Read files, determine the means, the mean of the mean and the standard deviation:

```
clear all

A = [importdata('test.xlsx') importdata('test2.xlsx')] ;
% add files if needed but excel files need to be same length!!!!

p = [2:8];

for i = 1:length(A)/10; % change i. i is same as amount of files you ...
    wanna read

    neutral(:,i) = str2num(cell2mat(A(2:end,p(1))));
    maxneutral(i) = max(neutral(:,i));
    meanneutral(i) = mean(neutral(:,i));

    happy(:,i) = str2num(cell2mat(A(2:end,p(2))));
    maxhappy(i) = max(happy(:,i));
    meanhappy(i) = mean(happy(:,i));

    sad(:,i) = str2num(cell2mat(A(2:end,p(3))));
    maxsad(i) = max(sad(:,i));
    meansad(i) = mean(sad(:,i));

    angry(:,i) = str2num(cell2mat(A(2:end,p(4))));
    maxangry(i) = max(angry(:,i));
    meanangry(i) = mean(angry(:,i));

    surprised(:,i) = str2num(cell2mat(A(2:end,p(5))));
    maxsurprised(i) = max(surprised(:,i));
    meansurprised(i) = mean(surprised(:,i));

    scared(:,i) = str2num(cell2mat(A(2:end,p(6))));
    maxscared(i) = max(scared(:,i));
    meanscared(i) = mean(scared(:,i));

    disgusted(:,i) = str2num(cell2mat(A(2:end,p(7))));
    maxdisgusted(i) = max(disgusted(:,i));
    meandisgusted(i) = mean(disgusted(:,i));

    p = p+10;
end

meanmeanneutral = mean(meanneutral);
meanmeanhappy = mean(meanhappy);
meanmeansad = mean(meansad);
meanmeanangry = mean(meanangry);
meanmeansurprised = mean(meansurprised);
meanmeanscared = mean(meanscared);
meanmeandisgusted = mean(meandisgusted);

errorneutral = std(meanneutral);
errorhappy = std(meanhappy);
errorsad = std(meansad);
errorangry = std(meanangry);
errorsurprised = std(meansurprised);
errorscared = std(meanscared);
errordisgusted = std(meandisgusted);


y1 = [meanmeanneutral meanmeanhappy meanmeansad meanmeanangry ...
```

```
    meanmeansurprised meanmeanscared meanmeandisgusted];
erry1 = [errorneutral errorhappy errorsad errorangry errorsurprised ...
    errorscared errordisgusted];
str = {'Neutral'; 'Happy'; 'Sad';'Angry'; 'Surprised'; 'Scared'; ...
    'Disgusted'};
figure
barwitherr(erry1,y1)
title('Expected emotion: Happy') %change name is needed
set(gca, 'XTickLabel',str, 'XTick',1:numel(str))
ylabel('Intensity')

y2 = [meanmeanhappy meanmeansad meanmeanangry meanmeansurprised ...
    meanmeanscared meanmeandisgusted];
erry2 = [errorhappy errorsad errorangry errorsurprised errorscared ...
    errordisgusted];
str = {'Happy'; 'Sad';'Angry'; 'Surprised'; 'Scared'; 'Disgusted'};
figure
barwitherr(erry2,y2)
title('Expected emotion: Happy')
set(gca, 'XTickLabel',str, 'XTick',1:numel(str))
ylabel('Intensity')
```

Code to make a bar graph with error:

```
%**************************************************************************
%
%   This is a simple extension of the bar plot to include error bars.  It
%   is called in exactly the same way as bar but with an extra input
%   parameter "errors" passed first.
%
%   Parameters:
%   errors - the errors to be plotted (extra dimension used if assymetric)
%   varargin - parameters as passed to conventional bar plot
%   See bar and errorbar documentation for more details.
%
%   Output:
%   [hBar hErrorbar] = barwitherr(..) returns a vector of handles to the
%                      barseries (hBar) and error bar (hErrorbar) objects
%
%   Symmetric Example:
%   y = randn(3,4);         % random y values (3 groups of 4 parameters)
%   errY = 0.1.*y;          % 10% error
%   h = barwitherr(errY, y);% Plot with errorbars
%
%   set(gca,'XTickLabel',{'Group A','Group B','Group C'})
%   legend('Parameter 1','Parameter 2','Parameter 3','Parameter 4')
%   ylabel('Y Value')
%   set(h(1),'FaceColor','k');
%
%
%   Asymmetric Example:
%   y = randn(3,4);         % random y values (3 groups of 4 parameters)
%   errY = zeros(3,4,2);
%   errY(:,:,1) = 0.1.*y;   % 10% lower error
%   errY(:,:,2) = 0.2.*y;   % 20% upper error
%   barwitherr(errY, y);    % Plot with errorbars
%
%   set(gca,'XTickLabel',{'Group A','Group B','Group C'})
%   legend('Parameter 1','Parameter 2','Parameter 3','Parameter 4')
%   ylabel('Y Value')
```

```
%
%
%   Notes:
%   Ideally used for group plots with non-overlapping bars because it
%   will always plot in bar centre (so can look odd for over-lapping bars)
%   and for stacked plots the errorbars will be at the original y value is
%   not the stacked value so again odd appearance as is.
