Lab 1
CO902 – Probabilistic and statistical inference – 2012-13 Term 2
Lecturer: Tom Nichols

MATLAB Tips (applicable to any scripting language, really)

A. Create a directory for project/lab work. Change to that directory first thing, e.g.

```
cd /path/to/my/LabWorkDir
```

B. Avoid typing commands directly into the console window. Instead, always work inside an editor, copy-&-pasting commands as you go, creating a record (i.e. a script) of your work. (Hint, find and use the keyboard short-cut for "Evaluate Selection"; on my Mac it's Shift-F7).

C. Always document your script. The MATLAB comment character is %; first lines beginning with % are shown as help. Give one-line description, followed by more detail; for example,

```
% Scratch work for CO902 Lab 1
%
% Lab exercise 1 scratch work, and demonstration of code documentation
%_____________________________________________________________________
% Script:   Lab01.m
% Purpose:  Discrete r.v. demonstration & Monte Hall Monte Carlo
% Author:   Tom Nichols
% Date:     8 Jan 2012

% This comment doesn't appear in help
disp('hello world')
```

D. Whenever possible use mnemonic variable names, with possible exception of dummy variables. Here's some bad code

```
h=500;
f=3;
x=rand(h,f);
s=mean(x);
```

Here's some better code

```
nSim=500;
nSubj=3;
Trials=rand(nSim,nSubj);
TrialMean=mean(Trials);
```

Which do you understand?
However, don't go overboard… index variables are universally i, j, k; i.e. this is a bit silly

```
for subject_index = 1:nSubj
      TrialMean(subject_index) = mean(rand(nSim,1));
end
```

when this is perfectly clear

```
for i = 1:nSubj
      TrialMean(i) = mean(rand(nSim,1));
end
```

E. Avoid "Magic Numbers". If, in your script, you're going to creating a simulate 1000 random numbers, don't write

```
X = rand(1000,1);
```

Rather, *define* the Magic Number, 1000, and then use the variable.

```
nSim=1000;
X = rand(nSim,1);
```

This makes for code that is much easier to read and maintain.

Lab Exercise

1. The MATLAB function *rand* returns a pseudo-random number between 0 and 1, that is a draw from a uniform distribution on the interval [0,1]. Use *rand* to write a new function *rand_int(a,b,n)* which returns *n* integers randomly chosen from between *a* and *b* inclusive.

   Hint 1: A MATLAB function, say, MyNewFunc, is a plain text file called MyNewFunc.m where the first line begins

   ```
   function [outvar1,outvar2] = MyNewFunc(invar1,invar2,invar3)
   ```

   where invar1, invar2 & invar3 are the input variable names, and outvar1 & outvar2 are the names of the variables you get to set that will be given back.

   Hint 2: Matlab assumes you always want a matrix as a result, so if you omit the 2nd dimension of a matrix, it assumes you want a square matrix. Compare `rand(10)` to `rand(10,10)` to `rand(10,1)`.

2. The classic "Monty Hall" problem has a game show with three (closed) doors, behind only one of which lies a prize. The host invites the participant to choose a door. She then opens one of the other doors and reveals it to be empty and asks the participant whether he wants to switch to the remaining door. By computer simulation in MATLAB, can you decide on whether it is better to switch or not?

3. (If time). Consider throwing a fair 6-sided die, and getting a value X, and then squaring it to X^2. Using a loop (say, for 500 times), repeatedly use your function rand_int(a,b,n) with n=10 to find the mean and variance of X^2, saving your (500) mean and variance estimates. Repeat with n=50 and n=100. Use histogram to see the variability in these estimates, and compare this to the 'true' mean and variance answer found by hand. Is n=100 enough? If more, how many more is need to get 'good enough'?