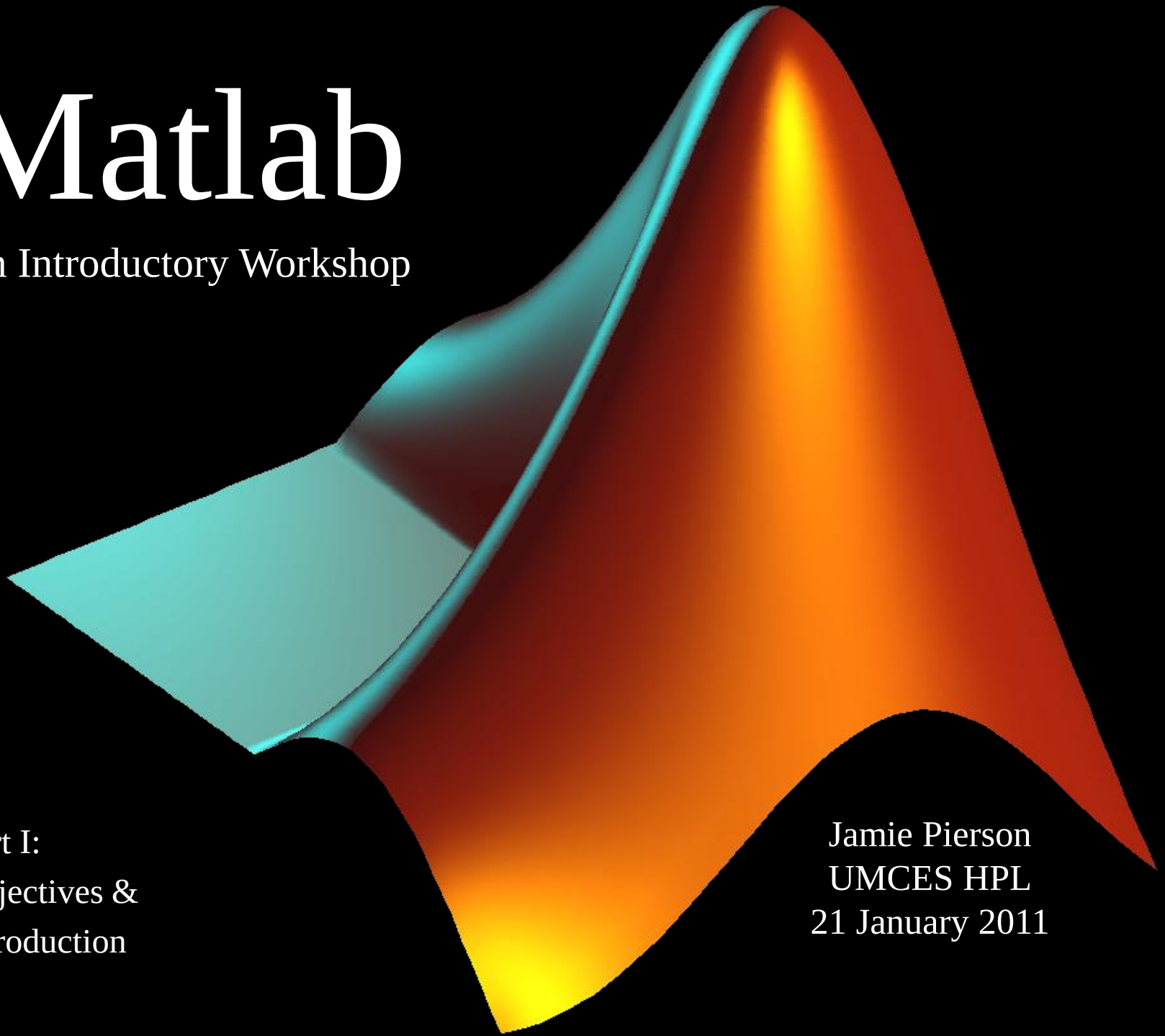
Matlab

An Introductory Workshop



Part I:
Objectives &
Introduction

Jamie Pierson
UMCES HPL
21 January 2011

Outline

1. Objective and Goals for today
2. Introduction to Matlab
3. Graphics
4. Statistics
5. Mapping
6. Matlab and Text (time permitting)

Objectives and Goals

After today you will

- Know what Matlab is and isn’t
- Be able to preform basic Matlab operations
- Have a set of tools to find help
- Have more questions than answers

Disclaimer 1

This is not a comprehensive course, it is an introduction with tips.

Matlab is HARD and will take a long time to master.

Matlab is incredibly versatile, powerful, and useful, so it’s worth the time and money investment.

Some suggestions

- Have a problem you need to use Matlab to solve – it makes learning easier when you have a goal.
- Find programs that have been written to do things you know or need then take them apart to understand them
- Use the tutorials and resources that come with Matlab.

Me?

I began using Matlab in 1995.

Really.

I am not a programmer, I don’t think like one, and I’m really glad there are programmers out there so I can do oceanography.

I can accomplish things in Matlab, but it’s not usually pretty.

Do as I say, not as I do.

You.

Demographics	
Student	15
FRA	6
Postdoc	3

Operating System	
PC	10
Mac	3
Linux	1

Experience	
None	5
Little	4
Some	16

Topic	
Plotting/Graphics	9
Stats	5
Mapping	4
Programming	4
Data Manipulation	3
Time Series (2) Solving Eq (1) Kriging (1)	

Why Matlab?

- Powerful computing environment
- Large online community
- Support from Mathworks
- Adaptability via toolboxes
- Interoperability with other languages and programs: e.g. maple, fortran, C++, Java, etc.

Free Alternatives

- SciLab
- Octave
- FreeMat

Why not use them?

No or very little support.

Not exact syntax.

Less stable.

Compatability.

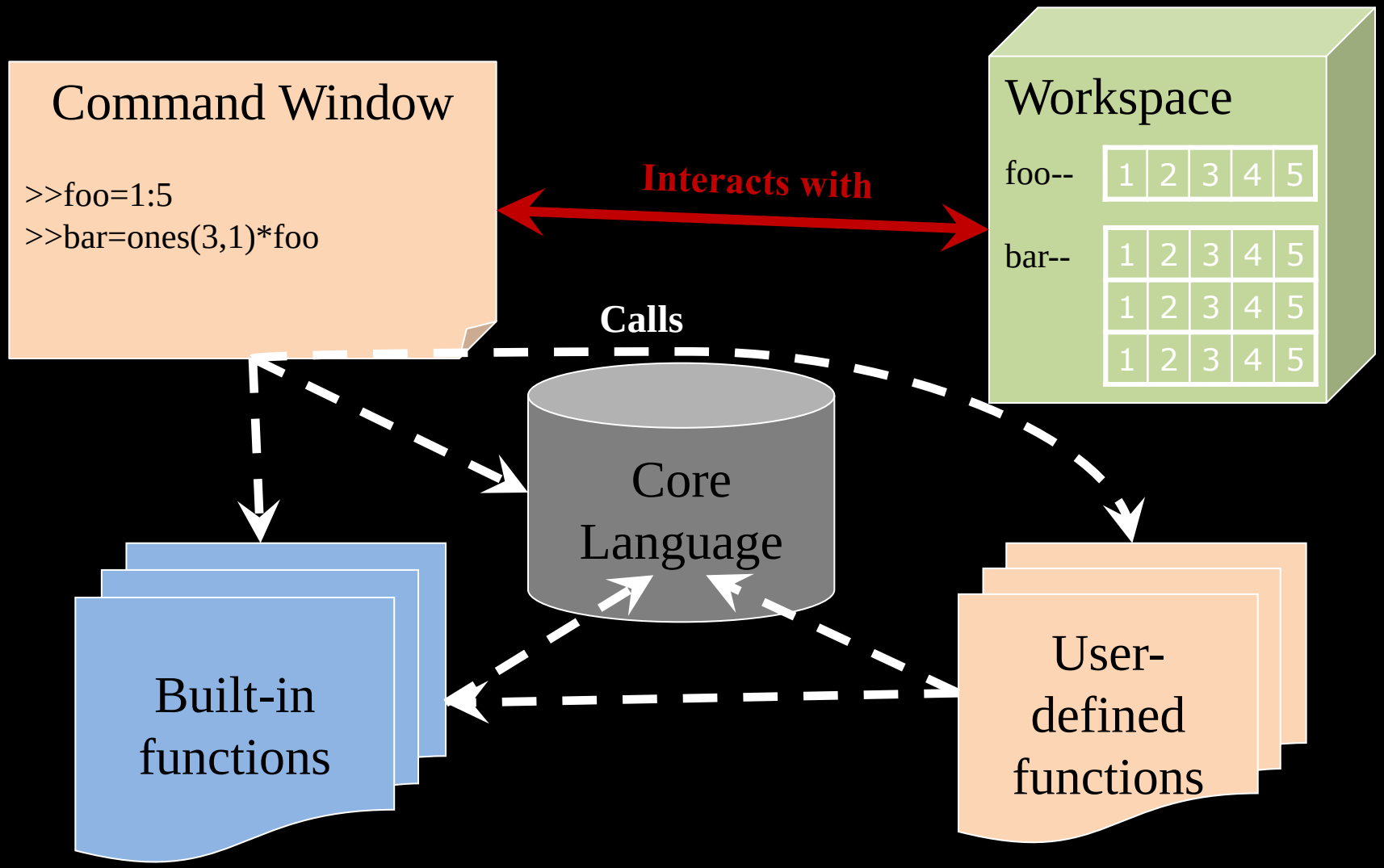
Legacy.