%
%   The data may not be in ascending order.  Only an issue if x-values are
%   passed to the fn in which case their order must be determined to
%   correctly position the errorbars.
%
%
%   24/02/2011  Martina F. Callaghan    Created
%   12/08/2011  Martina F. Callaghan    Updated for random x-values
%   24/10/2011  Martina F. Callaghan    Updated for asymmetric errors
%   15/11/2011  Martina F. Callaghan    Fixed bug for assymetric errors &
%                                       vector plots
%   14/06/2013  Martina F. Callaghan    Returning handle as recommended by
%                                       Eric (see submission comments)
%   08/07/2013  Martina F. Callaghan    Only return handle if requested.
%   18/07/2013  Martina F. Callaghan    Bug fix for single group data that
%                                       allows assymetric errors.
%                                       Also removed dot from display as
%                                       per Charles Colin comment. The
%                                       handle can be returned to control
%                                       appearance.
%   27/08/2013  Martina F. Callaghan    Ensuring errors are always stored
%                                       as lowerErrors and upperErrors even
%                                       if symmetric.
%   29/10/2014  Martina F. Callaghan    Updated for 2014b graphics
%
%****************************************************************************

function varargout = barwitherr(errors,varargin)

% Check how the function has been called based on requirements for "bar"
if nargin < 3
    % This is the same as calling bar(y)
    values = varargin{1};
    xOrder = 1:size(values,1);
else
    % This means extra parameters have been specified
    if isscalar(varargin{2}) || ischar(varargin{2})
        % It is a width / property so the y values are still varargin{1}
        values = varargin{1};
        xOrder = 1:size(values,1);
    else
        % x-values have been specified so the y values are varargin{2}
        % If x-values have been specified, they could be in a random order,
        % get their indices in ascending order for use with the bar
        % locations which will be in ascending order:
        values = varargin{2};
        [tmp xOrder] = sort(varargin{1});
    end
end

% If an extra dimension is supplied for the errors then they are
% assymetric split out into upper and lower:
if ndims(errors) == ndims(values)+1
    lowerErrors = errors(:,:,1);
    upperErrors = errors(:,:,2);
```

```matlab
elseif isvector(values)~=isvector(errors)
    lowerErrors = errors(:,1);
    upperErrors = errors(:,2);
else
    lowerErrors = errors;
    upperErrors = errors;
end


% Check that the size of "errors" corresponsds to the size of the y-values.
% Arbitrarily using lower errors as indicative.
if any(size(values) ~= size(lowerErrors))
    error('The values and errors have to be the same length')
end

[nRows nCols] = size(values);
handles.bar = bar(varargin{:}); % standard implementation of bar fn
hold on
hBar = handles.bar;

if nRows > 1
    hErrorbar = zeros(1,nCols);
    for col = 1:nCols
        % Extract the x location data needed for the errorbar plots:
        if verLessThan('matlab', '8.4')
            % Original graphics:
            x = get(get(handles.bar(col),'children'),'xdata');
        else
            % New graphics:
            x =  handles.bar(col).XData + [handles.bar(col).XOffset];
        end
        % Use the mean x values to call the standard errorbar fn; the
        % errorbars will now be centred on each bar; these are in ascending
        % order so use xOrder to ensure y values and errors are too:
        hErrorbar(col) = errorbar(mean(x,1), values(xOrder,col), ...
            lowerErrors(xOrder,col), upperErrors(xOrder, col), '.k');
        set(hErrorbar(col), 'marker', 'none')
    end
else
    if verLessThan('matlab', '8.4')
        % Original graphics:
        x = get(get(handles.bar,'children'),'xdata');
    else
       % New graphics:
       x =  handles.bar.XData + [handles.bar.XOffset];
   end

    hErrorbar = errorbar(mean(x,1), values, lowerErrors, upperErrors, ...
        '.k');
    set(hErrorbar, 'marker', 'none')
end

hold off

switch nargout
    case 1
        varargout{1} = hBar;
    case 2
        varargout{1} = hBar;
        varargout{2} = hErrorbar;
end
```