Matlab for Psychologists

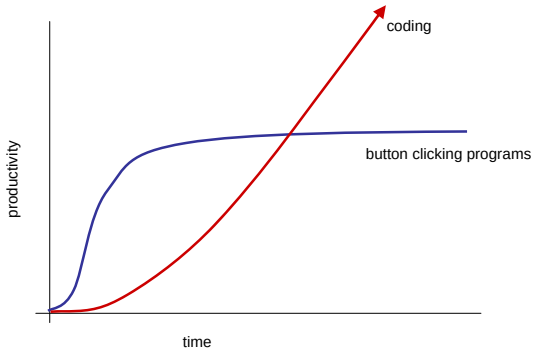
Introduction

Matlab is a language

- Simple rules for grammar
- Learn by using them
- There are many different ways to do each task
- Don't start from scratch - build on what other people have done and improve it
- It takes a long time to become fluent

- Use Matlab to generate stimuli and run experiments
- Analyse fMRI data in SPM
- Manipulate data, sound, graphics, video …

Coding v. button clicking



Overview

7 weeks

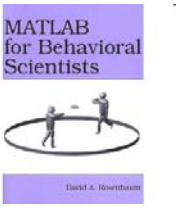
- Week 1 – basics
- Week 2 – indexing, scripts and flow control
- Week 3 – functions and graphics
- Week 4 – running experiments and manipulating text files
- Week 5 – starting on SPM
- Week 6 – SPM analysis
- Week 7 – more SPM

Other resources

Matlab tutorial on my website

www.antoniahamilton.com

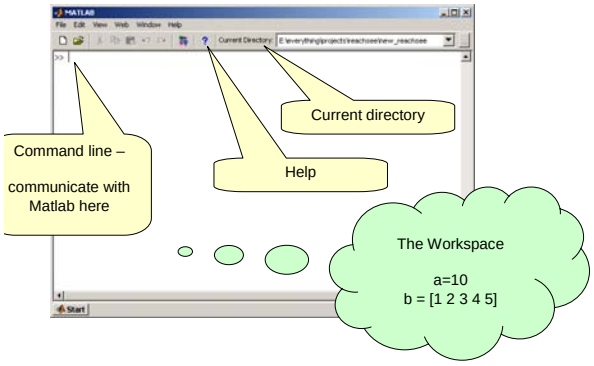
David Rosenbaums' book



The Plan

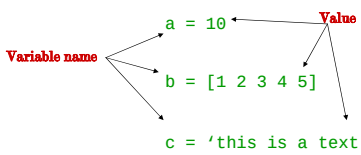
- Basic Grammar
  - operators
  - manipulating matrices
- Sentences
  - Scripts
  - Saving and Loading
  - Graphs
- Real World Examples

Hello, nice to meet you



Variables

Matlab stores data in Variables



Variable name rules:

- must start with a letter
- no spaces
- no punctuation
- up to 32 characters
- case sensitive

You can create a variable by typing at the command line

```
a = 10   %% creates the variable a with value 10
```

Or you can change a variable

```
a = 20   %% a now has the value of 20
```

Variables let you store and manipulate all your data in Matlab

Some types of variable

Scalars ➔ one number

```
a = 10
```

Vectors ➔ list of numbers (row or column)

```
b = [6 3 7 4]
e = [5
     8
     4]
```

Matrix ➔ grid of numbers

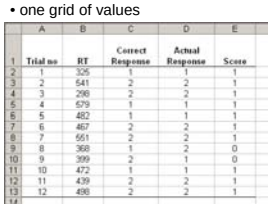
```
c =  [10 40 73
      43 95 28]
```

String ➔ text

```
d = 'this is a string'
```

Compare to Excel

- one grid of values



	A	B	C	D	E
1	Trial no	RT	Correct Response	Actual Response	Score
2	1	325	1	1	1
3	2	541	2	2	1
4	3	298	2	2	1
5	4	579	1	1	1
6	5	482	1	1	1
7	6	467	2	2	1
8	7	551	2	2	1
9	8	368	1	2	0
10	9	399	2	1	0
11	10	472	1	1	1
12	11	439	2	2	1
13	12	486	2	2	1

