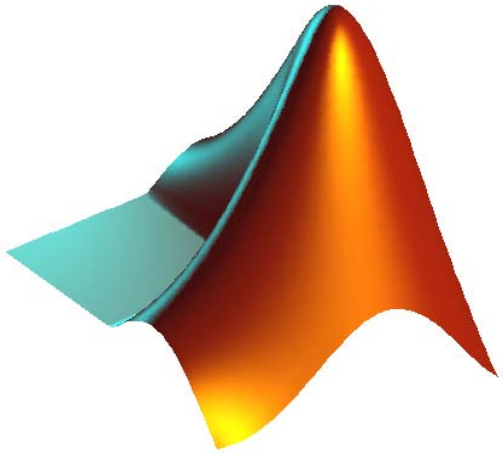
Lecture III

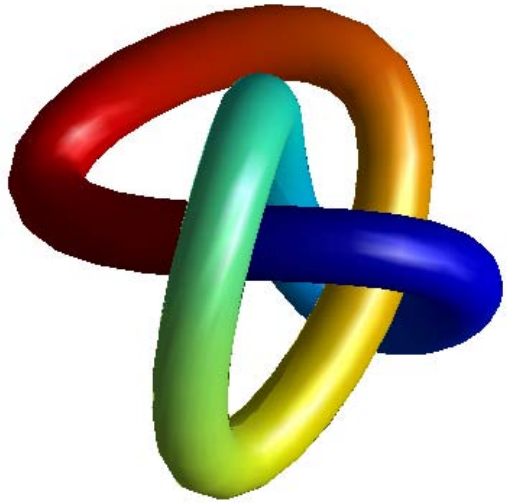


Matlab Crash Course

[Empirical Finance and Financial Econometrics]

Antonio Gargano
antonio.gargano@unibocconi.it
Department of Finance
Via Rontgen 1, room: 2.C3.05

Roberto Steri
roberto.steri@unibocconi.it
Department of Finance
Via Rontgen 1, room: 2.C3.04



Course and Class Structure

Basics

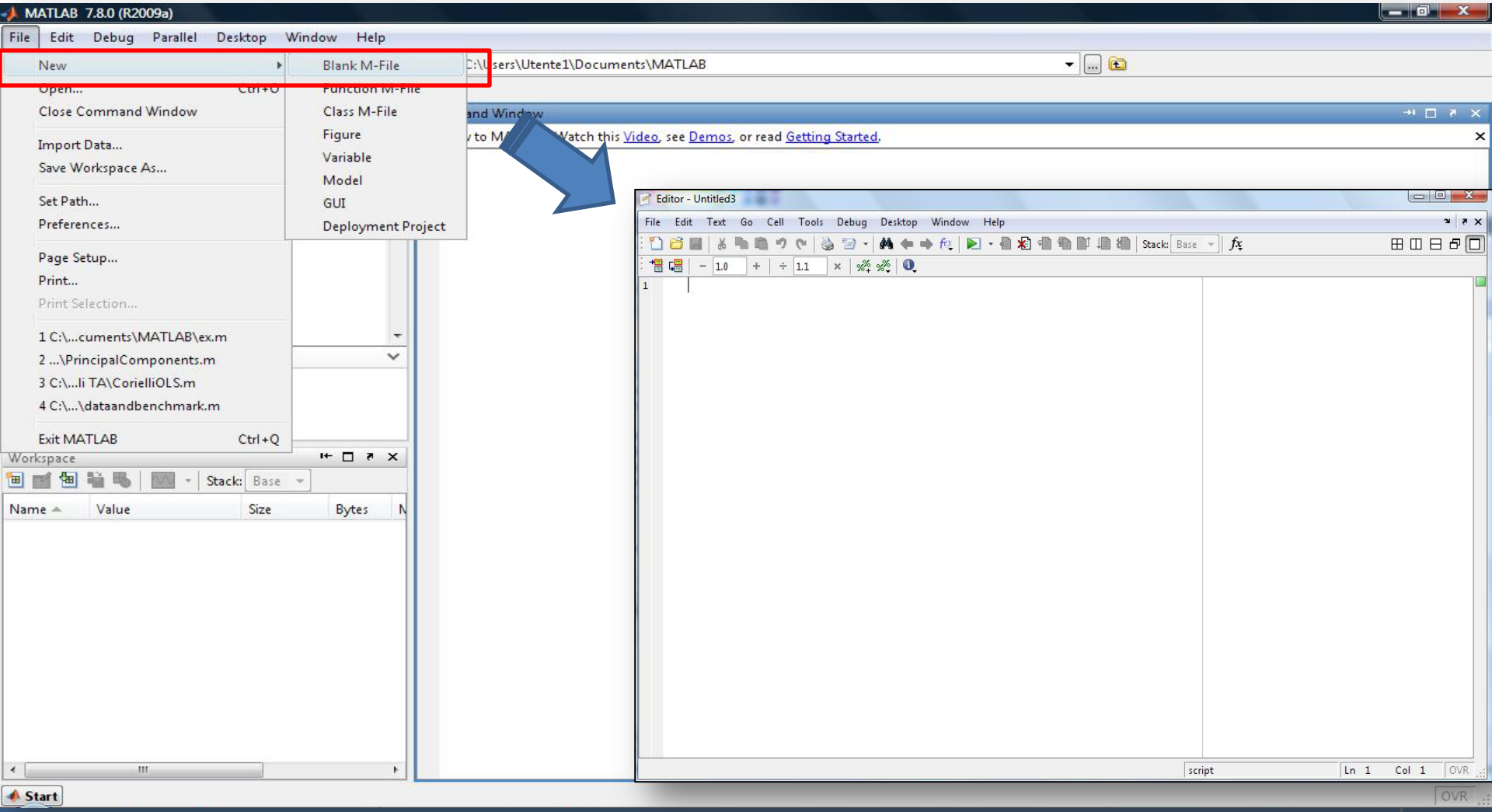
- **1st Lecture** — Intro to Matlab and its Environvment
  1. Variables Definition
  2. Matrix Access
- **2nd Lecture** — Intro to Matlab and its Environvment
  1. Matrix Access (exerc.)
  2. Single Matrix Manipulation (exerc.)
  3. Multiple Matrix Manipulation (exerc.)
- **3rd Lecture** — Intro to Programming
  1. Functions & Scripts(exerc.)
  2. For cycle(exerc.)
  3. Conditional Statetment: if (exerc.)

