Part One: Getting the Data

This lesson was adapted from the stats extension course by PJ Karoly. The lesson can be found here: https://resbaz.gitbooks.io/resguides-matlab-statisticstoolbox/content/

Folders and Files

In MATLAB you navigate between files using strings

```
% Strings are defined using apostrophes
str= 'Goodbye Cruel World'

% File names are defined using strings
file_name = 'MyData';
```

```

```

Navigating Folders

The function `cd` stands for 'change directory', and is used to set the current workspace.

```
% get to where you want to be
cd('D:\Resbaz\statsextension\MATLAB-StatisticsToolbox-master');

% If you want to navigate backwards, you can type this instead-

cd ..

% What are the files in the directory?The function dir(file_name) returns a list of files insi

file_name=dir('MyData')
```

The function `dir` returns the files in the folder as a structure. A structure is like a tree. The main trunk is the structure, the major branches form the major attributes. The folder structure is a twig. We can access the individual fields using the '.' notation.

```
file_name.name % This displays all the folders
% We can find individual files. For example, we can find the first item in the folder.
file_name(1).name
% We can use this to then change the folder
cd MyData
cd (file_name(3).name);
```

Reading in files

Filenames

If you're looking for particular file names the functions 'string compare' (`strcmp`) and 'string find' (`strfind`) can be useful!

Dealing with strings is not as easy as working with numbers. For instance, you cannot simply use == to tell you if two strings are the same. Try typing the following commands into MATLAB to see why.

```
'hello' == 'hello'
```

```
'hello' == 'goodbye'
strcmp('hello','hello')
```

We can use `strcmp` to only load certain files, or to find the index of a particular file

```
file_number = find(strcmp({Data.name},'Patient1'))
```

Challenge One

```
% CHALLENGE
% Work out the difference between strcmp and strfind
```

We can read in excel files into MATLAB using the `'xlsread'` function. For example, we can read in the `'data1.xls'` like so.

```
data1= xlsread('data1.xls');
```

We can use `'dir'` to find all the files in the folder agaim

```
filenames= dir;
```

Let's get the first filename.

```
first_file= filenames(3).name;
```

Then we can read in the file using the variable.

```
cd('Patient1')
data1= xlsread(first_file);
```

Challenge Two

```
% CHALLENGE
% Use a loop to read in the first file in each folder
folders= dir('MyData') % Give the path to the folder
for n=3:length(folders) % Initiate for loop to skip through each of the directories
    %in folders
    foldername= folders(n).name; % get the foldername
    cd (foldername) % change directory to requisite folder
    files= dir; % get all the folders in the current working directory
    filename= files(3).name;  % Get the first filename
    data= xlsread(filename) ; % Read in the first file
    cd .. % Go back to previous folder
end
% EXTENSION
% use two loops to read in all the files in all the folders
```

Regular Expressions

If you need to get particular dates, numbers or names from a file name the 'regular expression' function (`regexp`) might help. For instance, let's say you only want to load data from measurments that were taken after a certain day. Using `strcmp` that would require an `if` statement that listed every possibility:

```
% looping through lots of files
% for n = 1:1000
%
%   % get the file name
%   file_name = Data(n).name
%  [str, first, last]= regexp(file_name, '\d', 'match')
% day_number= {str}
%
%   % using strcmp check if the file is one that we want
%   if
%                   % this could go on forever .....
%
%     % load the file
%     load(file_name)
%
%   end
%
% end
```

Instead, we can use `regexp` to just pick out the part of the name we are interested in - i.e. the number. In MATLAB format strings and numbers are represented by `\d`.

```
[first,last] = regexp(file_name,'\d')
```

This command will just return the first and last indices in the string `file_name` that contain a number. This makes it easier to write a simple `if` statement that works for every file name:

```
% looping through lots of files
for n = 1:1000
% get the file name
file_name = Data(n).name;
[first,last] = regexp(file_name,'\d');
day_number = file_name(first:last)
% using strcmp check if the file is one that we want
if day_number >= 15
load(file_name)
end
end
```