Finding out about variables

`whos`

➢ shows you what your variables are and how big they are

`size(A)`

➢ tells you how big matrix `A` is

`length(A)`

➢ tells you the biggest dimension of variable `A`

`ndims(A)`

➢ tells you how many dimensions variable `A` has

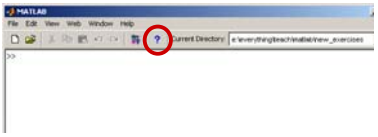
Using Help

Matlab gives you help on every function at the command line

Type

```
>> help length
```

Matlab documentation



Matlab website

http://www.mathworks.com/

Creating variables [ ] , ;

Use Square Brackets [ ] to create a vector or matrix variable

AND to join together two variables

Use Commas , to separate items on the same row

Use Semi-colon ; to start a new row

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

b = [5; 6; 7]

c = [3, 4, 5, 6; 9, 8, 7]
```

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

```
>> b = [5; 6; 7]
b =
     5
     6
     7
>>
```

```
>> c = [3, 4, 5, 6; 9, 8, 7]
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

Getting data out of a variable ( )

Matrix variables are arranged in a grid

```
a = 35     1     6    26    19
     3    32     7    21    23
    31     9     2    22    27
     8    28    33    17    10
    30     5    34    12    14
```

We reference items within a matrix by rows, then by columns

Use round brackets () to get things out of a matrix

`a(2,4)` ➔ a, row 2, column 4 ➔ 21

`a(5,1)` ➔?

`a(3,2)` ➔?

`a(6,1)` ➔? Index exceeds matrix dimensions

Getting more out :

Use a colon : to get a whole row or column out of a matrix

```
a = 35     1     6    26    19
     3    32     7    21    23
    31     9     2    22    27
     8    28    33    17    10
    30     5    34    12    14
```

Again, we use () and reference rows and columns

Read a colon as 'everything'

```
a(2,:) ➔ a, row 2 everything ➔ 3   32    7   21   23
a(:,1) ➔ a, everything in column 1 ➔
a(3,:) ➔  ?
a(:,5) ➔  ?

a(:) ➔  ?
```

Special case = everything in a in one long column

Changing variables

We can also change the values within a variable.

```
a = 35     1     6
     3    32     7

a(2,3) = 10  ➔ ?
```

And we can join two variables

```
b =  1     5     8
     4     9     0

c = [a, b] ➔    35   1    6   1   5   8
                 3  32    7   4   9   0

c = [a; b] ➔
```

Beyond the end

a is a 2x3 matrix

```
a = 12  2   70
    33  64  21
```

```
a(4,1)
```

➔ Index exceeds matrix dimensions

```
a(4,1) = 20

a = 12  2   70
    33  64  21
     0   0   0
    20   0   0
```

Matlab expands a to include the new value

Quick Exercise

```
a = 1  2
    3  4
```

```
b = 7  8
    9  0
```

1) value of `a(2,1)`

2) value of `b(2,2)`

3) `a(1,1) = 5`, now a = ?

4) `c = [a,a]`

5) `d = [4 5 6; 7 8 9]`

6) `b(:,2)`

7) `a(2,:)`

8) `[a(:,1); b(:,1)]`

Commands

You use commands to tell Matlab to do things

e.g. to ask 'what variables are in the workspac

```
>> whos
```

Matlab replies

Name	Size	Bytes	Class
a	1x1	8	double
b	5x5	200	double

To clear a variable from the workspace

```
>> clear a
```

To clear all variables from the workspace

```
>> clear all
```

Maths

```
a = 1  2
    3  4
```

```
b = 7  8
    9  0
```

```
a + b  ➔    8  10      add every element of a and b
           12   4
a – b  ➔ subtract every element of a and b
a – 10 ➔ subtract 10 from every element of a
```

a and b MUST be the same size

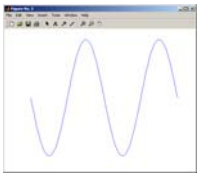
```
BUT
a * b ➔ matrix multiplication of a and b
a .* b ➔ multiply each element of a and b
Similarly
a / b ➔ matrix division
a ./b ➔ divide each element
More advanced
a' ➔ a transpose ➔ convert rows to columns
a.^2 ➔  a squared ➔ a .* a
Combine maths with round brackets
(a + b).* 6
```

```
a' = 1  3
     2  4
```