Applications

- **4th Lecture** — Applications

M-Files: Function and Scripts

Up to now we have just used the command window: it's not very efficient if you need many lines of code (as usually happens) to perform your tasks.

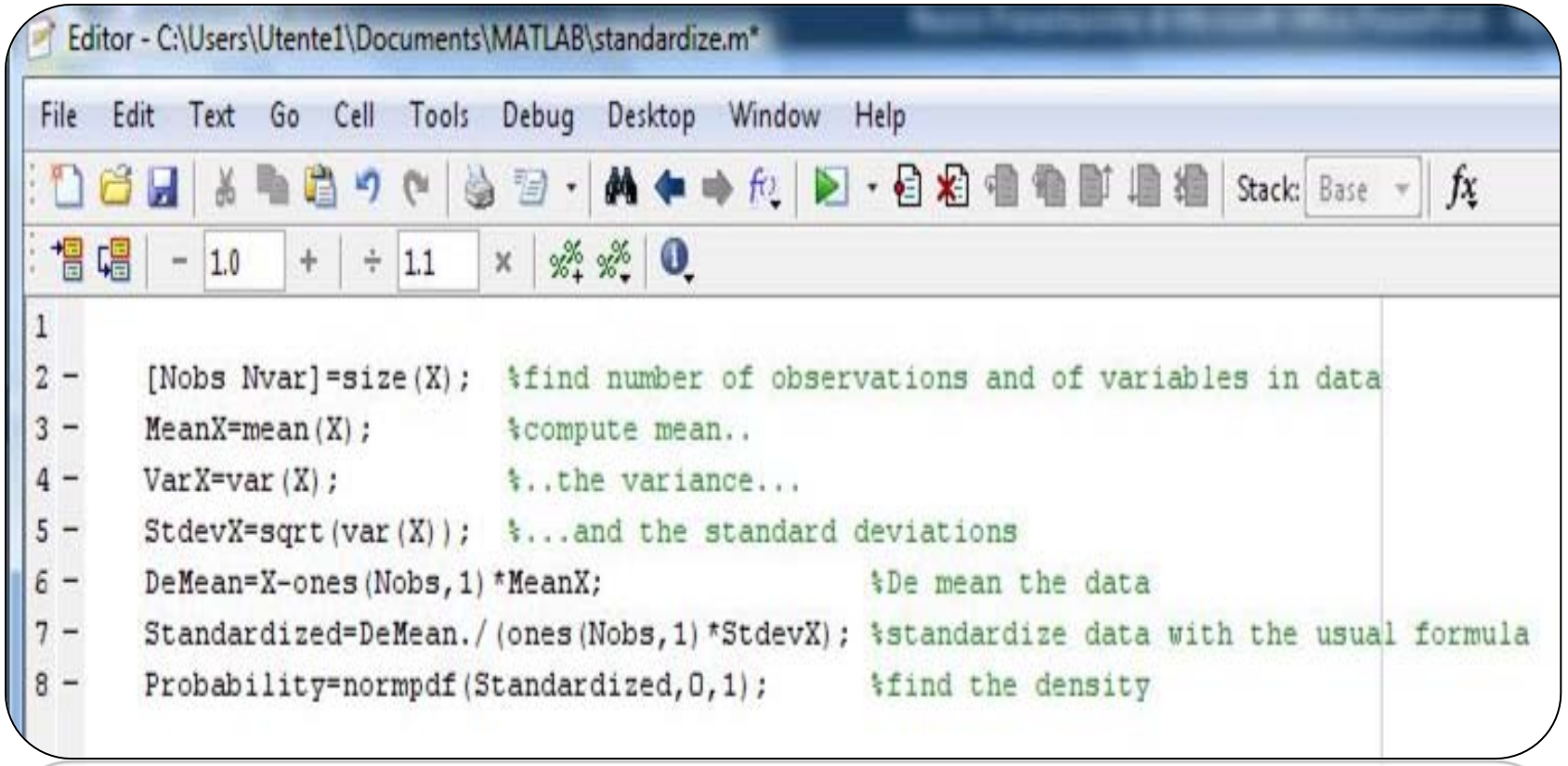


Example of script

What? A set of instructions executed consecutively.

Why? More Efficient (and easy to store and re-use) than write the code line by line.

