```
clear;close all;clc;
```

Workshop Notes and Solutions for MATLAB Data Manipulation & Processing-08/05/18

Created by Doruk Usluel

Based on the Data Processing and Visualization Course

Datasets obtained from: "Teachers and Professors". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/teachers-and-professors' [Online Resource]

Introduction

In this workshop we will go through data manipulation techniques as well as pre-processing by introducing advanced methods for uploading, slicing, dicing and analyzing the data. There will also be some smart smart tips and tricks to make life easier when working with datasets.

As we are in an academic institution, we will be analyzing a real dataset on teacher quality and quantities around the world (citation given above), but you will be able to apply the methods easily to your own data.

> More datasets can be found on the GitHub repository at: https://github.com/awesomedata/awesome-public-datasets#machinelearning

1. Importing Data into MATLAB

1.1 Table Form

We are going to use the readtable function to transfer our data into MATLAB wich will give us a nice representation of the data in table format.

```
teachNum_acrossLvls = readtable('number-of-teachers-across-education-levels.csv');
% Observe the data and look at the different headings
% Observe what happemned to the empty fields of the data
```

You can access specific variables in the table using the dot (.) notation.

```
E_teachNum = teachNum_acrossLvls.Entity;
Y_teachNum = teachNum_acrossLvls.Year;
Tertiary_teachNum = teachNum_acrossLvls.TertiaryEducation_number_;
```

As an example lets plot a scatter plot to see how many teachers were available in each country in each year.

```
scatter(Y_teachNum,Tertiary_teachNum)
```

Now we will try and import another set of Data to see what changes we might have to make for some data presentations. Import the 'pupil-teacher-ratio-for-primary-education-by-country' file.

```
pupil2teachratios = readtable('pupil-teacher-ratio-for-primary-education-by-country.csv');
% Observe the header of the created table and the first row, Why has this happened?
```

```
% You can look at the previous document in txt form to see the difference.
```

To get over this problem extra inputs can be placed into the readtable function.

```
pupil2teachratios = readtable('pupil-teacher-ratio-for-primary-education-by-country.csv','Heade
```

By defining the HeaderCount number parameter, we were able to select the correct line to start reading the data. This is just one example of the available 'Name-Value' pairs for the 'readtable' function.

Another 'Name-Value' pair is the 'CommentStyle'. We will see how this works after importing the 'share-of-teachers-in-primary-education-who-are-trained' file.

```
share_TrainedTeachersPrimaryEd = readtable('share-of-teachers-in-primary-education-who-are-trai
% observe the lines where the country name changes for some of the topmost countries.
% What are the common characters preceding every extra line
```

To get rid of certain lines (usually intermediate headers in data) with specific characters in the beginning you can set the 'CommentStyle' parameter.

```
share_TrainedTeachersPrimaryEd = readtable('share-of-teachers-in-primary-education-who-are-trai
```

If your data file was not a CSV but a tab delimited file you would change the 'Delimeter' parameter to '\t' as a 'Name-Value' pair.

We will now delete all our variables to try out an advanced data importing tool "Datastores".

```
clear;close all;clc;
```

1.2 Datastores

In MATLAB it is possible to utilize "datastore"s in order to import data or parts of data from multiple files.

In order to avoid using repetitive data import functions you can use datastores. Datastores are essentially references to one or multiple data locations. From this reference you can import either all data or parts of all data and you can customize every type of import to compensate for the different types of data.

The three steps to using a datastore are:

1. Create a datastore
2. Adjust custom parameters
3. Read data from the datastore

Creating a Datastore

Try to create a datastore object which is a reference to only one file using the datastore function.

```
ds_teachNum = datastore('number-of-teachers-across-education-levels.csv')
% Observe the properties listed in the command window for the
% datastore object or click on the variable to observe the full set of parameters
```

After creating a datastore the "preview" function can be used to observe the first few lines of data that is referred to by the datastore.

```
preview(ds_teachNum)
% Observe the header lines of the table, first few lines, any missing values etc
```

Modifying Datastore Parameters

Create two other datastore objects for the two other files we imported previously and modify the corresponding paramaters for each.

```
% File containing pupil to teacher ratio
ds_pupil2teach = datastore('pupil-teacher-ratio-for-primary-education-by-country.csv','NumHeade
preview(ds_pupil2teach)
% File containing Trained Teacher in Primary Education
ds_trainedteachprim = datastore('share-of-teachers-in-primary-education-who-are-trained.csv','
preview(ds_trainedteachprim)
```

You can also change parameters of the datastore using the 'dot' notation after creating the reference.

Now try creating a datastore element that refers to all three of the files.

```
DATA = datastore('Files')
```

Reading Data from Datastore

The "read" and "readall" functions are used to read from a datastore.