Mathematical commands

Matlab lets you do all the kinds of maths you might expect

```
sin(a)
cos(a)
tan(a)
log(a)
```



```
floor(a)  ➔ round downwards
ceil(a) ➔ round upwards
round(a) ➔ round to nearest integer
rem(a)➔ remainder after division
abs(a)➔ convert negative values to po
```

All these commands act on each element of a matrix individually

`>> help elfun` for full details

Summarising commands

```
a = 1  2
    3  4
    5  6
   10  0
```

Many commands combine perform a calculation over all the rows or all the columns of a matrix

e.g.

`min(a)` ➔ find the lowest value in each column
➔ `[1 0]`

`max(a)` ➔ find the highest value in each column

`mean(a)` ➔ average the values in each column

`sum(a)` ➔ add up the values in each column

`std(a)` ➔ standard deviation of each column

You can store the output of any command in a new variable.

e.g.

```
ma = mean(a)
```

`>> help stats` for full details

Generating variables

Often, it is useful to have matlab generate some variables for you

•a matrix full of zeros ➔ `a = zeros(3,4)`

```
a = [0 0 0 0
     0 0 0 0
     0 0 0 0]
```

•a matrix full of ones ➔ `b = ones(2,3)`

```
b = [1 1 1
     1 1 1]
```

•a matrix full of some other number ➔
`c = ones(4,2)*5`

```
c = [5 5
     5 5
     5 5
     5 5]
```

Generating sequences

The colon : operator lets you generate a sequence of values

•a sequence of numbers ➔ `a = 1:5` ➔ `a = [1 2 3 4 5]`
(read this as `a` gets the values from one to five)

•a sequence with bigger steps
➔ `a = 1:3:10` ➔ `a = [1  4  7  10]`

start, step size, end

If you know how many items you want, use linspace

start, end, number of elements

•a linearly spaced sequence ➔ `a = linspace(0,10,4)`
➔ `a= [0  3.3333  6.6667  10.0000]`

Re-arranging things

Reshape lets you change the shape of a matrix. The number of elements must stay the same

```
a = [16   2   3  13
      5  11  10   8
      9   7   6  12
      4  14  15   1]
```

rows, columns

```
b = reshape(a,2,8)
```

```
b = [16   9   2   7   3   6  13  12
      5   4  11  14  10  15   8   1]
```

Quick Exercise

```
a = 4  5
    3  6
    2  7
    1  0
```

```
b = 3  8
    5  4
    7  6
    9  2
```

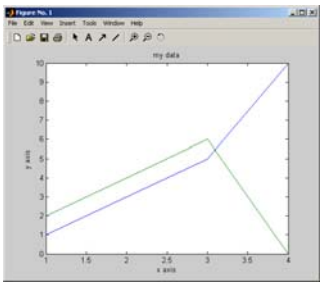
1) `a + b`
2) `mean(b)`
3) create a 4 by 2 matrix `c` where every element is 2
4) `a./c`
5) create a matrix `d` which joins `a` and `b` one above another
6) find the standard deviation of the columns of `d`
7) add up the rows of `a`
8) create a vector `e` with 4 elements, equally spaced from 0 to 1
9) create a vector `f` with 4 elements all with the value 20
10) multiply the elements of `e` and `f`
11) create a 5 by 6 matrix `g` of the values from 1 to 30 in order
12) find the biggest value in each column of `g`

Commands to make graphs

```
a = 1  2
    3  4
    5  6
   10  0
```

Visualising your data is very important, and matlab gives you many tools for visualisation

```
>> plot(a)
```



Data columns plotted
Colors / axes chosen by default

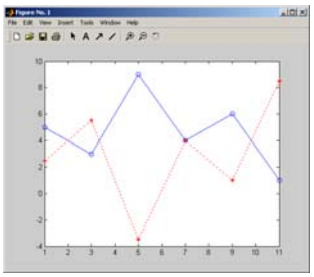
```
>>title('my data')
>>xlabel('x axis')
>>ylabel('y axis')
```