How? Write "run standardize.m" in command window or in other scripts

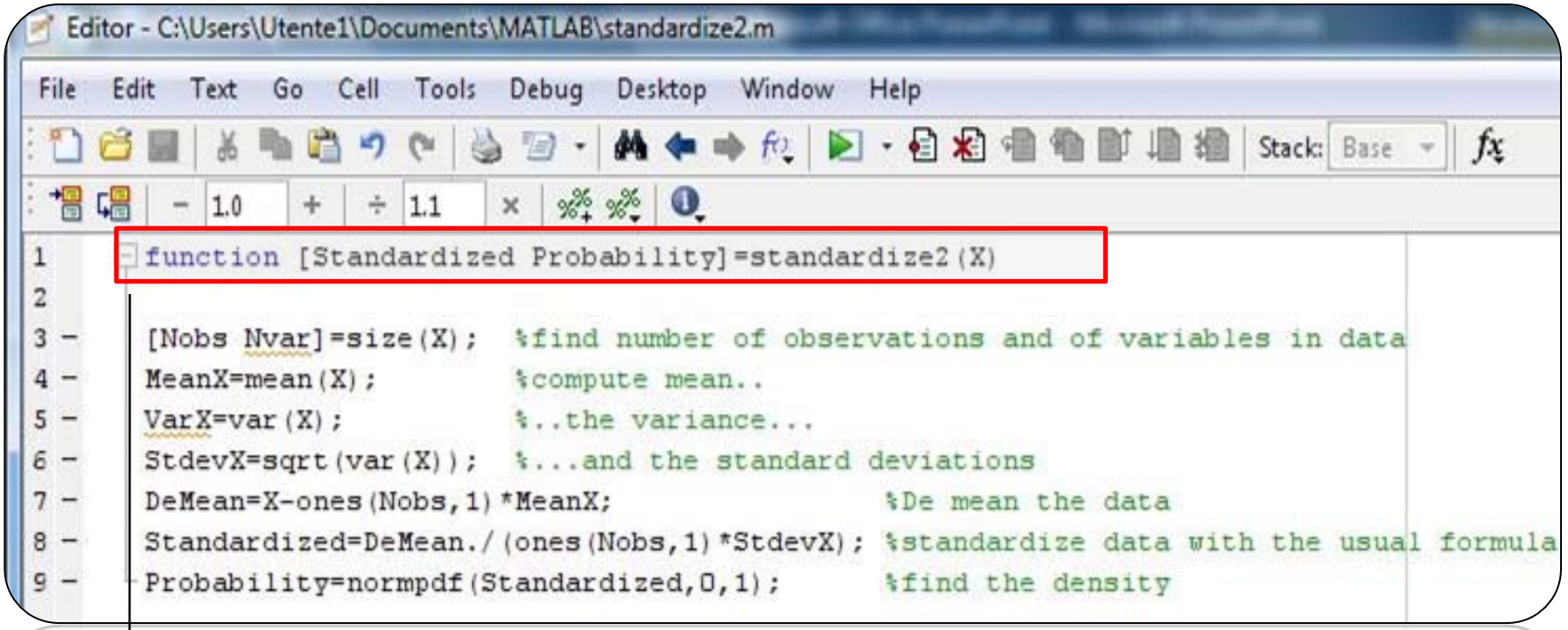


Example of function

What? Instructions executed consecutively.

Why? More efficient than writing the code line by line.

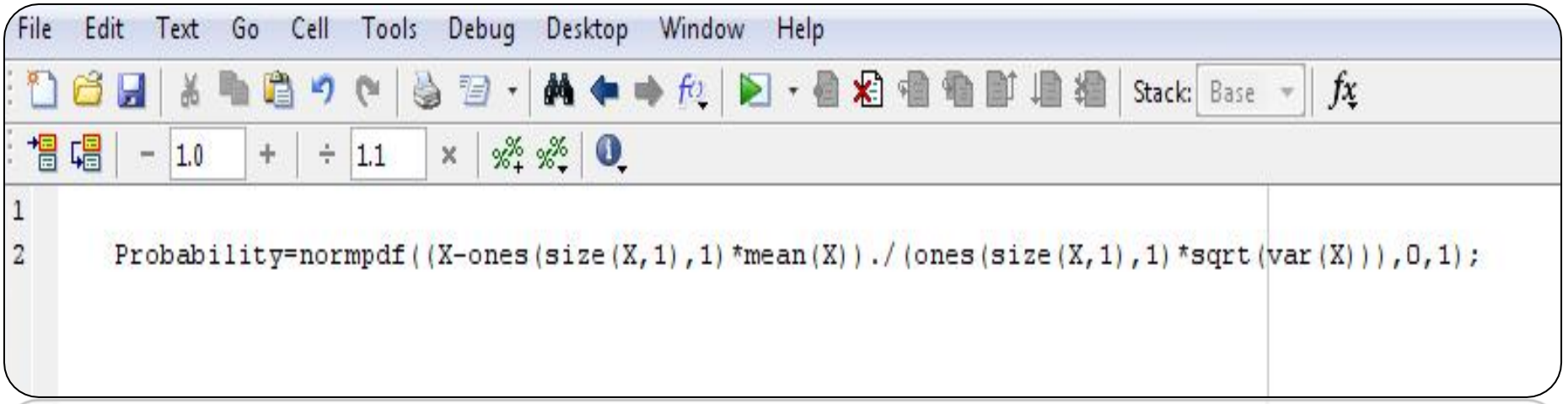
How? Output=functionname(input) in comm.win. or in other scripts



a) Write "function" and give to the file.m the same name!
b) Specify the outputs
c) Specify the Input

Parenthesis about programming…

I might have written the previous code in just one (faster) line…



…at the cost of being less clear. With experience you will find your optimal point between clarity and speed.

