TOWARDS DATA MINING

EXERCISE 1. INTRODUCTION TO Matlab

1. GETTING YOUR DATA INTO Matlab

To get started, you have to read your data from the file. Let's work with the file `concretedata.csv`.

First, if you have any commas used as decimal separators you need to replace them with decimal points.

```
comma2point_overwrite(concretedata.csv');
```

Note that a semicolon avoids the output of a Matlab command.

You can read the data with for example the `readtable` command:

```
mydata = readtable('concretedata.csv','Format','%u%f%s%s%f%f%f%f%f%f%u%f');
```

With additional arguments, you can add more details. Do you have variable names in the first row? What if there are missing values? How they are indicated?

```
mydata2 = readtable('concretedata.csv','Delimiter', ';' , 'ReadVariableNames',
false);
mydata3 = readtable('concretedata.csv', 'TreatAsEmpty',{'.','NA','N/A'});
```

Notice that '%' is used for comments in Matlab!

You can see your data in the workspace window. If there are objects that you do not need, you can remove them with

```
clear mydata2;
```

When you end your work, you can save your entire workspace

```
save('workspace.mat');
```

or selected objects to a file.

```
save('testfile.mat', 'mydata3');
```

And when you are ready to work again, you can load them back simply by writing:

```
load('workspace.mat');
```

If you want to save a data table to a .txt file, use the command

```
writetable(mydata,'mydata.txt');
```

2. SIMPLE DATA ANALYSIS

Now look at your data more closely. First, check that your data is data table type:

```
istable(mydata)
```

You can find the dimensions of a data table

```
size(mydata);
```

This command gives you first 6 rows of the data

```
head(mydata)
```

You may want to see more

```
head(mydata, 10)
```

and similarly the last lines

```
tail(mydata)
```

With this command you see the data types of each variable

```
summary(mydata)
```

You can get the variable names of your data table using

```
mydata.Properties.VariableNames
```

If you did not have the names, you can enter them manually. Let's make a data table containing only zeros and use the names of mydata:

```
mymatrix = zeros([10 12])

mymatrix = array2table(mymatrix)

mymatrix.Properties.VariableNames = mydata.Properties.VariableNames
```

When you data is data table type, you can refer to each variable with their name by writing

```
Mydata.water
```

3. BASIC STATISTICS

You can calculate the correlations between 2 variables in your data set with

```
corr(mydata.cement, mydata.coarse_aggregate)
```

If you have missing values in your data, you may need the `'omitnan'` argument. It removes the missing values while using the selected function.

You can calculate the mean for one variable (vector) with

```
mean(mydata.cement, 'omitnan')
```

You can calculate means or medians for the data table, and select the dimension 1 for column wise and 2 for row wise.

```
median(mydata{:,[1:2,5:12]},1)
```

```
median(mydata{:,[1:2,5:12]},2)
```

Do you want to learn more about the function, or maybe you need help with it?

```
help median
```

4. SELECTING DATA

When you want to select (or view) only specific columns, you can refer to their dimensions

```
mydata(:,2:4)
```

Note that you need to specify a vector with [] if you do not want all the columns between 2 and 4

```
mydata(:,[2,4])
```

What happens, if you select all observations with value >20 for variable `age`?

```
mydata.age>20
```

Compare the result to this one:

```
mydata.age(mydata.age>20)
```

You can continue by selecting both rows and columns

```
mydata(mydata.age>20,[2:4])
```

or refer to columns by their names

```
mydata(mydata.age>20,{'age', 'cement'})
```

Other way to do it:

```
age20 = mydata.age>20;
```

```
cols = {'age', 'cement'}
```

```
mydata(age20, cols)
```

Maybe you need only observations that have the maximum value of variable `age`

```
mydata(mydata.age==max(mydata.age),:)
```

5. DATA VISUALIZATION

Next you can produce tables or visualizations from the data. For categorical data, you may find the tables to be useful. For single variable

```
catdata = mydata.grade; % a categorical variable

tabulate(catdata)
```

For continuous variables, plot visualization is more useful

```
plot(mydata.water)
```

Scatter plots may show interesting relationships

```
scatter(mydata.water, mydata.age)
```

You may add more information to the figure

```
title('Water vs. age')

xlabel('water') % x-axis label
ylabel('age') % y-axis label
```

Attach this figure to your exercise report. You can save it by selecting File -> Save as… on top the figure window, or simply by taking a screenshot and adding it to your document.

There are lots of graphical parameters you can set in the figure using Plot Tools (icon on the top right corner of figure window).

For more plot types, see

```
https://se.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html
```

Lastly, tell about your previous experience with Matlab.