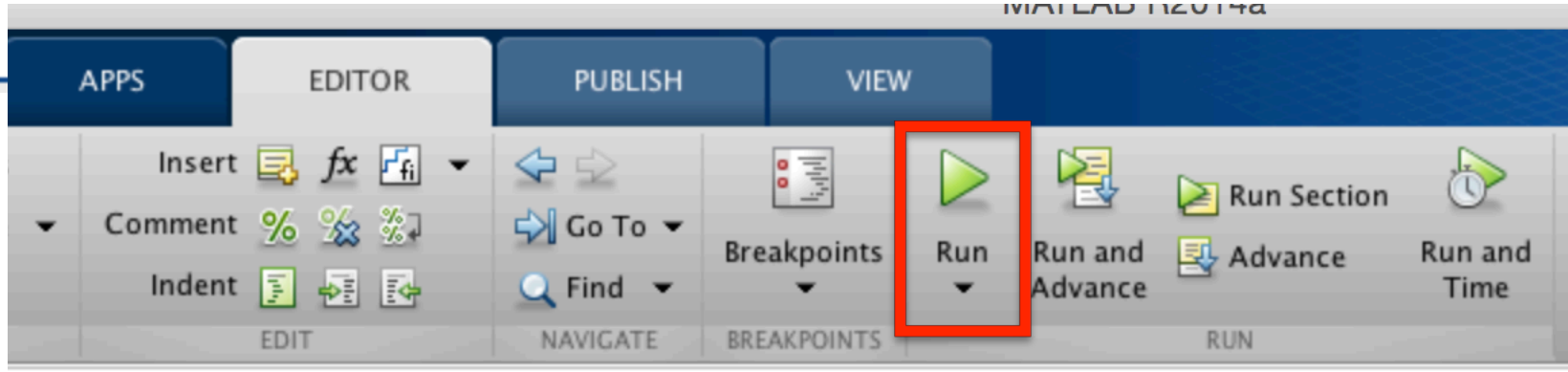
Editor - /Users/simon/Uni/MATLAB/Workshop/myNewFile.m

myNewFile.m

```
genesExp1 = 260;
genesExp2 = 58;
genesExp3 = 79;
totalGenes = genesExp1+genesExp2+genesExp3;
avgGenes=totalGenes/3;
disp(avgGenes);
```

Command Window

```
>> edit
>> edit myNewFile.m
>> myNewFile
  132.3333

>>
```



Scripts can create variables in the workspace

Current Folder

Name	Date Modified
myNewFile.m	23/04/2016 18:48
myNewFile.m~	23/04/2016 18:46
myFile.mat	23/04/2016 16:57

Details

Workspace

Name	Value	Min
avgGenes	132.3333	132.3333
genesExp1	260	260
genesExp2	58	58
genesExp3	79	79
totalGenes	397	397

Editor - /Users/simon/Uni/MATLAB/Workshop/myNewFile.m

myNewFile.m

```
genesExp1 = 260;
genesExp2 = 58;
genesExp3 = 79;
totalGenes = genesExp1+genesExp2+genesExp3;
avgGenes=totalGenes/3;
disp(avgGenes);
```

Command Window

```
>> clear
>> myNewFile
  132.3333
```

Data Types

* Type declarations are not necessary in MATLAB

* MATLAB automatically decides data type

Command Window

```
>> myVar = 7.3;
>> class(myVar)

ans =

double

>> myVar = 'test'

myVar =

test

>> class(myVar)

ans =

char

>>
```

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

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

Editor – /Users/simon/Uni/MATLAB/Work

myNewFile.m | dataTypes.m

```
geneName = 'nfkbia';
disp(geneName);
geneExp = 367.54323;

doubleVal = double(geneExp);
intVal = uint32(geneExp);

disp(doubleVal);
disp(intVal);

disp(isinteger(doubleVal));
disp(isinteger(intVal));
```

Command Window

```
>> dataTypes
nfkbia
  367.5432

       368

    0

    1

>>
```

Testing Data Types

Editor – /Users/simon/Uni/MATLAB/Workshop/dataTypes2.m

myNewFile.m | dataTypes.m | dataTypes2.m

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

Command Window

```
x =

     1     2     3


ans =

     0


ans =

     1


ans =

     1


ans =

     0
```