Example of a “well written” function

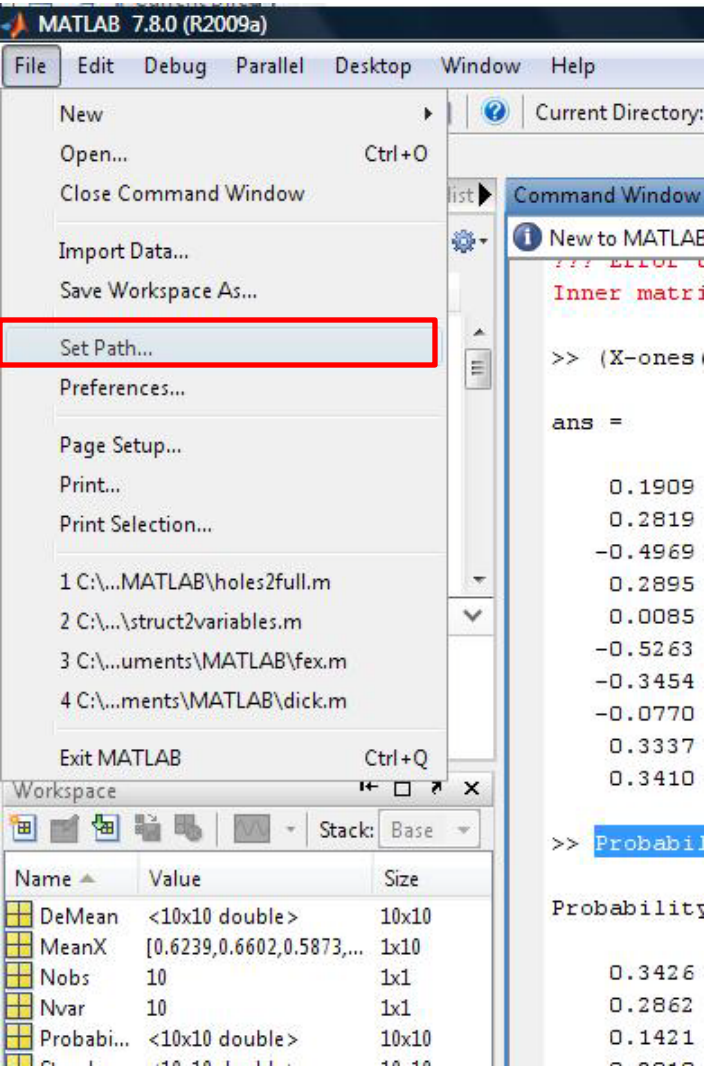
```
function [Standardized Probability]=standardize2(X)

% PURPOSE: Given a matrix of random variables (normally distributed) find
%          the corrispective standardize values and their probability
%
% USAGE:   [Standardized Probability]=standardize2(X)
% ------------------------------------------------
% INPUT:
% Required: X: NobsxNvar matrix of random variable normally disributed
%
% ------------------------------------------------
% OUTPUT:   Standardized: Original data standardized
%           Probability:  Probability of standardized data
%
% ************************************************************
% REFERENCE: "Introductory Statistics", (S. Ross), McGrawHill, 2001
```

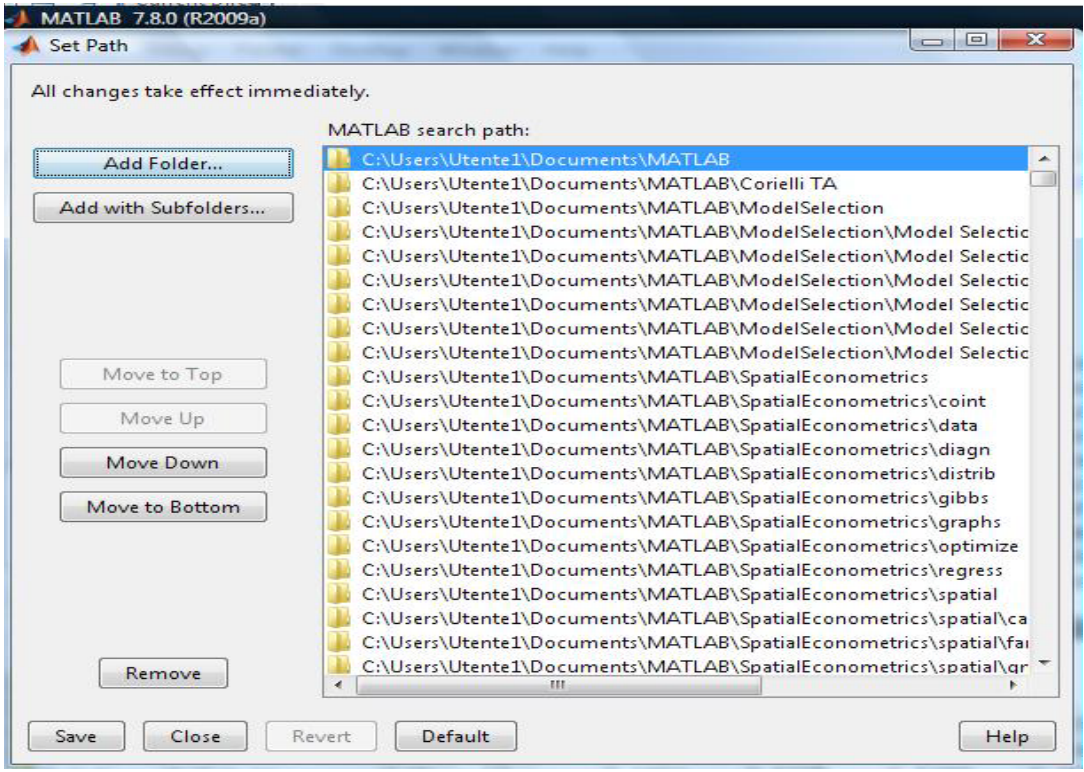
a) It helps others to understand what you have done (very often you are expected to co-work on complex projects)