Try using the "read" function to create the teachNum_acrossLvls variable we created earlier already stored in one of the datastores.

```
teachNum_acrossLvls = read(ds_teachNum)
```

Now try doing the same thing for the pupil2teachratios variable but change the 'ReadSize' property of the relevant datastore first, using the 'dot' notation.

```
ds_pupil2teach.ReadSize = 1000;
pupil2teachratios = read(ds_pupil2teach) % observe the contents and number of elements in the r
% Not all the data is read, try it again and observe the output
pupil2teachratios = read(ds_pupil2teach)
% To get back to the beginning of the file referenced in the datastore use reset
reset(ds_pupil2teach)
% Make the readsize larger and read the file again
ds_pupil2teach.ReadSize = 20000;
pupil2teachratios = read(ds_pupil2teach)
```

Now we are going to try and create all the previous data holding variables in section 1.1 by reading from the DATA datastore which holds all three of the data files.

```
% Observe the readsize of the datastore
DATA.ReadSize
```

```
% Try reading the first file
teachNum_acrossLvls = read(DATA)
% Try reading the second file but you will have to change a parameter
DATA.NumHeaderLines = 4;
pupil2teachratios = read(DATA)
% Do a similar operation for the 3rd file
DATA.NumHeaderLines = 0;
DATA.CommentStyle = '>>'
share_TrainedTeachersPrimaryEd = read(DATA)
```

2. Cleaning the Data

As we observed earlier there can be missing or corrupted data in our datasets. We want to be able to work around or eliminate this problem in order to automate mathematical and computational analysis on our data.

e.g. If you try to take the mean of a dataset with even one NaN (not a number) value in it the answer will also be NaN.

2.1. Working around the NaN values

Clean the workspace and obtain the Tertiary teacher numbers again by importing the 'number-of-teachers-across-education-levels.csv' file and extracting the relevant parameter.

```
clear; close all; clc;
teachNum_acrossLvls = readtable('number-of-teachers-across-education-levels.csv');
Tertiary_teachNum = teachNum_acrossLvls.TertiaryEducation_number_;
```

Try to take the mean of the variable keeping the number of tertiary teachers in all countries for every year:

```
mn_Tertiary_teachNum = mean(Tertiary_teachNum)
```

As expected the result is NaN due to the NaN values in the dataset.

The 'omitnan' property for the mean could be used.

```
mn_Tertiary_teachNum = mean(Tertiary_teachNum,'omitnan')
max(Tertiary_teachNum)
```

The 'median' function has the same property.

'max' and 'min' functions have the omitnan property by default. Other mathematical operations should also be checked if they do or do not have a omitnan option.

An example of a function that does not have a omitnan property is the 'isequal' function.

```
% Every array should be equal to itself
isequal(Tertiary_teachNum,Tertiary_teachNum)
```

Because of the NaN values are not considered to be equal to any value the result is given as '0' or 'not equal'.

To compensate for the NaN values another function 'isequaln' ahould be used.

```
% Every array should be equal to itself
isequaln(Tertiary_teachNum,Tertiary_teachNum)
```

2.2. Locating and Eliminating Missing Data

Not every function or application will make it easy to workaround the missing data and NaN values, therefore it is cinvenient to know where and how many missing values are in the data.

The 'isnan' function returns an array wich correspond to its input with 1's instead of the place for each NaN value and 0's for other values.

```
% Use isnan on Tertiary_teachNum and observe the output
nan_idx_TerTeachNum = isnan(Tertiary_teachNum)
```

After using 'isnan' the function 'nnz' which gives the number of non-zero elements, could be used to obtain the number of NaN values in the set.

```
% Find isnan and nnz to find the number of NaN values in Tertiary_teachNum
nnz(nan_idx_TerTeachNum)
```

One option to deal with the NaN values is to replace them with a non-conventional/unexpected value such as a negative number in a teacher count dataset...

As the isnan function gives a logical array it can be used as an index of the NaN values.

```
% Replace NaN values with a negative number
Tertiary_teachNum(nan_idx_TerTeachNum) = -1
% or the locations with NaN values can be deleted all together by equating to an empty array
Tertiary_teachNum(nan_idx_TerTeachNum) = []
```

There are other types of missing data according to data types:

1. NaN (Not a Number) - for number data
2. NaT (Not a Datetime) - for date time data
3. undefined - for categorical data *more on categoricals further in the workshop

Instead of 'isnan', 'ismissing' can be used to locate all of these types similarly.

3. Managing Data

3.1. Categorical Data

Sometimes you might want to categorize your data into smaller sets or groups with certain properties or values. In order to achieve this in a simple manner categorical data can be used.

Also, using categorical arrays can reduce the memory space used by MATLAB as well as allowing for further applications.

