Functional Data Analysis (Lecture 1) – Matlab: Quick Summary

Zoltán Szabó

October 4, 2016

Demos, help, documentation, version, open, demos

```
>>help sin  %help on 'sin' with examples
>>doc sin   %more detailed help
>>ver       %version, installed packages
>>open conv %look at the source of function 'conv'
>>demo
>>demos
```

Matlab is case-sensitive.

Toolboxes

- Many Matlab packages. Examples:
  1. Optimization, Statistics, Curve Fitting, Image Processing, Signal Processing, ODE, Symbolic Maths, Control, Financial, ...
  2. Parallel, Distributed.
- Central repo for user codes:
  - File Exchange
  - https://fr.mathworks.com/matlabcentral/fileexchange/

'Everything is a matrix'

```
>>A = [1,2,3; 4,5,6]  %result is printed
>>A = [1,2,3; 4,5,6]; %result is NOT printed
>>b = [7;8;9]; A * b  %column vector, matrix x vector
```

Some operations:

```
>>A', A.^2                  %transpose, square elementwise
>>A(5,5) = 0.4              %enlarge A(!), fill with 0
>>B = rand(3); A * B,       %product of A and B
>>A(:,2), A(:,end-1:end)    %second column, last 2 columns
>>A.*A                      %Hadarmard product(similarly: ./)
>>kron(A,B),                %Kronecker product
>>A = rand(10); A^3         %3rd power of A
>>inv(A); pinv(A)           %(pseudo)inverse of A
>>A(1:2:10,:)               %extract the odd rows of A
>>diag(A),tril(A),triu(A)%diagonal, lower/upper triangular
```

See also: `repmat`, `norm`, `trace`, `expm`, `logm`, `sqrtm`, `eig`.

Creating matrices/vectors

```
>>A = rand(2,3)                        %U[0,1] coordinate-wise
>>size(A)                              %size of A
>>A = randn(2,3)                       %N(0,1) coordinate-wise
>>A = zeros(2,3)                       %zero matrix
>>B = eye(5), B = eye(2,3)             %identity (on the diagonal)
>>C = ones(2,3)                        %matrix of ones
>>D = hankel([1:5])                    %Hankel matrix
>>E = toeplitz([1:5])                  %Toeplitz matrix
>>v = [6:-2:-4]                        %[6,4,2,0,-2,-4]
>>w = [0:pi/2:2*pi], w = []            %2nd: empty matrix
>>w = randperm(5)                      %random permutation {1,...,5}
```

Reshaping matrices

```
>>A = rand(3,4)
>>A(:), length(A(:)) %vectorise A, length of A(:)
>>reshape(A,[2,6])   %3x4 -> 2x6
>>fliplr(A)          %left-right flip
>>flipud(A)          %up-down flip
```

Special names

```
>>ans  %result of the last computation
>>Inf  %infinity
>>NaN  %not a number
>>i,j  %complex i
>>pi   %3.1415...
```

Elementary functions

- `sin`, `cos`, `tan`, `sqrt`, `nthroot`, `log`, `exp`, `log2`, `log10` ...
- `abs`, `max`, `min`, `prod`, `sort`, `cumsum`, `cumprod`, ...
- `floor`, `ceil`, `round`: rounding
- They are acting coordinate/column-wise on matrices.

Saving, loading of variables (.mat)

Save:

```
>>A=2, B=pi,
>>who                 %list of variables
>>whos                %more detailed list of variables
>>FN = 'results.mat'; %filename
>>save(FN,'A','B')    %save 'A' and 'B' to FN
```

Load (after clearing):

```
>>clear A            %clear variable A
>>clear              %clear all variables from memory
>>FN = 'results.mat';
>>load(FN,'A','B')   %load 'A' and 'B' from FN
```

Plotting

```
>>t = linspace(0,2*pi,100);
>>y1 = sin(t); y2 = cos(t);
>>plot(t,y1,'r',t,y2,'g--','LineWidth',2);
>>grid off                                        %turn off grid
>>legend({'sin function','cos function'}) %put legend
>>plot(t,y1.^4);
>>hold on %hold current graph, similarly 'hold off'
>>plot(t,y1.^2,'g');
>>xlabel('variable t');       %similarly: 'ylabel'
>>figure; plot(rand(1,100)); %plot in a new figure
```

- Figures can be saved to .fig
- See also: `plot3`, `stem`, `mesh`, `surf`, `contour`, `scatter`, `pie`, `bar`, . . .

Random numbers

We have already used: `rand`, `randn`.

```
>>r = randi([-10 10],5,1) %integers
>>rand(2),rand(2),
>>rng(1),rand(2), rng(1),rand(2) %reproducible research!
```

Non-matrix types: string

```
>>s1 = 'Ecole'
>>s2 = 'Polytechnique'
>>s1(1:3)
>>length(s2)
>>strcat(s1,s2), [s1,' ',s2]
>>[s1,' ',s2]
>>strfind(s1,'ol')
>>disp(s1);
```

See also: `findstr`, `strcmp`.

Non-matrix types: cell array

Its elements can be anything.

```
>>c = {'apple',rand(5),pi}
>>c{1:2}
>>cell2 = {} %empty cell
```

Relations, logical operators

Relations:

```
>>a=1, b=3, c=1
>>a<b, a<=b, b==c, a~=c
>>d=rand(1,10), d>0.5 %acts entry-wise
>>any(d)
>>idx = find(d>0.5)
```

See also: `all`.

Logical operations:

```
>>(a>b) || (a==c) %or (short-circuit)
>>a>b && a==c     %and (short-circuit)
>>~(a>b)          %not
```

Scripts and functions

- extension: *.m*
- script: set of commands.
- function:
  - input → output, with a set of commands
  - definition:

```
function [o1,o2,o3] = f(i1,i2)
%This part appears
%in the help

  command1
  ...
```

  - calling: `[o1,o2,o3] = f(i1,i2)`
  - # of arguments: can also vary, see `plot`.
- PATH: `addpath(pwd)`

Debugging code

Matlab has a pretty *good* debugger:

- breakpoints,
- run and time (see also: `tic`, `toc`),
- run section.

Control structures

- Branching
  1. `if`: 'if-elseif-else-end',
  2. `switch`: 'switch-case-otherwise-end'.
- loop: `for`, `while`.

Note:

1. loop: *slow* in Matlab $\Rightarrow$ matricization!
2. Object-oriented programming: so-so.

Summary

- Many packages, latest release: R2016b.
- Linear algebra: *fast*!
- Matlab nicely supports scientific experimentation.