# Assignment #1

This assignment is due on March 19th one hour BEFORE class starts via email to `wallraven@korea.ac.kr`.

If you are done with the assignment, make one zip-file of the `assignment1` directory and call this `<LASTNAME_FIRSTNAME_A1.zip>` (e.g.: `HONG_GILDONG_A1.zip`).

**Please make sure to comment the code, so that I can understand what it does. Uncommented code will reduce your points!**

**Also, please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!**

## Part1 Scripts, cell variables, return values, file handling (20 points):

Make a **script (not a function)** called `<YOURLASTNAME_YOURFIRSTNAME.m>` (e.g.: `HONG_GILDONG.m`). The first lines of the script should contain a comment that explains what the file does. The next few lines should clear the workspace, close all the figures, and clear the output window of Matlab. The goal of the script is to create a **`cell`** variable called `myInfo`. Assign the following pieces of information to this variable:

your firstname (as a string)

your lastname (as a string)

your birth-year (as a number)

your birth-month (as a number)

your birth-day (as a number)

The variable `myInfo` should now have five entries, two strings and three numbers.

Add a command to the script that calculates the number of seconds that you have been alive and assign this value to a **sixth entry** in `myInfo`. Assume (even if you know the exact time on which you are born) that you were born at midnight (12 AM) on your birthday. **Assume that there are no leap years!** To calculate your age in seconds, use the function `clock.` Type `help clock` , so you know how to deal with the output of this command (hint: it's an array!). Also, **do NOT use the function `etime`**, but use your brain to calculate the number ☺.

The script should calculate your age in seconds **every time** you execute it, so do not put in fixed numbers, but use the output of `clock`!

Insert a command that displays your age in seconds in a nice format in the **command window** using `sprintf` (such as: "At the time of calling this script, I was NNNN seconds old.").

The last commands in the script now will need to save the contents of the variable `myInfo` into a **text file** called `<LASTNAME_FIRSTNAME_myInfo.txt>` (e.g.: `HONG_GILDONG_myInfo.txt`).

The text file should get the following structure:

```
firstname

lastname

birth-year / birth-month / birth-day

age in seconds
```

Note, that the file has 4 lines, and that the third line also contains the character "/". The easiest way to achieve this file structure is to do something like:

```
file = fopen('HONG_GILDONG_myInfo.txt','w');

fprintf(file,'%s\n',…)

…

fclose(file);
```

Look up the help for `fopen`, `fprintf` to see more examples!!

## Part2 Functions, statistics, plotting (40 points):

In this assignment you will use a script and a function to calculate statistics of some example data. We have the heart rate, weight and amount of exercise in hours per week of 10 participants. This data is given in the following table.

| Participant | Heart rate(bpm) | Weight (kg) | Exercise (hrs) |
|---|---|---|---|
| 1 | 72 | 61 | 3.2 |
| 2 | 82 | 91 | 3.5 |
| 3 | 69 | 71 | 7.1 |
| 4 | 82 | 67 | 2.4 |
| 5 | 75 | 77 | 1.2 |
| 6 | 56 | 80 | 8.5 |
| 7 | 93 | 101 | 0.1 |
| 8 | 81 | 75 | 3.0 |
| 9 | 75 | 55 | 2.7 |
| 10 | 59 | 65 | 3.1 |
| 11 | 95 | 79 | 1.5 |
| 12 | 66 | 62 | 6.3 |
| 13 | 80 | 75 | 2.0 |

There are several ways to get the data from above into Matlab. One of the fastest ones is to make use of the "`Paste To Workspace…`" menu item found in the "`Edit`" menu in Matlab. So, mark **only the values** above with your mouse, copy the contents, go to Matlab and select "`Paste To Workspace`". Select "Space" as the column separator and give the resulting variable the name `experimentData`. If you can't get this to work, you can always enter the values yourself, like:

```
experimentData=[ [1 72 61 3.2]; [<second row>]; …];
```

Now, we actually don't need the first column of this variable, which has the participant number (think about why!), so delete it from the variable. In the end, `size(experimentData)` should give 13 x 3 as answer.

Save this variable as `experimentData.mat` file.

Now, create a file called `processData.m`, and write its preamble to clear all variables, close all figures and clear the command window. **This is a script.**

Insert a command to load `experimentData` from the file.

Now we will first design a **function** that will calculate **two values** from the data: **mean** and **standard error of the mean (SEM)**. The mean should be clear to everyone, I hope. The SEM gives you an idea of how sure you are about the mean given your measurements. It is related to the standard deviation of the data by the following code:

```
    SEM = std(data)./sqrt(length(data)-1);
```

Create a new file for this new function, which will calculate the mean and the SEM of our data, and call it `meanSEM.m`. Write the **function preamble (see lecture notes)**, so that we know what the function will do.

We are going to calculate the mean and SEM of the data, so the function will need **one** input variable and **two** output variables:

```
    function [output1,output2]= meanSEM(input1)
```

Choose appropriate names for the variables above, keeping in mind not to overwrite existing functions (avoid things like "`mean`", for example!).

There is no data initialization as we are only calculating new data. Now use the function `mean` to calculate the mean of the heart rate, weight and hours of exercise over the participants. If you need help, use the command `help mean` first.

Use the (modified) piece of code shown above to then calculate the SEM of the data.

**Note that the function will work on arrays with multiple columns of data!!**

Now, go back to `processData.m` and write a **function-call** to `meanSEM.m` with the experimental data. With the result, now make a **figure with three subplots** (that is a **single** plot which contains three subplots – for more, take a look at `help subplot`). Each subplot therefore contains one variable plotted (remember, we have three variables, heart rate, weight, and exercise – please also make sure to provide proper axis labels!). So, now plot the **mean** of each data subset into each subplot as a **line**. Plot the **SEM** around the mean of each data subset into the subplot as **two** dashed lines (one above the mean, one below the mean).

Insert a command that saves the resulting figure as `analysis.fig` (to get the current figure in Matlab for saving, use the figure handle `gcf`).

In `processData.m` insert a command to create a **SECOND figure with three subplots**. In the first subplot, plot the heart rate against the weight. In the second subplot, plot the heart rate against the exercise rate. In the third subplot, plot the weight against the exercise rate. Note that these are **scatter-plots, so use the command `scatter`!** Save this figure as `analysis2.fig`.

**Your script will produce two figures, each with three subplots – both figures will need to be displayed and saved to disk in the script.**

**What can you say about these plots, and what is the resulting (cautious) interpretation of the data? Insert a comment at the end of `processData.m` with your observations.**

Code examples:

```
% comments, bloody hell!

function writeMatrixToFile(A)

f = fopen('matrixTxt.txt','w');

[rows,columns]=size(A);

for i=1:rows
    fprintf(f,'line %d %s:
',i,'test');
    for j=1:columns
        fprintf(f,'%g\t',A(i,j));
    end
    fprintf(f,'\n');
end
fclose(f)
```

```
% createMatrix: create a matrix that I
like
% the matrix contains values that...
% input: dimension = number of
dimensions of the matrix
% output: matrix A
% ...
% created by cw 2013/09/11
% last updated by cw 2013/09/11

function [A,dummy] = mean(dimension)

% create dummy variable
mean=zeros(1000);


if nargin==0
    error('please provide number of
dimensions');
else
    % variable A contains the matrix
    A = zeros(dimension);
end

return;
```