What is Matlab?

Matrix Laboratory

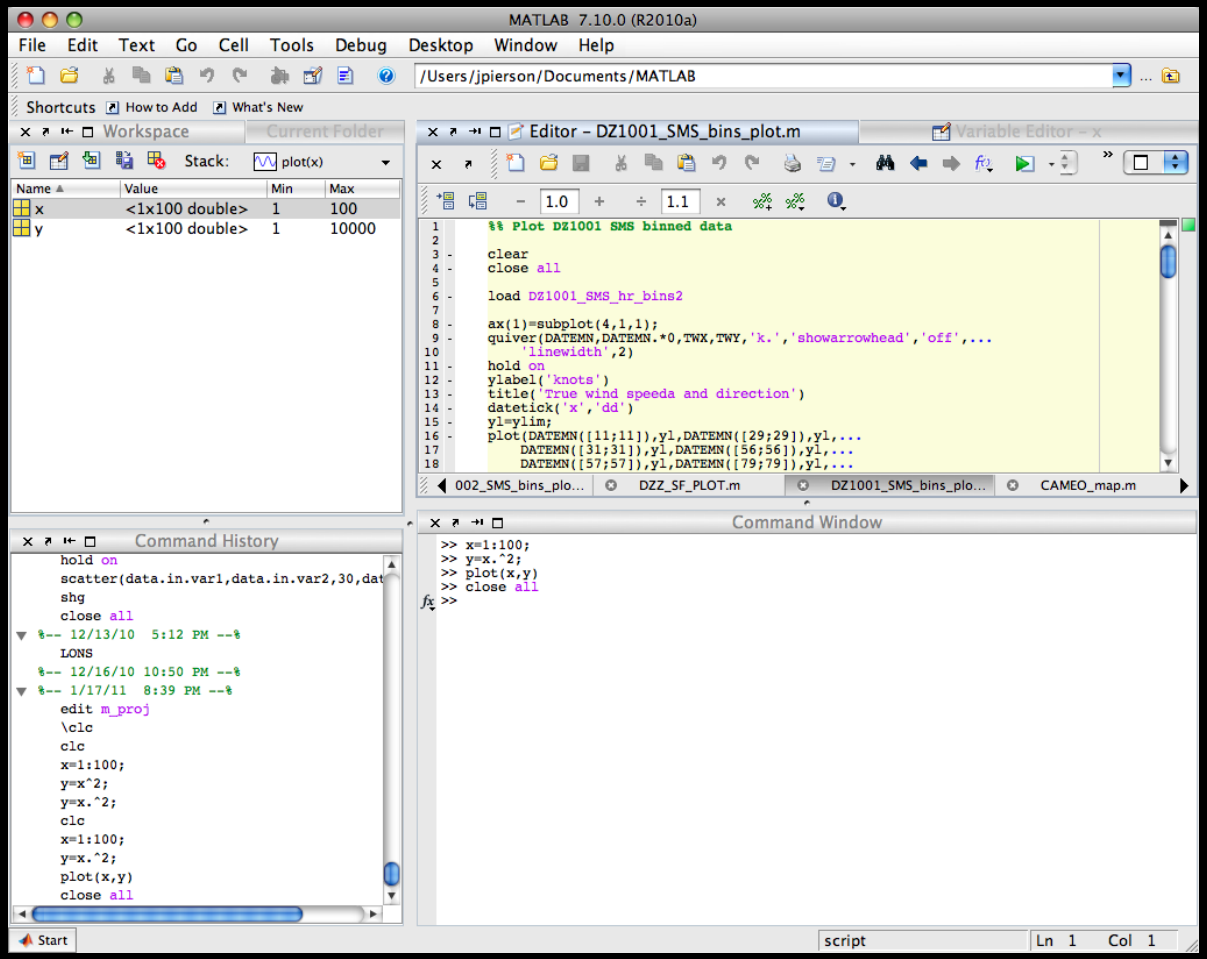
- Matlab is an environment for scientific computing
- Essential features:
  - Workspace--contains your data
    - named arrays of numbers
  - Language--a set of basic commands for manipulating data
    - assignment (=), dereferencing (()), arithmetic (+-*/^), logic (&|~), control (for,while, if-else, switch)
  - Functions--more complicated commands built with core language
    - mean, min, max, statistics, signal-processing, graphics,
    - YOUR STUFF!

What is Matlab?



