Res.Search MATLAB session

Learning Objectives:

- Learn how to read in data from a spreadsheet.
- Learn how to ask questions of the data.
- Learn how to answer questions using grouping, subsetting and visualising the data.

Part 1: Importing Data

Reading in data

We can read in data from a spreadsheet using the import tool. In the home tab, click on the Import data icon. You can select the file you want to import and the import tool opens up an excel like spreadsheet. In this part, we can choose the kind of output the tool will give us and what to do with missing values.

Alternately, if you have excel installed on your computer you can read the xlsread function. xlsread has an option to import both numeric and text data. Here is an example of howyou would read in data using xlsread.

```
[~,~,data]= xlsread('pedestriancounts_melbourne.csv')'
```

The resulting matrix 'data' will be a cell array with a combination of strings and numbers. The '~' ignore the first two outputs of the xlsread function which are the numeric and strings values separately. The first output is a numeric matrix and the second output is a cell array.

If you need more help about xlsread, see here: https://au.mathworks.com/help/matlab/ref/xlsread.html

If you want more options for reading in spreadsheets into MATLAB, see here: https://au.mathworks.com/help/matlab/spreadsheets.html

Dealing with missing data

While using the import tool, there is an option to replace missing data with anything we want. For example if there are missing cells in a dataset, we can replace them with 0s or 1s or 25s. MATLAB automatically imports missing data with 'NaN' which stands for not a number. If importing text data in to a numeric array, we will also get NaNs.

Once data is imported, we can deal with NaNs using conditional indexing. The 'isnan' command returns a logical 0 or 1 depending on whether the value in the cell is a NaN or not.

For example:

```
ped_counts= cell2mat(data(:,end)); % making the last column of data equal to a new vector ped_
% The numeric data is in a cell. We need to convert this into double by using the cell2mat fur
ped_counts(isnan(ped_counts))=0;
```

Mini-Challenge

In your groups, convert all the NaNs in your data to your favorite number. Make sure your number is not one that is likely to occur in your data.

Discussion- what do NaN mean for your analysis?

Also try **ismissing, rmmissing** to deal with missing data

Have outliers in your data? Try **isoutlier** and **filloutliers**

Looking at data

The Workspace

In the workspace, where data is stored in MATLAB's short term memory, a summary of the dataset can be readily available if we want it to be. We can choose the columns that are visible, which can include the size, the minimum and the maximum values.

Descriptive statistics

We can also look at the above mentioned data by directly getting the values from the array.

See what the following functions do.

```
max
min
size
length
```

See here for more commands you can use: http://au.mathworks.com/help/matlab/descriptive-statistics.html

Part 2: Framing a research question

What are all the things we need to make a research question?

First of all, what are the variables available to us?

```
whos % displays the variables in the workspace
```

For doing the kind of analysis we want to do, we want to subset data or group by variables. For the grouped data, we also want to calculate some descriptives. These groupings will entirely depend on the research question we want to ask. So before we do anything else, break in to groups and figure out what question you want to ask!

We have a date variable. We will can use datevec to split our date variable to years, months, day, time.

```
datevector= datevec(Date);
```

Say I want to ask the question, which month has the largest number of pedestrians. I would have to group my variables by month. We can use regular MATLAB conditional vectors to get the monthly counts out.

```
for n=1:12;
monthlycounts(:,n)={Pedestrian_Counts(datevector(:,2)==n)};
end
```

We can then get the mean by adding an extra line of code in the above for loop.

```
for n=1:length(unique(datevector(:,2)));
mean_monthlycounts(n)= mean(monthlycounts{:,n})
end
```

This method has the advantage that we get the subsetted data. We can plot the data. If this isn't something we are interested in, we can use the splitapply function

```
mean_monthlycounts= splitapply(@mean, Pedestrian_Counts, datevector(:,2));
```

We now have the mean values. We can look at the minimum and maximum of the monthly averages using one of the descriptive stats functions earlier.

Part 3: Visualising data to answer our questions

The easiest way to visualise data is plotting the data. We can plot the mean of the monthly counts using either the Plots tab or the plotting function.

```
h=plot(mean_monthlycounts);
```

Once here, we can use the interactive plotting environment to add and remove titles and axis labels, set axis limits and export code.

Another means of visualising data: Histograms.

```
figure; hold
for n=1:12
    h(n)= histogram(monthlycounts{:,n});
end
```

Challenge

Now try and test your hypothesis in your groups.

What did you find?

Conclusion

In this part, we learn how to:

- Import data into the MATLAB environment both interactively and from the command line.
- Deal with missing data to perform analysis better.
- Get summary statistics
- Convert dates into date vectors
- Frame meaningful research questions to interrogate the dataset.
- Group data according to the dataset.
- Plot the data and edit plot properties.

- Test a hypothesis.

If you want to perform some basic stats, discuss it with our stats consultant - TIM RICE

If you need more help: Mathworks, StackExchange, Hacky Hour (You can also come along to work at hacky hour).

Come to one of the MATLAB trainings/meetups (Watch this space)

Join MATLAB@Unimelb.

Contact me at yamnimohan@gmail.com, tweet @YamniMohan.