Controlling your graph

plot has lots of options to customise your graph

```
plot(x,y,'b-o')
```

data for x axis — data for y axis — line specification



```
hold on
plot(x,z,'r:*')
```

```
>> help plot
```
for full details

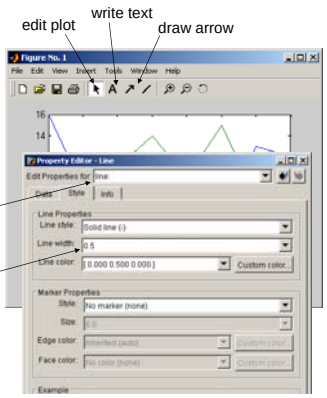
Controlling your graph with clicks

You can edit a graph with the GUI ➔
You can change the properties of any item with the Property Editor (Edit – Figure Properties)

Select the item at the top

Then change its properties

We will learn later how to do this in code



Other types of graph

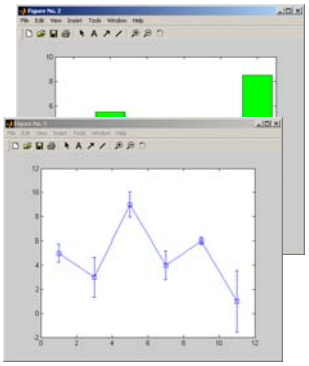
Matlab can do bar plots

```
figure(2)
bar(x,z,'g')
```

And errors bars

```
figure(3)
errorbar(x,y,ee,'b-s')
```

We will learn more graphics in week 4



Cleaning up

When you are dealing with lots of variables, it is easy to get in a muddle.
To see what variables you have, use `whos`

To remove a variable, use `clear`

To remove all variables, use `clear all`

To close graphs, use `close(n)` where n is the figure number or `close all`

Before starting a new exercise, always do `clear all`

Loading and saving data

Data can be saved in .mat files
save using

```
save filename variables
```

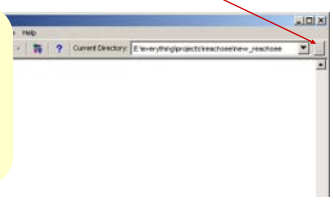
e.g. `save mydata.mat a b c`

load using

```
load filename
```

To see your files
`ls` OR `dir`

change directory
`cd` OR



WARNING:

windows always thinks .mat files (Matlab data files) are Microsoft Access Tables.
They aren't.
Always open your .mat files by typing load in Matlab, NOT by double clicking in Windows.

Exerise One - Introductions

1) Enter the matrices **a b c**
2) Do some maths
   a) multiply **a** and **b**
   b) row 1 of **a** minus **c**
3) Change **a** row 3 to all 2s
4) Add a 4 to the end of **c**
5) find the standard deviation of each column of **a**
6) generate a random matrix **r** with the same size as **a**
7) convert **r** to a matrix with mean 5 and range 4 to 6
8) save your results in surname_ex1.mat

```
a =   0     2     3
      5    11    10
      9     7     6
      4    14    15

b = 5

c = 3  7  1
```

9) Create a new matrix **d** with **a** and then below that r
10) Put rows 3 and 5 of **d** into a new matrix **e**
11) Find the max and min of **d**
12) Find the range of values in **d**

Exercise Two – real data

Susie ran a stroop task on her labmate. She has reaction times for 15 incongruent trials, 15 congruent trials and 15 neutral trials saved in a data file called ex2.mat

1) Load ex2
2) There are some errors in the data – replace line 12 col 1 with 312 and line 4 col 3 with 234
3) Find the mean of each column and put it in a new variable called mdata
4) Plot the raw data as coloured stars
5) In a new figure, plot the mean reaction time with the standard deviation as error bars. Chose nice colours and symbols. Give a title and axis labels.

BLUE GREEN YELLOW
PINK RED ORANGE
GREY BLACK PURPLE
TAN WHITE BROWN

Exercise Three (Bonus)

- generate a variable **a** containing 100 values from 1 to 5
- generate a variable **b** containing 100 random values between 0 and 1 (>> *help rand*)
- **c = sin(a) + b,** plot c
- re-arrange c into 25 rows and 4 columns
- pretend each column of c is some data
- plot the median of each column of data
- add some nice labels