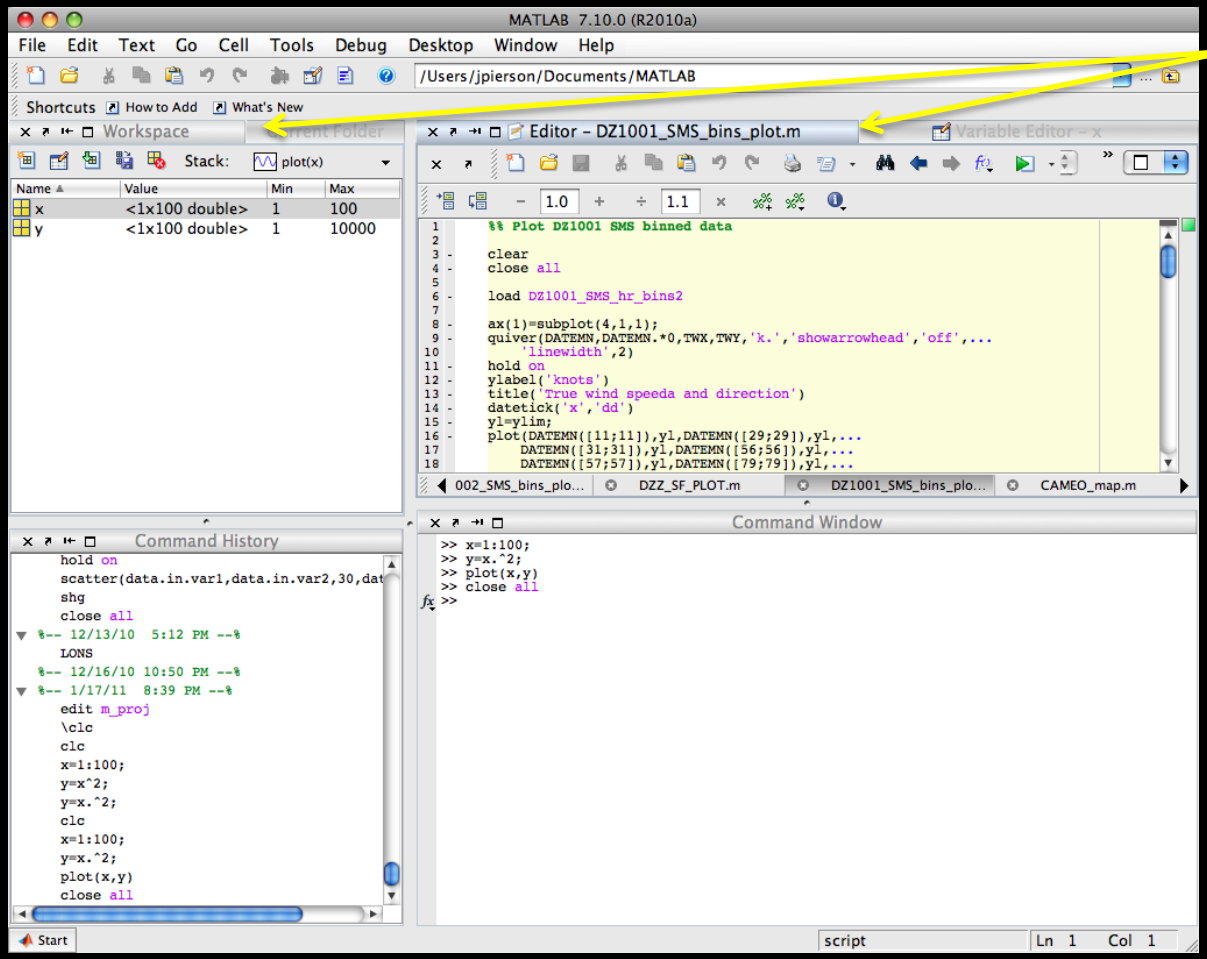
Adapted from A. Pershing 2010

The Matlab Window



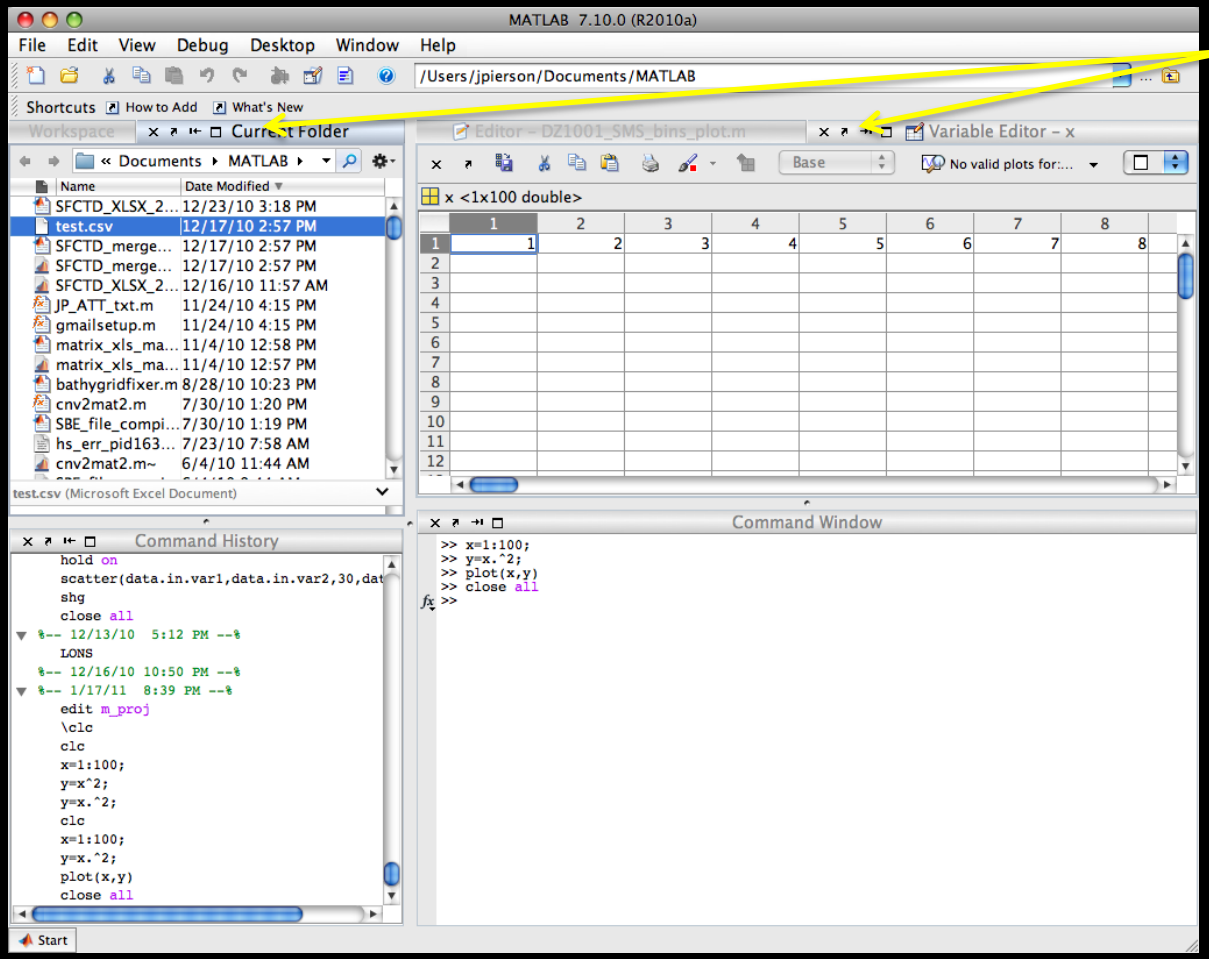
- Tabs
- Command window
- Workspace
- Variable Editor
- Editor
- Command History
- Working Directory

The Matlab Window



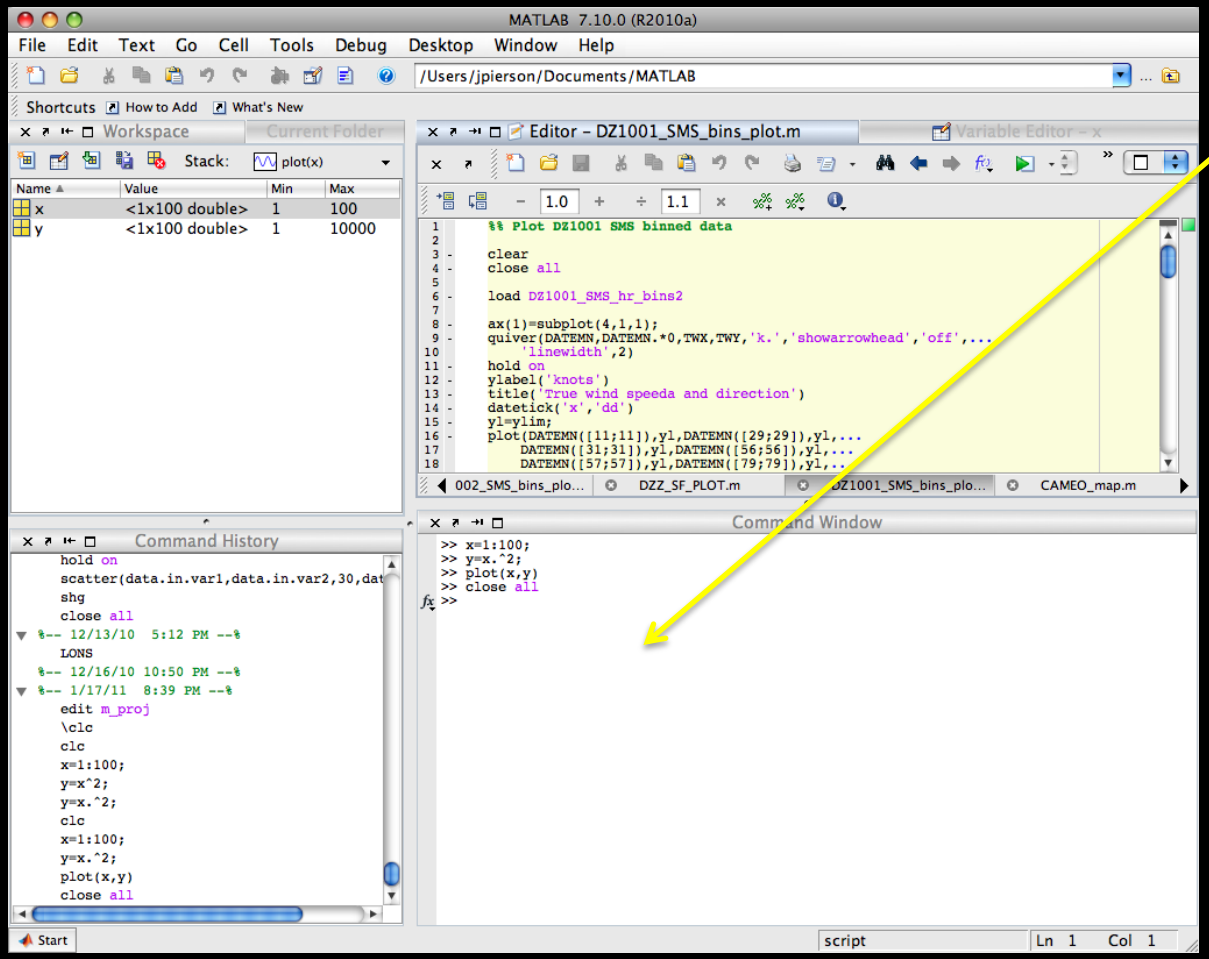
- **Tabs**
- Command window
- Workspace
- Variable Editor
- Editor
- Command History
- Working Directory

