academic year 2019

Introduction to Finance with MATLAB

University Paris-Dauphine
Magistère Banque, Finance & Assurance

Gauthier Vermandel[1] *gauthier.vermandel@dauphine.fr*

Lecture 4: Creating Functions

Content of the Lecture

Objectives of this lecture:

- Write a function with n-input arguments.
- Code a function n-output variables.

[1]Department of Economics, Paris-Dauphine University. Codes available on my website: : http://vermandel.fr/bfa1.

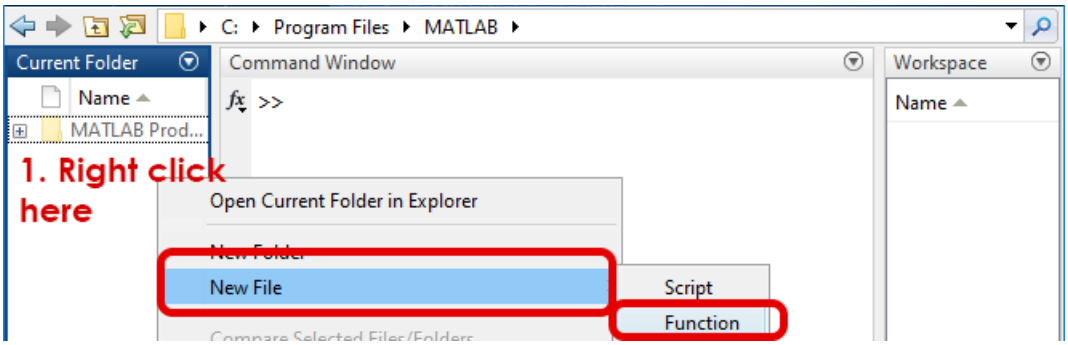
1. Introduction

Functions are self contained modules of code that accomplish a specific task. Functions usually take in arguments, process it, and return a result. Once a function is written, it can be used over and over (and over?) again. Functions can be called from the inside of other functions.

2. Declaring a simple function

Functions are m-files that can accept input arguments and return output arguments. The name of the m-file and of the funcction should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the Matlab command prompt. Functions are useful for extending the existing Matlab language for personal applications For example a practitioner might want to write various functions that return the theoretical price of a call option contract according to various options pricing models. Options pricing models like the Black and Scholes, the Merton's Jump's diffusion, the displaced diffusion model, Heston's stochastic volatility model, and other can easily be implemented via a different function.

First of all, let us create a function through MATLAB. The latter offers a simple way to create an empty function:



MATLAB automatically creates the following empty function (I called mine 'hello'):

```
function [ output_args ] = hello( input_args )
%HELLO Summary of this function goes here
%   Detailed explanation goes here


end
```

Here, input_args denotes the argument(s) that the function will process and transform them into the result()s stored in output_args.

As an example, we can code a function which says hello to someone:

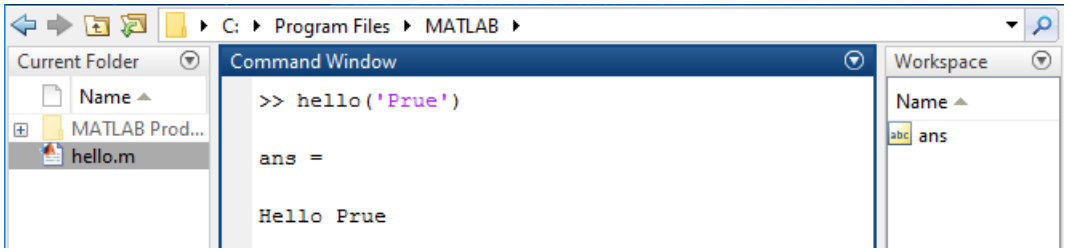
```
function [ to_be_shown ] = hello( name )

    % concatenate strings
```

```
    to_be_shown = ['Hello ' name];

end
```

Giving 'Prue' as a name, the function displays the following message right in the console window:



As an example, a function computing the asset return from the stock price is given by:

```
function dl_x = logdiff(x)
        dl_x =  diff(log(x));
end
```

Then we can call this function:

```
r =  logdiff(SP);
g =  logdiff(GNPPC);
```

This will provide the growth rates of each variable.

It is also possible to declare functions with multiple inputs/outputs. Suppose that we are willing to get both the log and the log difference as output, and we want to multiply by 100 or 1 the result:

```
function [dl_x,logx] = logdiff2(x,factor)
        dl_x =  factor*diff(log(x));
        logx = factor*log(x);
end
```

This function must be called this way:

```
% gross value
[r,logSP] =  logdiff2(SP,1);
% percentages
[r100,logSP100] =  logdiff2(SP,100);
```

Note that functions are always stored in external files (i.e. you are not allowed to store function directly in the script file), however you can store multiple function in a single matlab file.

Exercise 1

Take back the code of the first assignment:

1. Write a function to find cube of any number using function. Test your function with 2 random numbers.
2. Now this function should provide both the cube and the square of a number. Test your function with 2 random numbers.
3. Write a program to find maximum and minimum between two numbers using functions. Test your function with 2 random numbers.

Assignment 1

Copy this in a new MATLAB file. Check that the copy/paste has not broken the code, and download getMarketDataViaYahoo.m file. Make sure that the latter appears in your current folder in MATLAB, otherwise it won't work.

```
% get yahoo data
d = getMarketDataViaYahoo('IBM', '01-01-1990', 'now', '1mo');
plot(d.Date,d.AdjClose)
legend('IBM')
```

This code loads on Yahoo Finance the monthly data of the closing price of IBM and Microsoft shares on financial markets since 1991.

1. Create function "share_return" computing the rate of return from holding a share $(P_t/P_{t-1}-1)$. This function must include a loop. Use this function to compute the rate of return of IBM and Microsoft shares.
2. The function share_return must be userfriendly. To this purpose, if the argument provided is not a number, the function should stop and give an error message (use function error() with a message indicating why the function has suddenly stopped). Same question when the user has not provided any argument (check "nargin" condition)
3. Create another function, called moving_average, computing a moving average of the share return (the size of the window is 1 year, i.e. 12 months). Use it on IBM and Microsoft.
4. This function moving_average should include as a second argument the size of the window.
5. This function moving_correlation computes the contemporaneous correlation between two assets in the same fashion as the moving_average. The window size should be also an input of the function. Use function corr() to compute the correlation.
6. Report the moving averages of each asset on the same plot, and use subplot to draw the dynamic correlation right after the first plot.

7. Create a function called moving_fast, nesting moving_average, moving_correlation which takes as inputs the share prices of two assets and the size of the window (i.e. 3 arguments) and returns as output the share price of each asset (2 outputs) + the moving average of each asset (2 outputs) + the common rolling correlation (1 output). Call this function in a matlab file.