academic year 2019

Introduction to Finance with MATLAB

University Paris-Dauphine
Magistère Banque, Finance & Assurance

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

Lecture 3: Conditional Statements and Loops

Content of the Lecture

Objectives of this lecture:

- Mastering the use of conditional statements.
- Coding loops with both while and for.

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

1. Introduction

In computer science, conditional statements, conditional expressions and conditional constructs are features of a programming language, which perform different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false. Apart from the case of branch predication, this is always achieved by selectively altering the control flow based on some condition. Here, in MATLAB it is quit easy to implement them.

In computer science a loop is a control flow statement for specifying iteration, which allows code to be executed repeadly. Here, we will work on the for-loop and while-loop, which are the main way in computer science to performs iterations.

2. Conditional statements

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement.

For example, to report when a random number between [0,100] is above 50, we define the following conditional statement:

```
% Generate a random number
a = randi(100, 1);
% If above 50, send a console message
if a > 50
        disp('a is above 50')
end
```

note: num2str(.) converts a real number into a string, disp(.) displays the argument in the console and [string1 string2] concanates strings into a single string.

It is also possible to contrast the above/below situation using else statement:

```
% Generate a random number between 0 and 100
a = randi(100, 1);
% If above 50, send a console message
if a > 50
        disp('a is above 50')
else
        disp('a is equal or below 50')
end
```

Or even contrast three situations: above/below/equal using elseif statement:

```
% Generate a random number between 0 and 100
a = randi(100, 1);
% If above 50, send a console message
if a > 50
        disp('a is above 50')
elseif a == 50
        disp('a is equal to 50')
```

```
else
        disp('a is below 50')
end
```

It is also possible to make multiple conditional statements using "&&" meaning "and", and "||" meaning "or".

```
% Generate a random number between 1 and 4
a = randi(4, 1);
if a == 1 || a == 3
        disp('a is odd')
else
        disp('a is even')
end
```

Writing only one "&" or "|" will not work.

Exercise 1

1. Write a program to find maximum between two random numbers called x and y (use rand(·) function). You must employ conditional statements.
2. Same question, but with three numbers (the third one is called z).
3. We want to create a traffic light system.
   (a) Pick up a random integer in a uniform discrete distribution taking values {1,2,3} using function randi(·) and store in variable trafficlight.
   (b) Letting 1 denotes the red light, 2 the orange and 3 the green one, write a program that shows in the console the traffic light color, e.g. "The traffic light color is red" (use disp(·) function).
   (c) Pick up a random number in a uniform discrete distribution taking values {1,2} and store in variable decision, the latter determines the decision of the driver (2=pass the traffic light, 1=stop). Print the decision in the console.
   (d) Warn the driver (write "STOP" or "CONTINUE" in the console) mapping the decision of the driver and the color of the traffic light.

3. Loops

The for loop repeats a group of statements a fixed, predetermined number of times. A matching end delineates the statements. for statements loop a specific number of times, and keep track of each iteration with an incrementing index variable. For example, preallocate a 3-element vector, and show its content for each row:

```
% taking 3 random numbers in a vector
a = rand(3,1);
```

```
% open a loop to display the content of the vector
for i = 1:length(a)
        % show in console
        disp(['row ' num2str(i) ' has value ' num2str(a(i))]);
end
```

Where variable i is the incrementing index variable. Running this code in the command windows returns:

```
row 1 has value 0.49809
row 2 has value 0.90085
row 3 has value 0.57466
```

It is also possible to nest loops. As an example, if we want to show the content of a 3x3 matrix:

```
% taking 9 random numbers in matrix 3x3
a = rand(3,3);
% parsing rows
for i = 1:size(a,1)
        % parsing columns
        for j = 1:size(a,2)
                % show in console
                disp(['row ' num2str(i) ' and column ' num2str(j) ...
                                ' has value ' num2str(a(i,j))]);
        end
end
```

This code returns:

```
row 1 and column 1 has value 0.58145
row 1 and column 2 has value 0.016983
row 1 and column 3 has value 0.4843
row 2 and column 1 has value 0.92831
row 2 and column 2 has value 0.12086
row 2 and column 3 has value 0.84486
row 3 and column 1 has value 0.58009
row 3 and column 2 has value 0.86271
row 3 and column 3 has value 0.20941
```

The while loop repeats a group of statements an indefinite number of times under control of a logical condition. That is, as long as an expression is TRUE, then the segment of executable programming code that is included in the while statement is executed. A matching end delineates the statements. The syntax for the while loop is given by:

```
% taking 3 random numbers in a vector
a = rand(3,1);
% defining the starting value of the incremental variable
```

```
i = 0;
while i < length(a)
        % incrementing
        i = i + 1;
        % show in console
        disp(['row ' num2str(i) ' has value ' num2str(a(i))]);
end
```

This code returns exactly the same output as in the first loop example with for statement. The while statement is requires more code than the for statement, however the while loop offers more liberty for the stop conditions of loops.

We can also nests loops and provide exactly the same output as in the second example for for statement:

```
% taking 9 random numbers in matrix 3x3
a = rand(3,3);
% defining the starting value of incremental variables
i = 0;
j = 0;
% parsing rows
while i < size(a,1)
        % incrementing
        i = i + 1;
        % parsing columns
        while j < size(a,2)
                % incrementing
                j = j + 1;
                % show in console
                disp(['row ' num2str(i) ' and column ' num2str(j) ...
                                ' has value ' num2str(a(i,j))]);
        end
        % restarting the increment for columns
        j = 0;
end
```

Exercise 2

1. Declare x=[3,7,9,10]; and y=[0,7,8,1];. Code a for-loop which performs a dot product between x and y (*i.e.* multiply each element of x with the corresponding elements of y). Store the result in a matrix z.
2. Same question with a while loop.
3. Print the elements of matrix x in reverse (from n to 1).
4. Create a loop which sums the elements of x.
5. Pick up 50 random integers in a uniform discrete distribution taking values {1,...10} using function randi(·). Print elements of the vector which are above 7 and below 2.

6. Pick up an interger between 1 and 100, and store it in variable called mysterious_nb. Create a loop which each iteration also picks up a random integer between 1 and 100. As long as the latter integer is different from the mystery number, the loops keeps on running. When the loop is over, a variable called nb_iteration should contain the number of iterations of the loop to get the mystery number.

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 shares on financial markets over the last 30 years.

1. Create a loop computing the return from holding IBM shares ($P_t/P_{t-1}-1$) called r. Plot it.
2. Create a loop computing the annual return ($P_t/P_{t-4}-1$) called ra.
3. Use a loop to compute the mean and the variance of the return.
4. Compute the average IBM share return r for the first year of the sample.
5. Create a loop which computes the moving average of r. The window size computing the average should be of one year and the moving average should be stored in variable called ma. Plot both ma and r in the same graph.
6. We seeks the 10% extreme values of the sample of r. Create a loop which stores these values in vector called extr_val.
7. From the extr_val vector, determine the id/position of each extreme value in r. Store the ids in ext_id. Your code must include two nested loops combined with conditional statements.