The Matlab Window



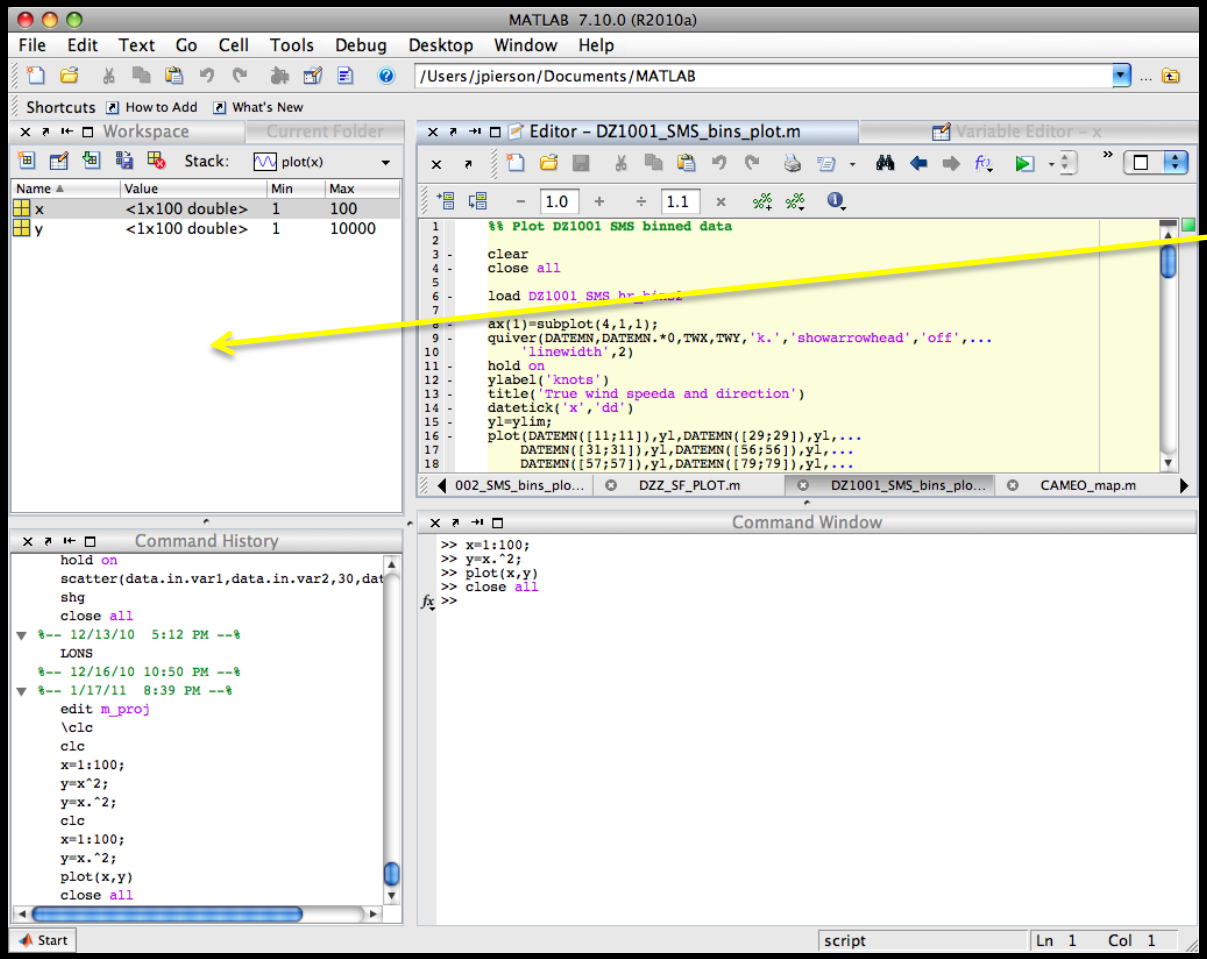
- **Tabs**
- Command window
- Workspace
- Variable Editor
- Editor
- Command History
- Working Directory

The Matlab Window



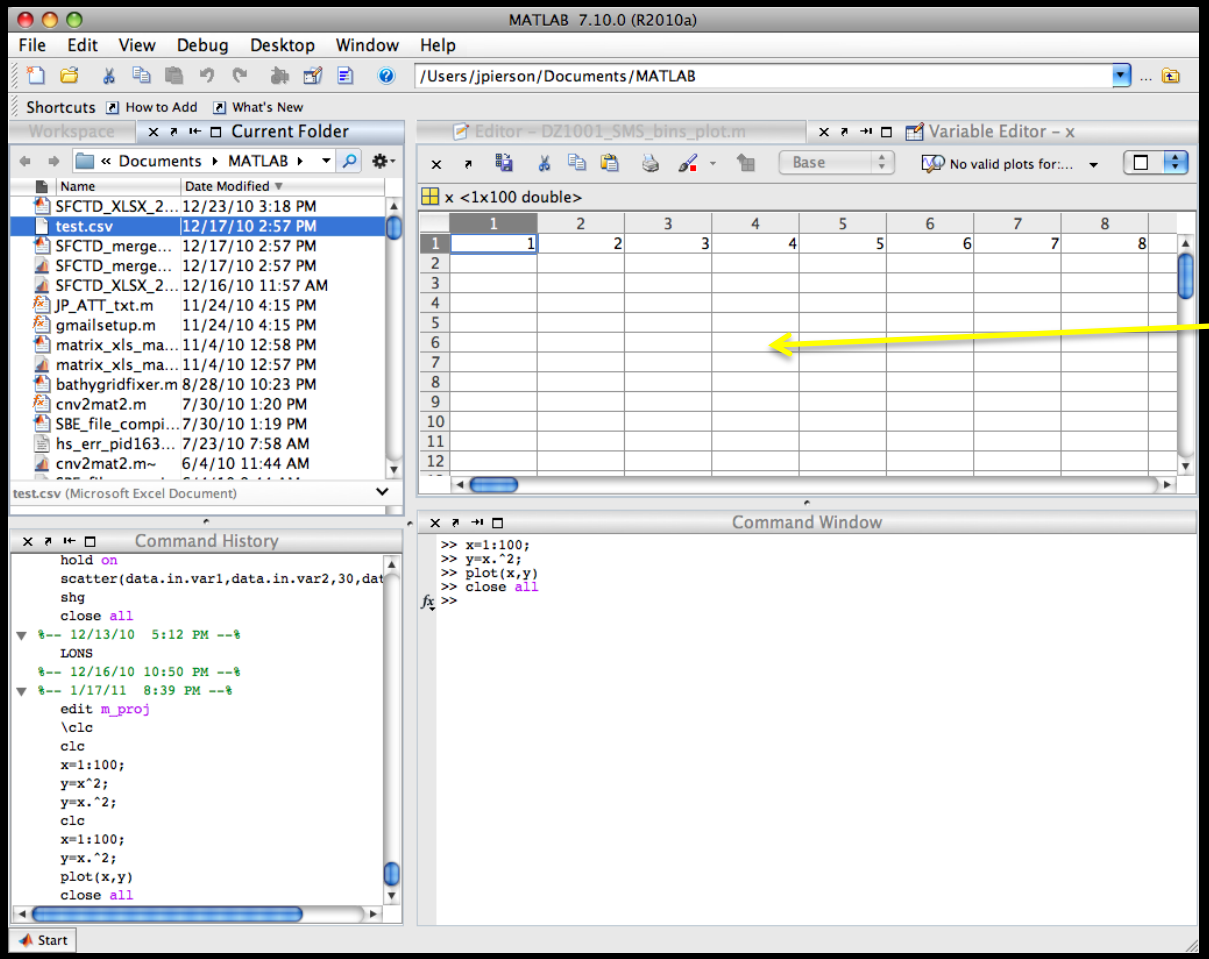
- Tabs
- **Command window**
- Workspace
- Variable Editor
- Editor
- Command History
- Working Directory

