Practice 1: Introduction to Matlab

Siria Angino*

September 25, 2014

1 Introduction

A Matlab script file (also known as an ".m file") is a method for executing a series of Matlab instructions without typing all of the commands from the Command Window. Usually, we use a script to declare variables, to make algebra operations, to call functions or to plot figures. The basic objects in Matlab are scalars, vectors and matrixes.

Code:

```
% Declare a scalar, alpha:
alpha=2
% Make simple operation:
myresult=alpha+2
%Declare a vector:
beta=[1 2 3 4]
% Declare a 4x4 matrix:
gamma=[1 2 3 4 ; 5 6 7 8 ; 9 10 11 12 ; 13 14 15 16]
% How to know the size of a matrix?
mysize=size(gamma)
% Special matrices:
myNaN = NaN(4,6)
myzero = zeros(2,4)
myones = ones(3,3)
myidentity = eye(3,3)
% Let's extract the diagonal of a matrix:
gamma
mydiag = diag(gamma)
% rank of a matrix
a=[2 4 6 9; 3 2 5 4; 2 1 7 8]
rank(a)
% how to stack matrices
x=[1 2;3 4]
```

*Thanks to Federica Romei for the original version of these notes.

```
y=[5 6; 7 8]
Z=[x,y,(15:16)] % join matrices side by side
% how to select part of a matrix
% Make a new matrix gamma2, with columns 1 to 2 and rows 1 to 3 of matrix gamma
gamma2=gamma(1:3,1:2)
% make a new matrix gamma3, with all the rows and the first column of gamma
gamma3=gamma(:,1)
% Simple logical operators:
% if TRUE, MATLAB returns value=1, if FALSE, MATLAB returns value=0
% ask MATLAB which values in gamma are either <=4 , ==9 or >=14
gamma<=4 | gamma==9 | gamma>=14
% Use "&" for "and"
```

Remarks:

- Text after the symbol "%" is a comment!
- To compile the code, select it and type F9 (or right-click and select "evaluate selection")

We can also generate matrices of random variables. Code:

```
%Simulate one column of 1000 normally (0,1) distributed observations
X=randn(1000,1);
%Simulate on row of 1000 normally (0,1) distributed observations
Y=randn(1,1000);
%Simulate a matrix 10x10 of uniformally distributed observations
Z=rand(10,10);
%Simulate 9 matrices of 10x10 of uniformally distributed observations
ZZ=rand(10,10,9);
%Simulate another column of 1000 normally (2,16) distributed observations
XX=2+4*randn(1000,1);
```

Remark: A semicolon at the end of the statement means that matlab wont display the result in the command window.
You can sort the data, find the mean and the variance respectively with:

```
sort(X)
mean(X)
var(X)
```

See our Cheat Sheet at the end of this document!

Simple matrix operations: Matrix multiplication, inversion, transpose etc.:
Code:

```
%Multiply X by Y
X*Y
%Put each element in X to the squared
X.*X
```

```
%or
X.^2
%But if you try to do
X*X
%you get an error!
%transpose X
X
%invert gamma
inv(gamma)
```

We can make some plots of our vectors:

```
%plot simulated data X
figure(1)
plot(X)
%plot matrix Y (plotting a row or a column vector is the same)
figure(2)
plot(Y)
%plot vector X and the other vectors on the same graph:
figure(3)
plot(X)
hold on
plot(Y,red)
%%% have a look at the help for plot, you have many options!!
hold off
figure(5)
plot(X)
hold on
plot(XX,r)
% or
figure(5)
plot(X,.)
hold on
plot(XX,.r)
```

To plot vector x1 versus vector y1 (as a line) and vector x2 versus vector y2 (plus marker) on the same graph, use the command plot(x1,y1,-,x2,y2,+)

Remarks:

- To scale the axis, you can use the command axis: axis([xmin xmax ymin ymax])
- Matlab allows you to make many useful distribution plots (cdfplot, normplot, ...). See Matlab Help.

With all this basic knowledge, I think we are ready for a first exercise!

