A Very Quick Guide to MATLAB

The versatile numerical program MATLAB is installed on the lab computers in Exactum C128. In the MATLAB command window, you can write expressions and commands that the program will calculate or evaluate when you press enter. You can get help and more information with the command **help**. For example, **help sqrt** will produce information about the square root function, sqrt.

In MATLAB, it is handy to create *vectors* (lists of numbers), around which you can define a wide variety of operations. For example, **sum(x)** calculates the sum of the elements in vector x.

Below are some examples. The user's command appears after the prompt >>. On the following line is the value of the expression evaluated by MATLAB. If you don't want the expression's value to appear (e.g. because it is a very long vector), add a semicolon (;) to the end of the expression.

```
>> 5 + 2*3          % Basic arithmetic
ans =
     11
>> x = 20           % Assigning a value to a variable with an equal sign
x =
     20
>> y = x*x          % Using a recent variable
y =
    400
>> z = [5  3  10  12  10]    % Defining a vector using square brackets
z =
      5      3     10     12     10
>> sum(z)           % Calculating the sum of the vector's elements
ans =
     40
>> z < 8            % Which elements are less than 8?
ans =
      1      1      0      0      0
>> z == 10          % Which elements are exactly equal to 10?
ans =
      0      0      1      0      1

>> sum(z==10)       % How many elements are equal to 10?
ans =
      2
```

Auxiliary functions

You can find some auxiliary functions for this course at `http://www.helsinki.fi/~kohonen/introprob/`. You can also get some guidance for their use with the help command (e.g. `help dice`). Hint for downloading: First start MATLAB so that it creates a subdirectory called "MATLAB"in your home directory. Then, save the files from the URL above one by one (depending on your web browser, you may for example need to click the right mouse button and select "Save Target As"or "Save Link As"). Make sure to save them in the MATLAB directory (Desktop > Libraries > Documents > MATLAB). After that you can use them in MATLAB. In MATLAB you can use the `dir` command to see which help functions you have saved.

- `dice` simulation of throwing a die one or many times
- `coin` simulation of throwing a coin one or many times

```
>> dice            % Simulates one throw of a die
ans =
     5
>> a=dice(10)      % Throws the die 10 times, generating results as a vector
a =
     5     3     3     6     4     6     2     4     4     6
>> sum(a==6)       % Counts the number of times a 6 appeared
ans =
     3
>> b = coin(10) % Flips a coin 10 times
b =
     1     1     1     1     0     0     1     0     1     1
>> c = coin(1000);       % The semicolon tells MATLAB not to display the result
>> sum(c)                % Counts the number of times the coin flip produced head
ans =
   483
```

Logic Expressions

In probability theory, logic expressions are essential for evaluating the truth values of individual claims as well as combinations of claims. For example the inequality `X < 3` is a logic expression, which is true (1) if the value of variable X is less than 3 and false (0) otherwise. Some important logic operations are:

`A & B`	conjunction, "A and B", i.e. whether both are true
`A | B`	disjunction, "A or B", i.e. whether at least one of them is true
`not(A)`	negation, "not A", i.e. whether A is false

You can evaluate logic operations for an entire vector. The operation you specify will be executed for each individual element of the vector, and the result will correspondingly be a vector.

Below we throw two dice (a red one, R, and a black one, B) 10 times each and investigate what happens each time.

```
>> R=dice(10)
R =
     1     4     4     1     6     6     3     5     1     4
>> B=dice(10)
B =
     2     3     6     2     2     3     3     5     6     1
>> R==3            % Which times did the red die land on a three?
ans =
     0     0     0     0     0     0     1     0     0     0
>> B==3            % Which times did the black die land on a three?
ans =
     0     1     0     0     0     1     1     0     0     0
>> R==3 & B==3  % Which times did both land on a three?
ans =
     0     0     0     0     0     0     1     0     0     0
```

```
>> R==3 | B==3  % Which times did at least one of the die land on a three?
ans =
     0     1     0     0     0     1     1     0     0     0
>> not(R==3)    % Which times did the red die NOT land on a three?
ans =
     1     1     1     1     1     1     0     1     1     1
```