Obtain the Entity parameter from the imported the 'number-of-teachers-across-education-levels.csv' file

```
E_teachNum = teachNum_acrossLvls.Entity
```

Turn the entities of the teachNum_acrossLvls parameter, held in the E_teachNum variable to a categorical array. Compare the memory size of the data using the 'whos' function.

```
whos E_teachNum
E_teachNum = categorical(E_teachNum);
whos E_teachNum
```

Start by finding the assigned categories in the recently created categorical E_teachnum variable, using the categories function:

```
cats = categories(E_teachNum)
```

One of the Categories is called world. Say we want to remove this category to observe countries individually. Try removing the world category using the "setdiff" function.

```
cats_countries = categorical(setdiff(cats,'World'))
```

You notice that there are some categories repeated such as "Europe & Central Asia" merge the repeated categories using the "mergecats" function on the original categorical set:

```
cats_countries = categories(mergecats(cats_countries,{'Europe & Central Asia' 'Europe & Central
```

3.2 Discretizing Data

Now create a categorical array by converting the cleaned number of primary education teachers in each country obtained from the teacher count dataset:

```
% Obtain the number of primary teachers in each country in each year
Primary_TeachNum = teachNum_acrossLvls.PrimaryEducation_number_
% Now assign levels and borders to determine bin intervals (create 4 bins/5 levels)
lvls = linspace(min(Primary_TeachNum),max(Primary_TeachNum),5)
% Use the discretize function to place each value into a bin
TeachNumBins = discretize(Primary_TeachNum,lvls)
```

You can add Inf or -Inf to the lvls (bin edges) to extend the discretization to values that lie outside the edges

Now we can categorize the bins into categorical values.

```
% Select 4 category names as we have 4 bins
catnames = {'too few', 'below par', 'above par','too high'}
% Use the categorical option of the discretize function to assign categories to the bins
TeachNumBins = discretize(Primary_TeachNum,lvls,'categorical',catnames)
% Since the categories relate to numerical values you can search for the number of elements tha
% Find the number of elements that have a teacher number categorised as below par or less
lessBelowPar = nnz(TeachNumBins<='below par')
% How would you get the indices of the variables that had teachers less than below par?
```

```
clear; close all;clc;
```

4. Analyzing Groups in Data

It is possible to find groups in MATLAB. This is a good reason to create categorical or discrete variables from your data.

First import the 'number-of-teachers-across-education-levels.csv' data file using either readtable or by creating a datastore

```
teachNum_acrossLvls = readtable('number-of-teachers-across-education-levels.csv')
```

Use the "findgroups" function to find the different groups of countries in the data

```
countryGrpNums=findgroups(teachNum_acrossLvls.Entity)
```

The variable contains a groupNumber for each element in the set and every different country name has a different group number. To obtain the values of each groupnumber, another output can be assigned to the "findgroups" function

```
[countryGrpNums, countryGrpVals]=findgroups(teachNum_acrossLvls.Entity)
% Observe the number of elements in both outputs
```

To obtain the number of elements in each group you can use the "histcounts" function. Find how many elements are in each group.

```
% Use histcounts, make sure the second input (bin number) is equal to the number of groups
countryCount = histcounts(countryGrpNums,233)
```

Do the same procedure to determine the groups in the years of observation across all countries

```
[YearGrpNums,YearGrpVals] = findgroups(teachNum_acrossLvls.Year)
% Find the bin counts
YearCount = histcounts(YearGrpNums,46)
```

You can also create groups depending on more than one variable in a dataset. Try to find groups (using findgroups) depending on both the country names and the year of observation:

```
[CntryYearGrpNum, CountryGrpVal, YearGrpVal] = findgroups(teachNum_acrossLvls.Entity,teachNum_a
% Observe how many different country year groups exist
```

Once you have grouped your data in different ways, you can apply certain operations on ony groups of data

Use "splitapply" to apply a function to only a group of data.

Lets say you want to take the mean value of all the teacher numbers everywhere in each year

```
% The seperate group numbers of each year is in the YearGrpNums variable,
% Take the mean of every value obtained for the number of teachers in thirtiary education
% in each year around the world using splitapply
```

```
omeanFcn = @(x)(mean(x,'omitnan'));
TerTeachMeanYears = splitapply(omeanFcn,teachNum_acrossLvls.TertiaryEducation_number_,YearGrpNu
% Try doing the same for each country
TerTeachMeanCntrys = splitapply(omeanFcn,teachNum_acrossLvls.TertiaryEducation_number_,countryG
% What is the country with the lowest mean teacher number in tertiary education across the year
[~,idx] = min(TerTeachMeanCntrys)
countryGrpVals(idx)
```

Finally lets visualise our group data using a bar chart.

```
% plot a bar graph of the years of observations vs. the mean number of tertiary teachers across
bar(YearGrpVals,TerTeachMeanYears)
```