The Matlab Window



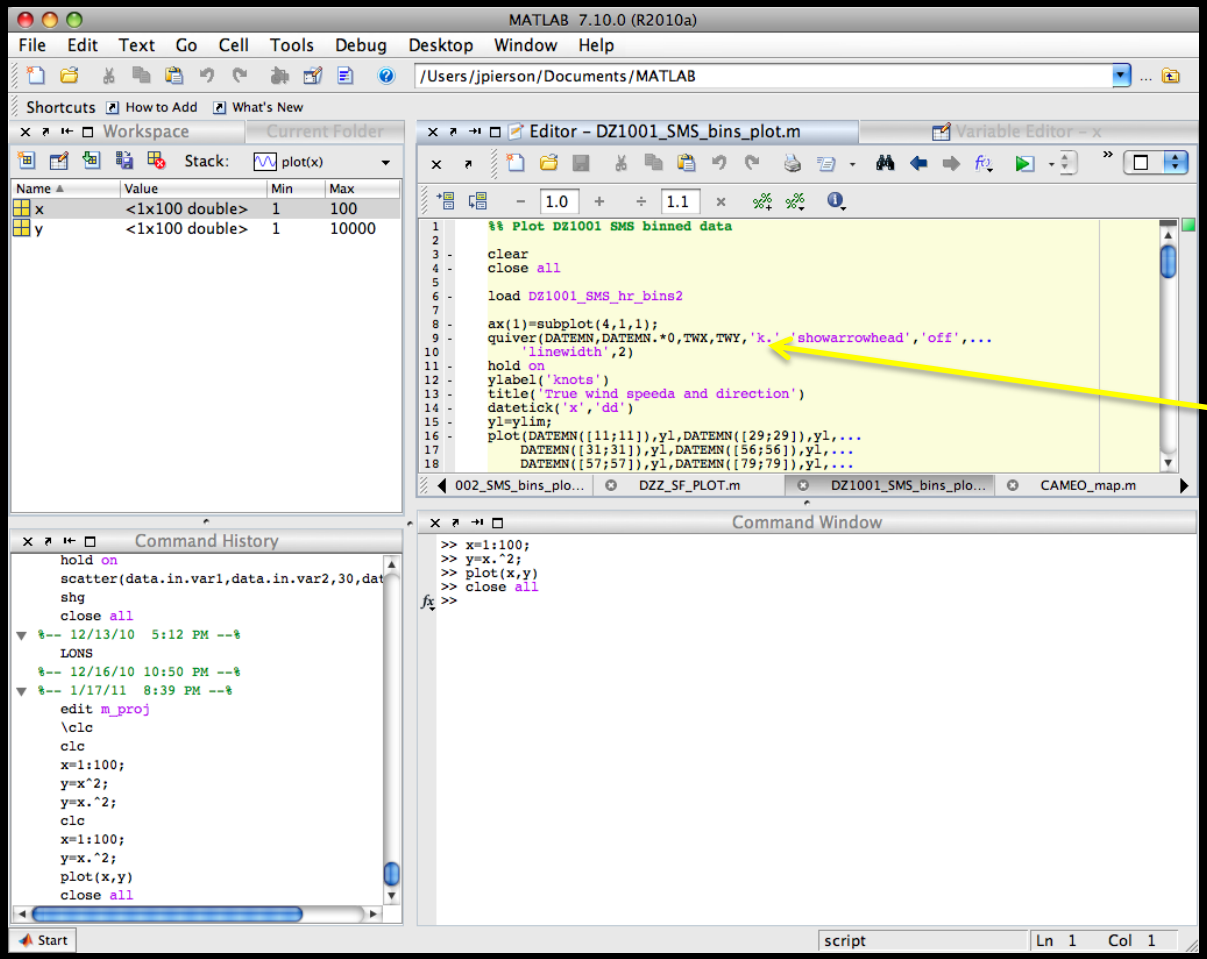
- Tabs
- Command window
- **Workspace**
- Variable Editor
- Editor
- Command History
- Working Directory

The Matlab Window



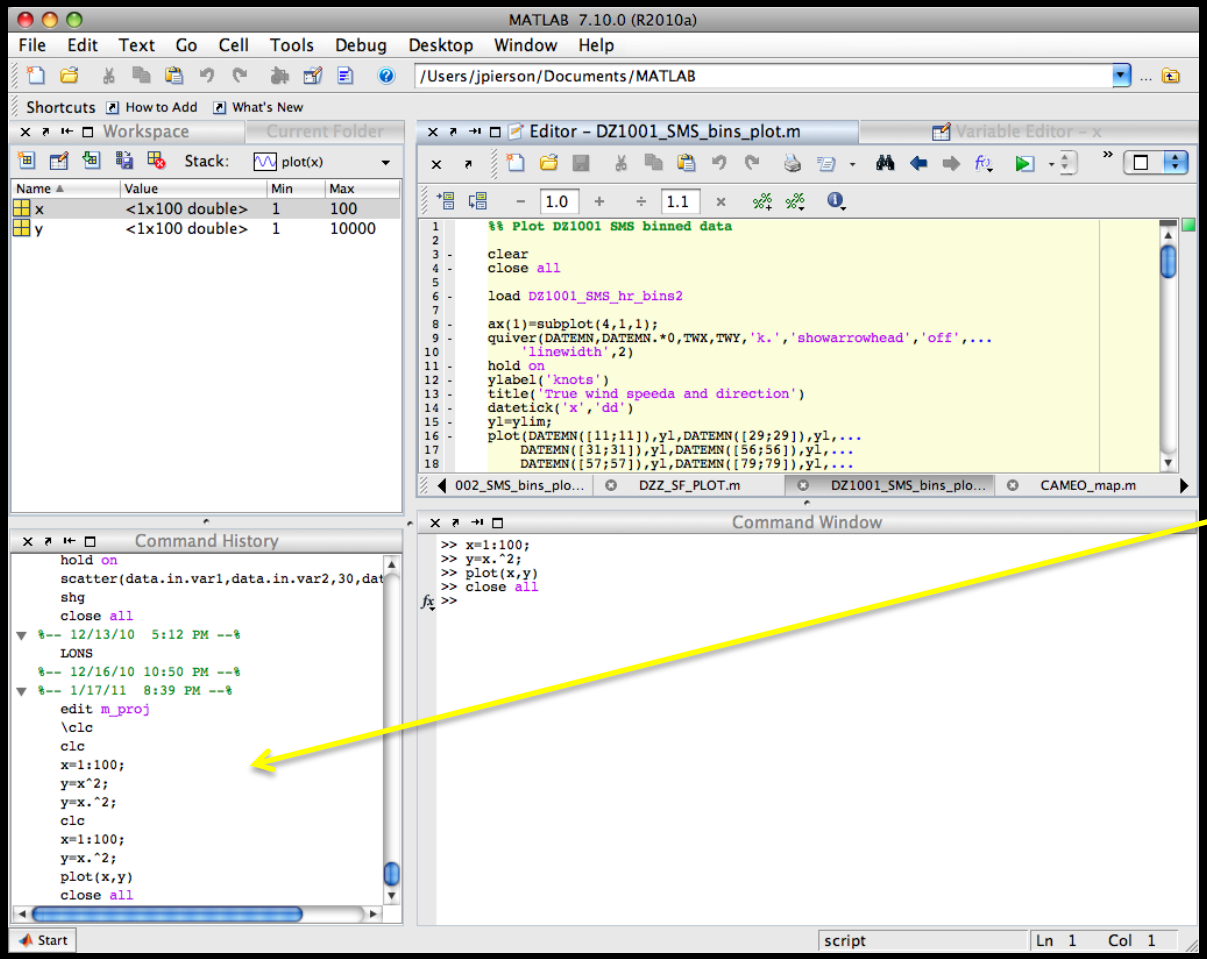
- Tabs
- Command window
- Workspace
- **Variable Editor**
- Editor
- Command History
- Working Directory