Exercise 1

- Generate a column of 100 normally distributed (0,1) observations X and a column of 100 normally distributed (0,1) observations epsilon.
- Define a column Y, as the sum of 2*X and epsilon. This means Y = 2X + ϵ.
- Using this data, run a simple OLS regression with Y as dependent variable, X as explanatory variable, and epsilon unknown.
- Make a matrix with as first column the actual value of Y, and as second column the OLS estimated value. Costruct the vector of residuals u.
- Plot the actual versus the estimated value.

2 Working with data: Loops

2.1 Importing data

Open a simple Excel file. Write the following numbers:
0 1
0 2
1 3
1 4
0 5
1 6
1 7
0 8

Save the file as ex2.xls.

Important: to avoid many problems, save the file as a Excel 97-2003 workbook.

Now, go back to MATLAB, and select as "current directory" the one in which you saved the xls file. To import the data in Matlab, just type the following command

```
%import the data as a matrix A
A=xlsread(ex2)
```

2.2 A basic loop

You might be interested in using loops, logical operators and IF/THEN.

Example 1

Consider an identity matrix (7x7) A, we want to subsistute the diagonal elements with an arbitrary vector b:

```
A=eye(7)
b=[1.5 2 3 4 5 6 7]
for i=1:7
A(i,i)=b(i)
end
```

Now lets substitute the small and large elements with 0, and the others with 5.

```
for i=1:7
if A(i,i)>=6 | A(i,i) <=2
A(i,i) = 0
else
A(i,i) = 5
end
end
```

Exercise 2

- Generate 10000 normally distributed (0,1) observations X
- Compute the fraction of observed values above 1,95 or below -1.95 (should be around 0.05)

Exercise 3

- Build three vectors of arbitrary numbers, considering that: the first must contain a binary indicator (0-1) for gender, the second contains age, the third contains salary. Write these vectors in one Excel sheet
- Import the Excel sheet in MATLAB as a matrix.
- Using the functions for and if separate the matrix gender gap in two sub-matrixes: one with the earnings of men and the other with the earnings of women.
- Plot the wage versus the age, using the plus sign (+) as marker style for the women, and a circle (0) as marker style for the men

3 Functions

You might find useful to write functions instead of doing everything in a single m-file. Moreover, you will use a lot of them as toolboxes such as the "statistical toolbox" will simplify your life a lot.

Example 3

Write a function "plusone". You give this function an input argument "value", and it returns you an output argument "added" defined as
added = value + 1
Save this m-file as "plusone" in your current directory. Code:

```
function[added]=plusone(value);
added=value+1;
```

To use this function, just declare the variable "value" in the command window and call the function Code:

```
value=2
added=plusone(value)
```

Remark: Define more input and output arguments as follows:

```
function [out-arg_1, ..., outarg_m] = myfunction(inarg_1, ..., inarg_n);
```

Exercise 4

Write a function "myownols" with as input arguments a column X and a column Y as defined in Exercise 1, and as output argument the estimated beta and the R^2 of your regression.

4 Cheat Sheet

- size(X,dim) returns the size of the dimension of X specified by scalar dim.
- ones(m,n) returns an m-by-n matrix of ones.
- zeros(m,n) returns an m-by-n matrix of zeros.
- diff(X) calculates dierences between adjacent elements of X and returns a matrix one element shorter than X.
- find(X) locates all nonzero elements of array X. You can use a logical expression to define X. For example, find(X ¿ 2)returns linear indices corresponding to the entries of X that are greater than 2.
- round(X) rounds the elements of X to the nearest integers.
- sqrt(X) returns the square root of each element of the array X.
- quantile(X,p) returns quantiles of the values in X. p is a scalar or a vector of cumulative probability values.
- corr(X) returns a p-by-p matrix containing the pairwise linear correlation co- ecient between each pair of columns in the n-by-p matrix X.