b) It helps yourself when you re-use the function after long time

```
[Nobs Nvar]=size(X);                          %find number of observations and of variables in data
MeanX=mean(X);                                %compute mean..
VarX=var(X);                                  %..the variance...
StdevX=sqrt(var(X));                          %...and the standard deviations
DeMean=X-ones(Nobs,1)*MeanX;                  %De mean the data
Standardized=DeMean./(ones(Nobs,1)*StdevX); %standardize data with the usual formula
Probability=normpdf(Standardized,0,1);        %find the density
```

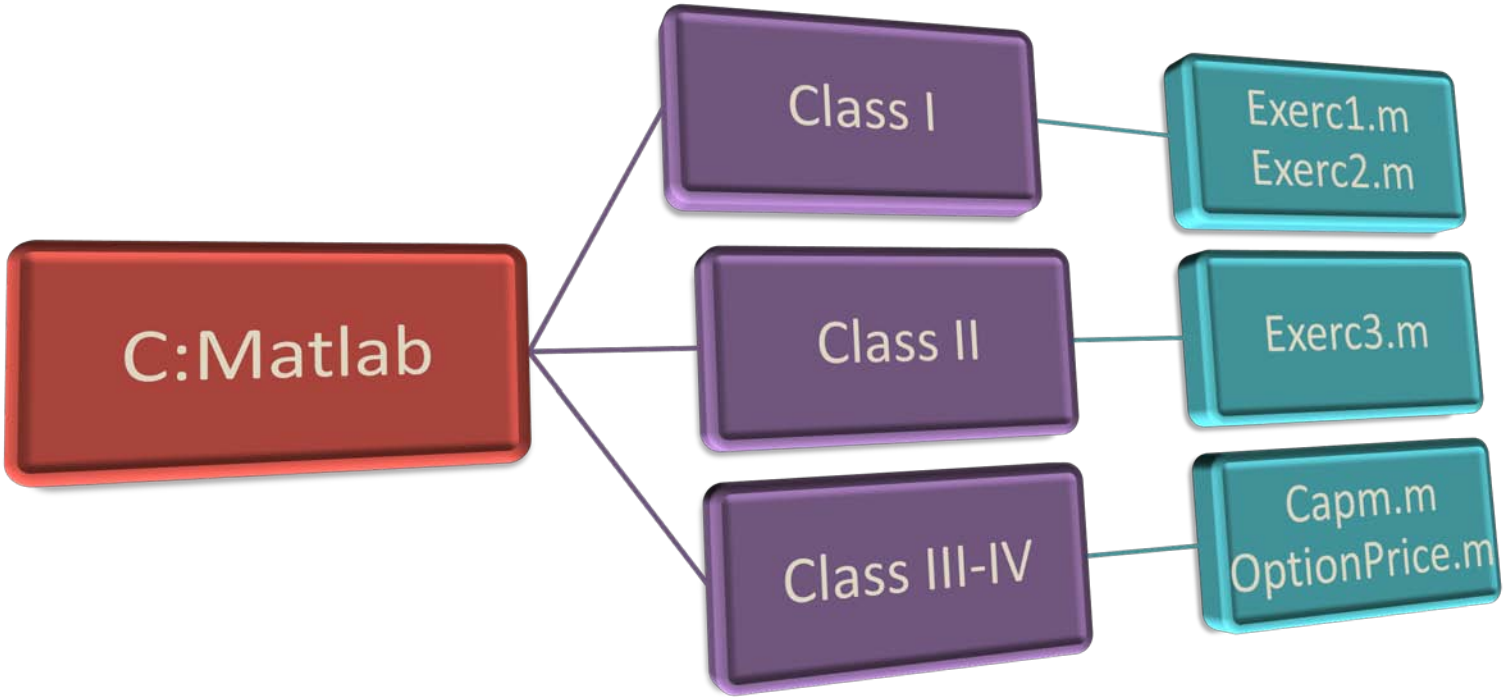
Few words about Matlab paths…





Group the files in folders and add them to the path structure, otherwise Matlab won't be able to find (and to execute) them!

Few words about Matlab paths…



If I add to my path only ClassI and Class II and i run the function "Capm.m" or "OptionPrice.m" it will display the following error:

??? Error using ==> run at 76
Capm not found.

STOP

Matlab does not "update" automatically your path structure when you create/delete folders trought windows

Exercises

a) Create a folder “Exercises” in your current directory and add it to your path.
hint: follow the procedure in previous slides

b) Create a script and a function that both solve the problem in point c and call them “regressfunction” and “regressscript”
hint: look at the slide “example of a well written function”

c) Given X=[ones(30,1) 2*randn(30,1)] and y=randn(30,1) , specified outside the function, compute the Betas and the residuals of the linear regression of y on X
hint: your code it’s just 3 lines long

$$Beta = \left(X'*X\right)^{-1} * X'*y$$

$$yhat = X * Beta$$

$$\text{Re}siduals = y - yhat$$

d) Write [Beta Residuals]=regressfunction(X,y) and press enter
Write run regressscript and press enter
What’s the difference?

Solutions

```
function [Beta Residuals]=regressfunction(X,y)
%  PURPOSE: find Coeffs and Residuals of the linear regression of y on X
%  USAGE:   [Beta Residuals]=functionregress(X,y)