The Matlab Window



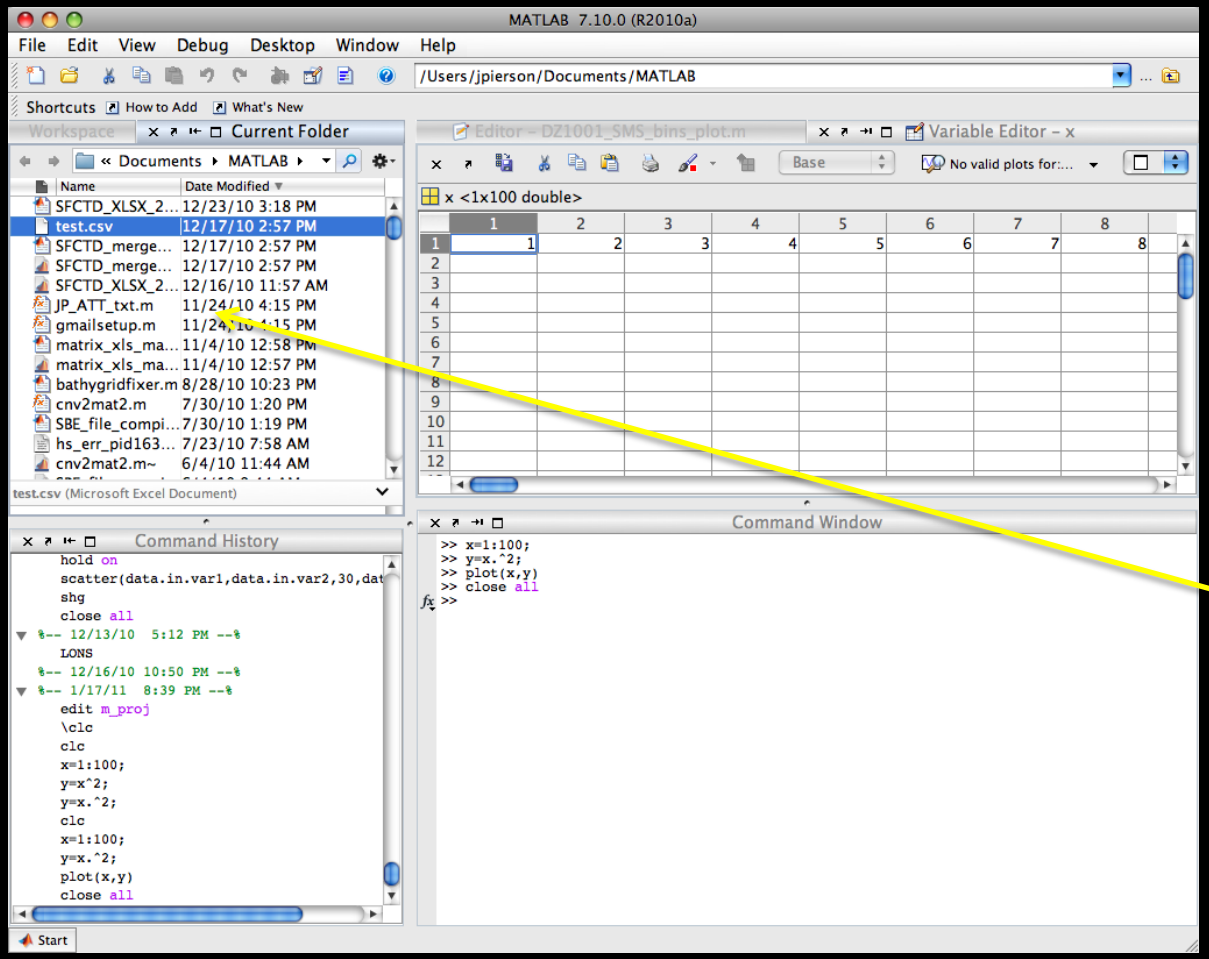
- Tabs
- Command window
- Workspace
- Variable Editor
- **Editor**
- Command History
- Working Directory

The Matlab Window



- Tabs
- Command window
- Workspace
- Variable Editor
- Editor
- **Command History**
- Working Directory

The Matlab Window



- Tabs
- Command window
- Workspace
- Variable Editor
- Editor
- Command History
- **Working Directory**

Some Syntax for Today and Matlab Help

Capitals denote 2D arrays:

X Y

Lowercase denotes 1D or vector arrays:

x y

Courier New font shows commands to type into Matlab at the Matlab Prompt*

```
>> plot(x,y);
>> pcolor(X,Y,Z);
```

* This is inconsistent in these slides, and I apologize, but I will try to fix it up before it goes online.

HELP!

1. `>> help [FUNCTION]`

will give you information about a function in the command window

2. `>> doc [FUNCTION]`

will open up the help browser with the same information from 1, but in a separate window.

Basic Math

- Matlab is a command-line calculator
  - Simple arithmetic operators
    - + - * / ^ \
  - Basic functions
    - sin(), log(), log10(), exp(), rem()
  - Constants
    - pi
  - Blanks etc.
    - NaN: Not a Number
    - Inf: infinity

Command History

- Matlab remembers your commands
- Use ↑↓ to go back/forward in time
- Type the first few letters of command , then use ↑ to recall commands beginning with that letter

Workspace

- Contains your data (variables)
- “who” lists variables
- “whos” lists variables + details
- Use “clear” to delete variables
  - clear deletes everything!
  - clear A B deletes A and B
  - clear –except A … deletes everything except A

  - *Use “help clear” or “doc clear” to find out more!*

1D Arrays--aka Vectors

- An array is anything you access with a subscript
- 1D arrays are also known as “vectors”
- Everything (nearly) in Matlab is a “double array”
- Create arrays with brackets [ ]
- Separate elements in a row with commas or spaces, separate rows with semicolon
- Access with ()’s

Regular arrays

- Create arrays using ":"
  - ***A=st:en produces [st, st+1, st+2, … en]***

```
>> A=1 : 5
A =
    1    2    3    4    5
>>A=-3.5 : 2
A =
   -3.5    -2.5  -1.5  -0.5  0.5  1.5
```

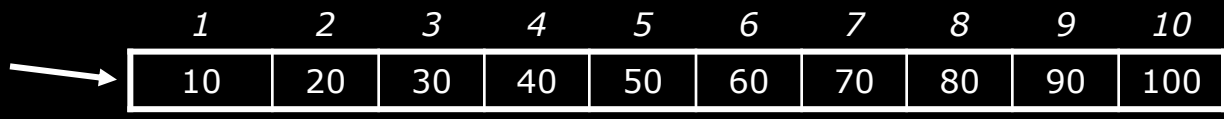
  - ***Can also insert a "step" size***

```
>> A=0 : 2 : 6
A =
     0    2    4    6

>> A=5 : -2.5 : 0
A =
     5    2.5   0
```

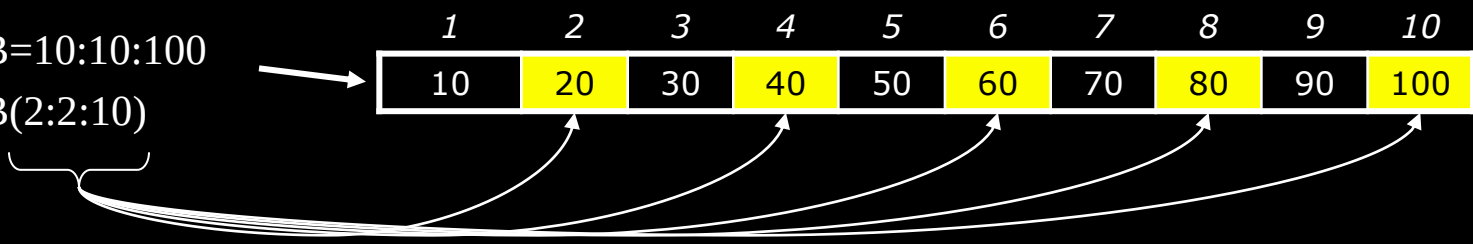
Accessing vectors

- Matlab arrays start at 1
- In most languages (C, Java, F77) can only access arrays one element at a time:
  - a(1)=1; a(2)=2.5; a(3)=-3; etc.
- In Matlab, can access several elements at a time using an array of integers (aka an *index*)
  - Ex:
    - B=10:10:100
    - B(2:2:10)