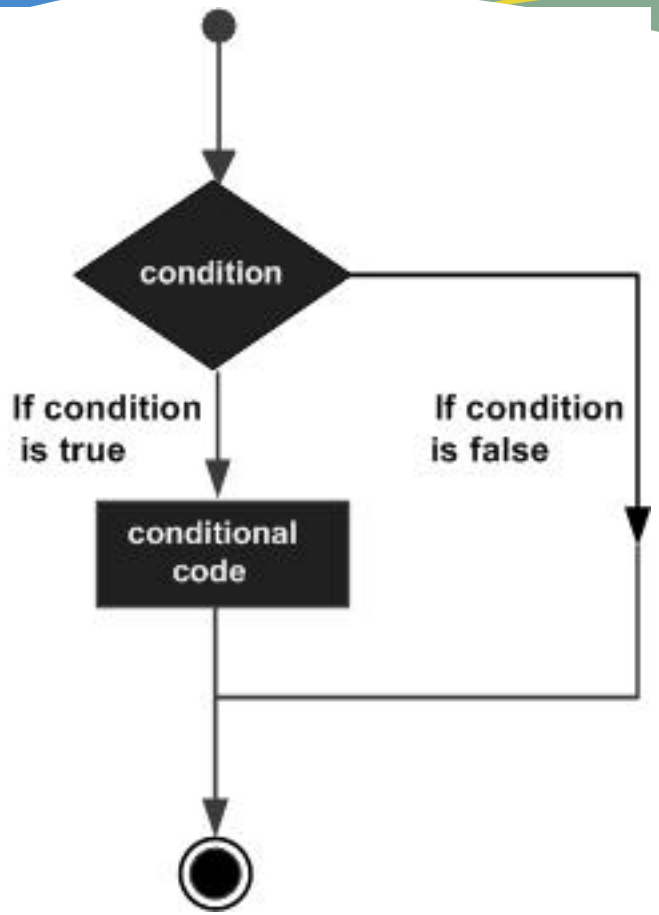
Relational Operators

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

```
>> x=6; y=9;
>> x>y

ans =

     0

>> x<=y

ans =

     1
```

Useful for if statements!

If Statement



Editor – /Users/simon/Un

ifStatement.m

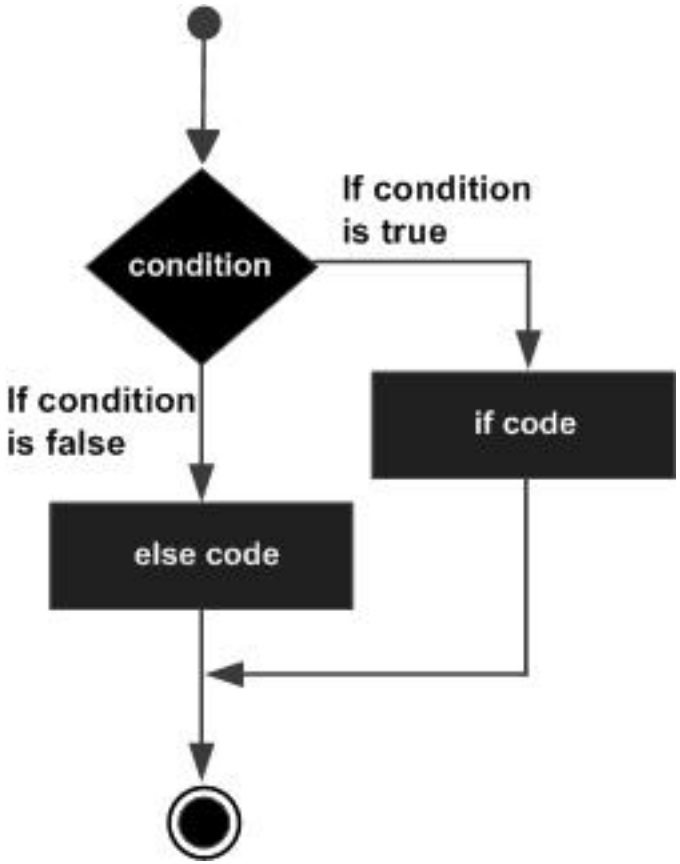
```
exp1 = 400;
exp2 = 500;
if (exp1 >= exp2)
    max = exp1
end
```

Never run

Command Window

```
>> ifStatement
```

If else Statement



Editor – /Users/simon/Uni/

myNewFile.m | dataType

```
exp1 = 400;
exp2 = 500;
if (exp1 >= exp2)
    max = exp1
else
    max = exp2
end
```

Command Window

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

max =

   500

>>
```

If elseif Statement

ifStatement.m

```
exp1 = 400;
exp2 = 400;
if (exp1 > exp2)
    max = exp1
elseif (exp1==exp2)
    max = 'Both are equal'
else
    max = exp2
end
```

Command Window

```
max =

Both are equal
```

Nested if Statement

Editor – /Users/simon/Uni/MATLAB/Workshop/ifStatem

ifStatement.m

```
exp1 = 440;
exp2 = 400;
if (exp1 > exp2)
    if ((exp1-exp2)>50)
        fprintf('exp1 is much bigger')
    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

Editor - /Users/simon/Uni/MATLAB/Workshop/forLoopExample

ifStatement.m | forLoops.m* | forLoopExample.m

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

Command Window

```
>> 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
>>
```

Combining What We've Learned

Write a **script** that takes a list of gene names and gene expression values and outputs only those gene names over a threshold.

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

PSUEDO CODE:

For each gene in a list:

if its expression value is over the threshold

print the gene name and the expression value

Editor – /Users/simon/Uni/MATLAB/Workshop/forLoops.m

ifStatement.m | forLoops.m | forLoopExample.m | +

```
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
```

Combining What We've Learned

Editor – /Users/simon/Uni/MATLAB/Workshop/forLoops.m

ifStatement.m | forLoops.m | forLoopExample.m

```
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
```

Command Window

```
>> edit forLoops
>> forLoops
Gene NFKBIA expression 56
Gene BCL2 expression 21
Gene CCL5 expression 39
Gene MYC expression 21
>>
```