- var(X) returns the variance of X for vectors. For matrices, var(X)is a row vector containing the variance of each column of X.
- mean(A) returns the mean values of the elements along dierent dimensions of an array. If A is a matrix, mean(A) treats the columns of A as vectors, returning a row vector of mean values.
- cov(x), if X is a vector, returns the variance. For matrices, where each row is an observation, and each column is a variable, cov(X) is the covariance matrix.
- diag(cov(X)) is a vector of variances for each column, and sqrt(diag(cov(X))) is a vector of standard deviations. cov(X,Y), where X and Y are matrices with the same number of elements, is equivalent to cov([X(:) Y(:)]).
- xlsread(filename) returns numeric data from the first sheet in the Microsoft Excel spreadsheet file named filename. The filename argument is a string en- closed in single quotation marks. xlsread(filename, sheet) reads the speci- fied worksheet, where sheet is either a positive, double scalar value or a quoted string containing the sheet name.
- xlswrite(filename, M) writes matrix M to the Excel file filename.
- S = sym(A) constructs an object S, of class sym, from A.
- subs(S) replaces all occurrences of variables in the symbolic expression S with values obtained from the the MATLAB workspace.

5 Law of Large Numbers and Central Limit Theorem

State the Law of Large Number and the Central Limit Theorem. Draw N observations from a Normal Distribution with parameters μ and σ, from a χ^2 with parameter q, from a Cauchy distribution, from a Pareto and a Binomial. Compute the sample average and store it. Do this M times and then plot the histogram for every mean. Do the Law of Large Number apply to all of them? Do the Central Limit Theorem as well? (In case of negative answer, explain.)

Law of Large Number:
If y_i are a sequence of random variable i.i.d. and if exists $E[y_i] = \mu$, then the sample average will converge in probability to the expected value.

Central Limit Theorem:
If y_i are a sequence of random variable i.i.d. and if exists $E[y_i] = \mu$, and the $Var[y_i] = \sigma^"$ and we define y the sample average then: $sqrt(y - \mu)/n \rightarrow N(0, 1)asn \rightarrow \infty$

We are interested in testing when the Law of Large Number and the Central Limit Theorem are valid. Then we open our .m file and we write clear; clc. This two commands are really important. The first clear all the variables you

create before, therefore you are creating a new matlab space. The second, instead, clear the command window. It is not necessary. Then we set N= s of random draw and M=s of MonteCarlo Simulations. Start the for loop for the Montecarlo simulations. You can use the .m file randraw to draw from the dierent distributions (you can check the syntax in the .mfile). Then you can store the sample average in a matrix. At this point you can use the command histfit to plot the sample average. You can find all the program below with the comments:

```
clear;
clc;
%% First Practice
N=100; % Number of sample draw
M=30; % Number of time we do the loop
mu=0; % Mean of the Normal
sigma=1; % Variance of the Normal

Matrix1=NaN(M,1);
Matrix2=NaN(M,1);
Matrix3=NaN(M,1);
Matrix4=NaN(M,1);
Matrix5=NaN(M,1);
%% We write the random draws
%s= randraw(cauchy, [], [1 1e5]);
for m=1:M
y1=normrnd(mu,sigma,N,1); % Draw from the Normal
y2 = randraw(chisq, [2], [1 N]);% Draw from a chi-square parameter 2
y3=randraw(cauchy, [], [1 N]); % Draw from a Cauchy
y4 = randraw(pareto, [1, 1.5], [1 N]); % Draw from a Pareto parameters 1 and 1.5
y5 = randraw(binom, [10 0.5], [1 N]); % Draw from a Binomial parameters 10 and 0,5

% We store the sample average
Matrix1(m,1)=mean(y1);
Matrix2(m,1)=mean(y2);
Matrix3(m,1)=mean(y3);
Matrix4(m,1)=mean(y4);
Matrix5(m,1)=mean(y5);
end
%% Plot
% we plot the histogram of the sample average;
figure(1)
subplot(3,2,1)
histfit(Matrix1);
xlabel( x );
ylabel( frequency );
title(Normal)
```

```
subplot(3,2,2)
histfit(Matrix2);
xlabel( x );
ylabel( frequency );
title(chi)
subplot(3,2,3)
histfit(Matrix3);
xlabel( x );
ylabel( frequency );
title(cauchy)
subplot(3,2,4)
histfit(Matrix4);
xlabel( x );
ylabel( frequency );
title(pareto)
subplot(3,2,5)
histfit(Matrix5);
xlabel( x );
ylabel( frequency );
title(binomial)
```

Exercise 5

Do the same Montecarlo simulation (for the same distributions with the same parameters) for the variance. Are the Law of Large Numbers and the Central Limit Theorem valid?