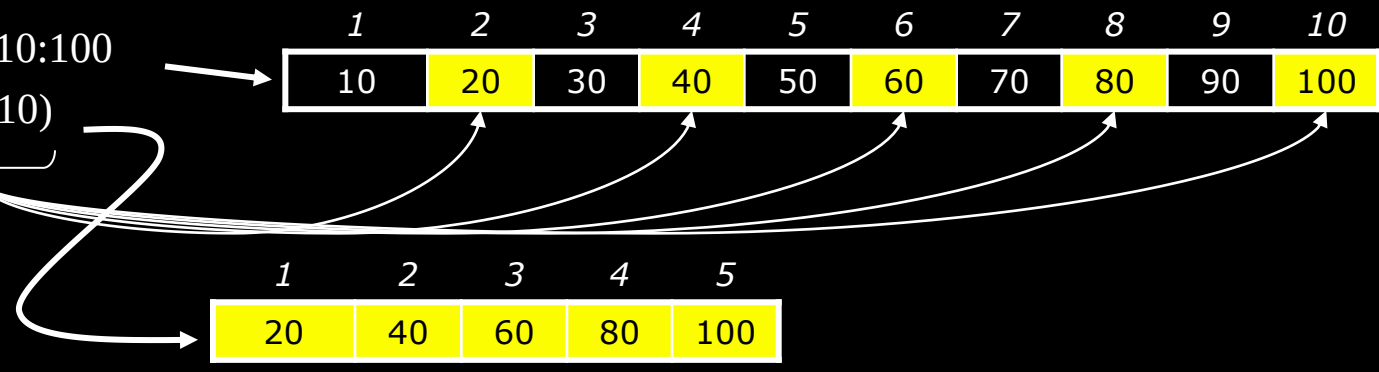
Accessing vectors

- Matlab arrays start at 1
- In most languages (C, Java, F77) can only access arrays one element at a time:
  - a(1)=1; a(2)=2.5; a(3)=-3; etc.
- In Matlab, can access several elements at a time using an array of integers (aka an *index*)
  - Ex:
    - B=10:10:100
    - B(2:2:10)



Accessing vectors

- Matlab arrays start at 1
- In most languages (C, Java, F77) can only access arrays one element at a time:
  - a(1)=1; a(2)=2.5; a(3)=-3; etc.
- In Matlab, can access several elements at a time using an array of integers (aka an *index*)
  - Ex:
    - B=10:10:100
    - B(2:2:10)



Accessing vectors

- Index vectors can be variables:

```
>>B=10:10:100;
>>I=2:2:10;
>>B(I)
   [20,40,60,80,100];
>>J=[1:2:9];
>>B(J)
   [10,30,50,70,90]
```

  - *What does B(I)=B(J) do?*

Column vectors

- “row vectors” are 1-by-n
- “column vectors” are n-by-1
- Row/column distinction doesn’t exist in most languages, but VERY IMPORTANT in MATLAB
- Create column vectors with semi-colons
  - A=[1; 2; 3]
- Can force to column vector with (:)
  - A=1 : 3 is [1 2 3]
  - A(:) is

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Column vectors

- Convert column-to-row and back with transpose (’)
- Can access the same way as row vectors

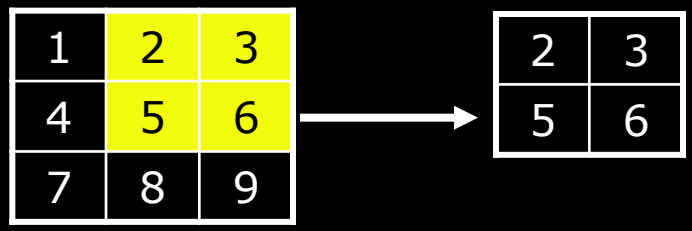
2D arrays--matrices

- From using commas/spaces and semi-colons
  - A=[1 2 3; 4 5 6; 7 8 9];
  - A(j,k)= j'th row, k'th column
    - A(1:2,2:3)

1	2	3
4	5	6
7	8	9

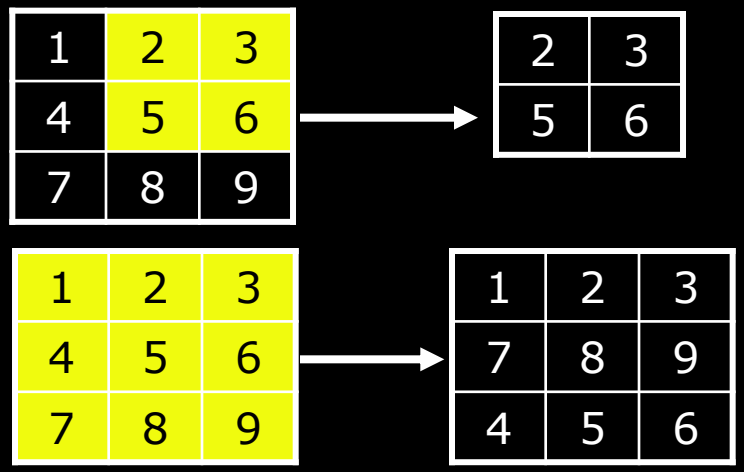
2D arrays--matrices

- From using commas/spaces and semi-colons
  - A=[1 2 3; 4 5 6; 7 8 9];
  - A(j,k)= j'th row, k'th column
    - A(1:2,2:3)= rows 1 through 2 and columns 2 through 3
    - A([1,3,2], :)



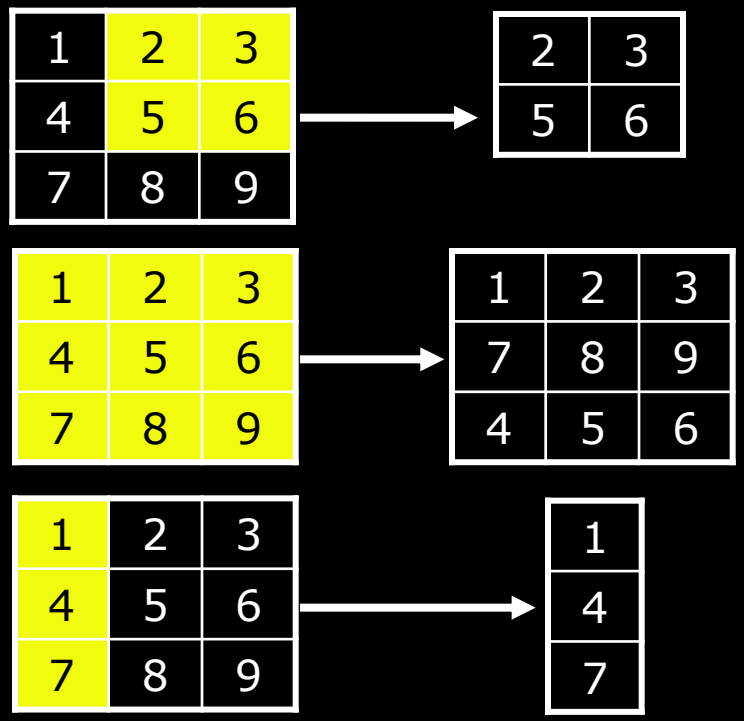
2D arrays--matrices

- From using commas/spaces and semi-colons
  - A=[1 2 3; 4 5 6; 7 8 9];
  - A(j,k)= j'th row, k'th column
    - A(1:2,2:3)= rows 1 through 2 and columns 2 through 3
    - A([1,3,2], :)= all of rows 1, 3 and 2
    - A(:, 1)