% ------------------------------------------------
%  INPUT:   X=(Nobservations*NVariables) Matrix of regressors
%           y=(Nobservations*1)          Vector of the dependent variable
%  ------------------------------------------------
%  OUTPUT:  Beta= (NVariable*1) vector of coefficients
%           Residuals= (Nobservations*1) vector of residuals

Beta=inv(X'*X)*X'*y;
yhat=X*Beta;
Residuals=y-yhat;
```

The main difference you can observe between the script and the function is that, when executed, the function stores in memory (in your "workspace") only the Output Variables (in this case Beta and Residuals) while the script stores everything (in this case Beta, Residuals and yhat)

Another big difference is that functions are more flexible and allow you to specify also optional parameters

Conditional statements I: for and while

```
For i=1:N
...
end
```

```
While condition
...
end
```

Some examples:

```
For i=1:N
rand
end
```

```
For i=1:N
i
end
```

```
For i=1:N
M(i,:)=5
end
```

```
For i=1:N
M(i,:)=5*i
end
```

```
For i=1:N
M(i+1,:)=M(i,:)
end
```

```
while i=<N
rand
i=i+1
end
```

```
while x<=100
x=i^2
i=i+1
end
```

If you design the while cycle correctly you might be able to avoid useless iterations.

Problems Set I

Conditional statements II: if and switch

```
if condition
...
elseif condition
...
else
...
end
```

```
switch condition
...
case condition
...
case condition
...
end
```

Some examples:

```
if  x>100
    x=100
end
```

```
x=input('Select  number regressors')
if x==1
Regr=Data(:,1)
elseif x==2
Regr=Data(:,1:2)
else
Error('you can not select more than 3 regressors')
end
```

```
switch  x
    case >=0
    'x is positive'
    case <0
    'x is negative'
end
```

Problems Set II

Conditional statements III

For and nested if:

```
Big=[]; Small=[];
For i=1:N
    x=rand()
    if x > 0.5
    Big=[Big  x]
    elseif x<=0.5
    Small=[Small x]
    end
end
```

If and nested for:

```
x=input('Select  # regressors')
if N>0
    For i=1:N
    i
    end
Else
    For i=1:abs(N)
    -i
    end
end
```

Problems Set III