NB: to keep numbers together using `regexp` use `'\d+'` instead of just `'\d'`. Try it out to see the difference:

```
regexp('Patient104','\d')
```

or

```
regexp('Patient104','\d+')
```

Challenge Three

```
% CHALLENGE
% Can you use regexp to find:

% 1) The index at the at start of the word
% 'cat' in the sentence 'the cat sat on the mat'

string = 'the cat sat on the mat';
pattern = 'cat';
regexp(string,pattern)
```

```
ans = 5
```

```
% 2) The indices at the start of the words 'cat' and 'mat'

pattern = '[cm]at';
regexp(string,pattern)
```

```
ans =
     5    20
```

```
% 3) The words that start with 'c' and end in
% 't' in the following list

word_list = {'cat', 'mat', 'hat', 'cot'};
pattern = 'c\w*t'
```

```
pattern =
'c\w*t'
```

```
regexp(word_list,pattern)
```

```
ans = 1×4 cell array
    [1]    []    []    [1]
```

```
% EXTENSION
% Can you write code that only extracts the day (as a number)
% from the following list of patient files

file_list = {'Pt1Day14', 'Pt12Day102', 'Pt009Day9'}
```

```
file_list = 1×3 cell array
    'Pt1Day14'    'Pt12Day102'    'Pt009Day9'
```

```
pattern='\d+';
all_num=regexp(file_list, pattern, 'match')
```

```
all_num = 1×3 cell array
```

```
    {1×2 cell}    {1×2 cell}    {1×2 cell}
```

```
for patient=1:length(all_num)
days(patient)= [all_num{1,patient}(1,2)];
end
```

Converting Data

Usually you need data to be numerical, although often it comes in all sorts of formats. MATLAB has some good inbuilt functions to work with dates, such as `datevec` and `datenum`. `datenum` converts a date to a number of days, and `datevec` converts it back. These are very helpful for sorting and organizing data.

```
% standard syntax is datenum(year,month,day)
datenum(2012,2,1)
```

```
ans = 734900
```

```
% but lots of things work
datenum('1 Feb 2012')
```

```
ans = 734900
```

```
% and if you're worried you can always
% specify the format
datenum('01~02**2012','dd~mm**yyyy')
```

```
ans = 734900
```

```
% it will accept time as well (hours, minutes, seconds)
datenum(2012,2,1,8,30,0)
```

```
ans = 7.3490e+05
```

`datenum` converts a date to thenumber of days since 00/00/0000 (so if your data is older than Christ you might be stuck). Then these numbers can easily be sorted into chronological order and converted back to a date using `datevec`

```
% datevec returns a date as a vector
date = datenum('Oct-21-2015');
[year,month,day,hour,min,second] = datevec(date)
```

```
year = 2015
month = 10
day = 21
hour = 0
min = 0
second = 0
```

We can combine these functions with `regexp` to extract and use dates from file names.

```
data = dir('House Price Data');
for n = 1:length(data)
file_name = data(n).name;
[s,e] = regexp(file_name,'\d\d-\d\d-\d\d\d\d');
if ~isempty(s)
date = file_name(s:e)
end
end
```

Challenge Four

```
% CHALLENGE
% loop through the files and read in the three excel datasheets
% save each dataset as a matlab file called HousingPrices_01,
% HousingPrices_02, etc in chronological order (ie you will need
% to read and use the date from the file name)
data=dir('HousePriceData');

for n = 1:length(data)
   file_name = data(n).name;
   [s,e] = regexp(file_name,'\d\d-\d\d-\d\d\d\d');
   if ~isempty(s)
      date=  file_name(s:e)
      date_number(n)= datenum(date)
   end
end


% HINT: the function datenum is very useful for sorting dates into
% chronological order
```

Challenge Five

```
% CHALLENGE
% the variables in HousingPrices need to be converted to something we can
% work with
% the column called 'Sold at Auction' contains variations of yes and no and
% n/a. It is sometimes blank (which is the same as n/a). Convert this
% variable into 0 = no, 1 = yes and -1 = n/a.
```