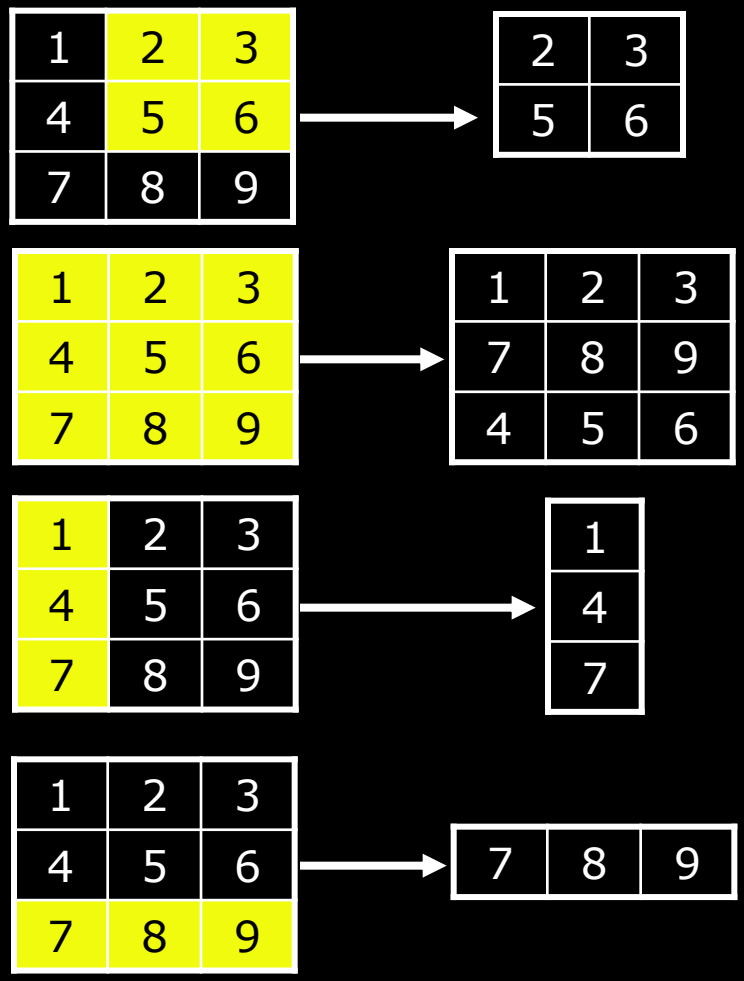
2D arrays--matrices

- From using commas/spaces and semi-colons
  - A=[1 2 3; 4 5 6; 7 8 9];
  - A(j,k)= j'th row, k'th column
    - A(1:2,2:3)= rows 1 through 2 and columns 2 through 3
    - A([1,3,2], :)= all of rows 1, 3 and 2
    - A(:, 1)= first column
    - A(3,:)=



2D arrays--matrices

- From using commas/spaces and semi-colons
  - A=[1 2 3; 4 5 6; 7 8 9];
  - A(j,k)= j'th row, k'th column
    - A(1:2,2:3)= rows 1 through 2 and columns 2 through 3
    - A([1,3,2], :)= all of rows 1, 3 and 2
    - A(:, 1)= first column
    - A(3,:)= last row



Size matters

- “A is m-by-n” means A has m rows and n columns
- [m,n]=size(A) gets size of A, storing sizes in scalars m and n
  - useful for error checking & getting limits on for-loops
- length(a) gets length of vectors (max of m and n).
- A(1:3,2)=v, v better be length 3 (or scalar)
- A(1:2:5,2:3)=B, B better be 3-by-2 (or scalar)

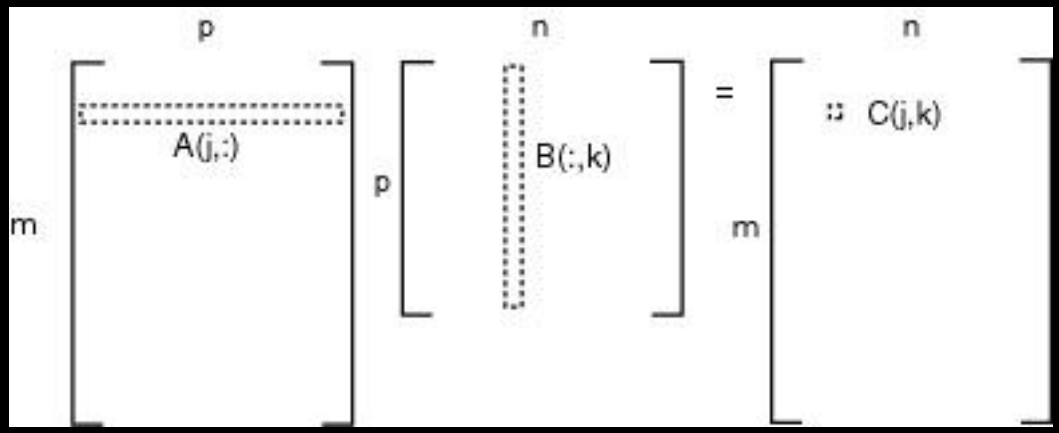
Array Arithmetic

- C=A+B
  - if A and B are the same size, C(j,k)=A(j,k)+B(j,k)
  - If A is a scalar, C(j,k)=A+B(j,k)
- Same for -

Array Multiplication

- Multiplication in Matlab is inherited from linear algebra
  - To multiply by a scalar, use *
  - To get C(j,k)=A(j,k)*B(j,k) use ".*"
    - Also applies to ".^" and "./"
  - Otherwise it uses matrix multiplication:

$$C(i,j) = \sum_{k=1}^{p} A(i,k) * B(k,j)$$



Adapted from A. Pershing 2010

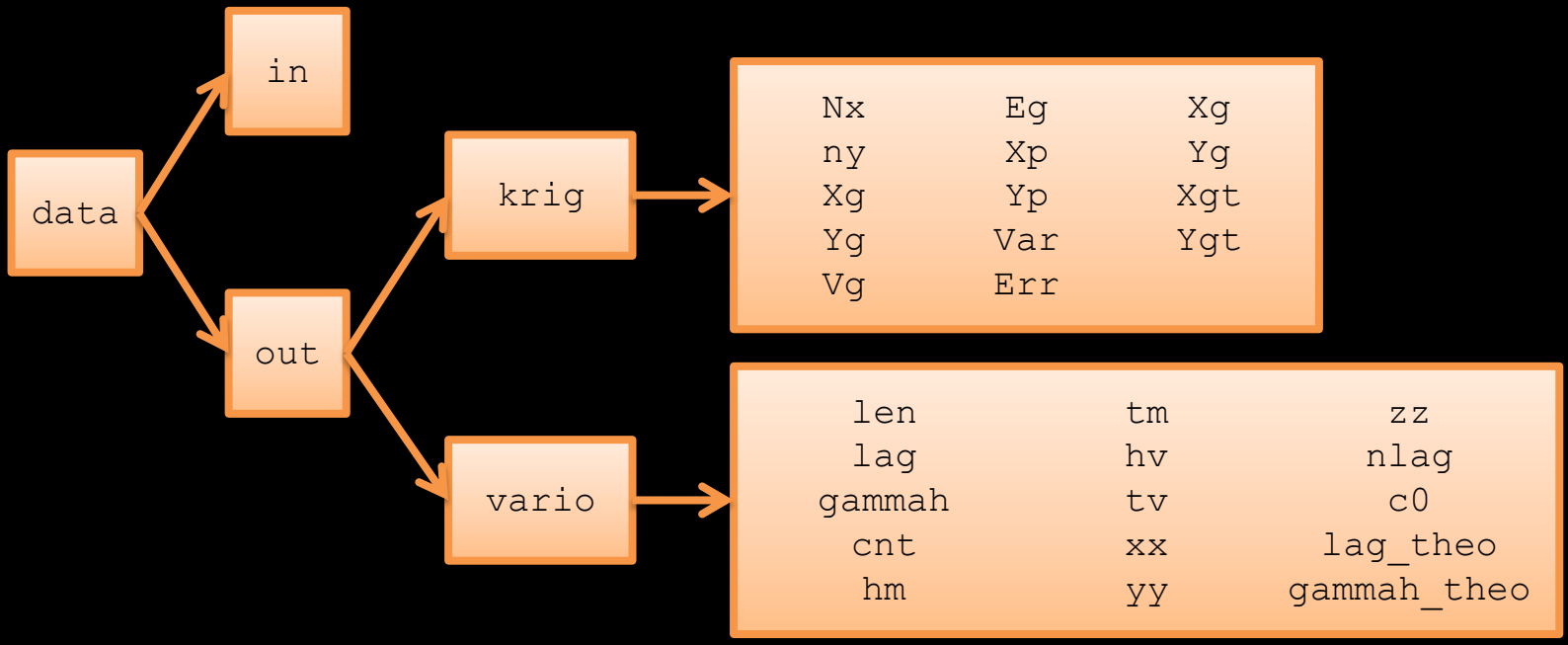
ND arrays

- Until V5, Matlab arrays could only be 2D
- Now has unlimited dimensions:
  - A=ones(2,3,2)
  - A is a 3D array of ones, with 2 rows, 3 columns, and 2 layers
  - A(:,:,1) is a 2-by-3 matrix

Structured Arrays

A variable that contains data in “fields”

Used in “EasyKrig” (just you wait) and some statistical functions, as well as for image analysis and other complex functions



Calling structured data

```
>> data.out.krig.Nx;
>> data.out.krig.Nx(1)
```