**Exercise #2 – Matrix Manipulation**

Tutor in charge of this HW: **Shir Maimon.** Questions will be answered through the Moodle forum or at the reception hour (Sunday 13:00-14:00).

Read **carefully** the following instructions, and don’t hesitate to reach us (preferably through the forum) if something is not clear.

**Moodle instructions:**

1. Before submission, you need to create a team through the “Team Work” window. You can team with different students for every HW, but even if you don’t change your team – you still need to do this step for every assignment.
2. Submit the .m file directly to the Moodle. Only one member of the team need to submit – but **all** of you should make sure that you see your submitted file in your Moodle!

**General Matlab instructions:**

1. **Do not submit a script that does not run properly**. To check that your script run without errors, make sure your workspace is cleared, and then run the script (press F5). If you are reading data, avoid full paths (“C:\My\_computer\HW\data.mat”), as they will return an error when you run the code on a different machine (i.e. the computer of the TA who checks your HW ☺).
2. Name your script "hw2\_< ID number 1>\_< ID number 2>.m".
3. Start your script with a header – a few lines of comments which describe your script. The header should contain **all** of this information: your names, IDs, HW number and the Matlab version (e.g. (R2021a) you wrote your code on.
4. Divide your code into **sections** according to the questions, or any other reasonable division (use the '%%' sign). At the beginning of each question, write its number and a short explanation of your answer – e.g. what the code does, etc.
5. If the question involves printing a result to the console, you should **also** print the question number: You can use “*disp('\*\*\* Question ## \*\*\*')*”.
6. **Document** your work with comments in the script (use the '%' sign). Over-documentation is better than under-documentation: we want to see that you understand how your code is working at each step.
7. Give your variables **meaningful names**.
8. For every question that has a defined output (a number, a figure, etc. – not an “open question”), **read carefully** the HW instructions and make sure that your output is **exactly** what we asked, and what you would expect it to be. This is very important also for you in your “debugging” procedure. For example, if you expect a scalar output but your code returns a matrix (and vice versa), you clearly have a mistake somewhere… ☺
9. **Do not** use "magic numbers" – **every** number that appears in your script should be assigned to a variable, preferably at the beginning of the code section.
10. Do not calculate anything implicitly, but make your code calculate everything explicitly. For example, if you have a vector with the values [2, 5, 3, 1] and you need to find its largest value, you **cannot** write “*disp(5)*”.
11. **Avoid** “code duplication”! Code duplication is copying & pasting lines of code with minimal changes. Instead, use matrix and vector operations, and loops.

**Specific instructions for this HW:**

1. This HW focuses on matrix and vector operations. The idea behind these is to perform multiple calculations with very few lines of code. As this is one of the most important practices in Matlab, we want you to demonstrate that you can use this **without using loops and without code duplication**.
2. Every numerical answer needs to be printed to the console. For most questions you can use either “*disp*” or “*fprintf*”, but make sure that you understand how to use “*fprintf*” with **vector** inputs, as it is needed in some of the questions.

In this exerciseyou will use a dataset of test scores of different individual mice. The matrix ‘*results*’ is a 30x10 matrix, where each row corresponds to a single animal, and each column corresponds to a particular day (each one of the 30 animals was tested for 10 days). In the next questions you will need to demonstrate correct indexing of this matrix, and perform matrix operations on the correct dimensions. This exercise deals with matrix and array operations. Most questions can be answered using only 1-2 lines of code, without copy-pasting the same calculation over and over, simply by using matrix operations. You are **not allowed to use loops**. You are expected to give full answers, which are displayed to the console (use “*disp*” or “*fprintf*”).

Load the file “data\_hw2\_2023.mat” into your workspace. Note that your script should run on any machine. Therefore, you should not use full path (**don’t do: C:\XXX\YYY**), but load the file from the current directory (and make sure the file is located in the current directory, and you set the current directory correctly).

1. Calculate the average (*mean*) values of all days (columns). Which day had the highest **average** test score? Example answer: “*Day XXX had the highest average score*”.

**Hint:** Read the documentation of the “*max*” function. You can do this either by searching the full documentation (clicking the "?" icon), running the command “doc max” or highlighting “max” with your cursor and pressing F1. What happens when you ask the function to return two outputs?

1. Repeat question 1 with only the valid results indicated in the variable valid\_results. ‘1’ means a valid result and ‘0’ is invalid.
2. Which day has the highest test score (i.e. a single score, not the average you calculated earlier)? What is this value?

**Hint:** There are two ways to calculate this:

1. Calculate the highest scores for each day (column), then find the maximum of these scores and its index.
2. Calculate the highest score in the entire matrix, and use “*ind2sub*” to find which day it belongs to.
3. Find the 4 animals with the lowest average test scores. For each one of these animals, display its number and its average score (rounded down). Your answer should look like this:

*Animal XXX had average score YYY*

*Animal XXX had average score YYY*

*Animal XXX had average score YYY*

*Animal XXX had average score YYY*

1. Calculate the standard deviation of the scores for each animal. Then, calculate the standard deviation of the scores for each day. Display the mean of the standard deviations in each case, and also the larger of the two. Your answer should look like this:

*Mean of day s.d: XXX*

*Mean of animal s.d: YYY*

*Larger mean s.d: XXX / YYY*

1. Which animal showed the smallest average improvement? **Hint**: First, calculate the improvement of each animal across days by using the function “*diff*” (make sure you compute it based on the correct dimension – see the documentation of the correct inputs for this function), and then calculate its average.
2. In this question, you need to display to the console the number of every odd animal (1, 3, 5, etc.). If you did not use it already, this is a great time to learn how to use the function “*fprintf*” with vector inputs. Read carefully the documentation and experiment a little until you understand how it works (what does ‘\n’ do?). Your code should display this to the console:

*Animal 1*

*Animal 3*

*Animal 5*

*…*

*Animal 29*

1. In this question we will use the vector ‘*norm\_day\_coeff*’, which stores normalization values between days. For each animal, print its total normalized test scores, rounded up. These normalized scores are the sum of the test scores of each day multiplied by the normalization coefficients. In other words, multiply the data from day 1 with the normalization coefficient for day 1, day 2 data with day2 coefficient, etc., and sum all the scores together. Your result should look like this:

*Animal 1: XXX*

*Animal 2: XXX*

…

*Animal 30: XXX*

**Hint**: Remember that matrix multiplication (‘\*’) is defined as for a matrix A with *n* columns, and a matrix B with *n* rows. Another option is using element-by-element multiplication (‘.\*’), which multiply every element of one array with the corresponding element from another array.

1. In this question we will use the vector ‘*experimental\_group*’ to divide our data into two groups. The numbers “1” and “2” in this vector indicate that the animal either received a treatment (group 1) or did not receive a treatment (group 2). Calculate the mean test score across days separately for each group. Which group has higher average scores for each day?

Your answer should look like this:

Day 1: group XXX had higher mean score

Day 2: group XXX had higher mean score

…

Day 10: group XXX had higher mean score

1. Sort the ‘results’ matrix by the results of the last test day (last column), from the highest scores to the lowest scores. All the rows of the matrix should be sorted together – make sure that you don’t scramble the matrix by mistake! Save the sorted matrix in a new variable.

**Test yourself:** After sorting, the mean of the first row (rounded to 2 decimal digits) should be 13.89.

**Hint**: use the option of ‘*sort*’ with two outputs, and use the 2nd output as an indexing variable.