Read file and determines for each sample the valence and assigns the corresponding 'traffic light' colours:

```
clear all

A = [importdata('test3.xlsx')] ;
time = 12;

p = 1;

for i = 2:length(A)
    neg = [str2num(cell2mat(A(i,4))) str2num(cell2mat(A(i,5))) ...
        str2num(cell2mat(A(i,7))) str2num(cell2mat(A(i,8)))];
    maxneg = max(neg);
    happy = str2num(cell2mat(A(i,3)));
    val(p) = happy - maxneg;
    q(p) = i;

    if val(p) > 0.3
        circles(1,5,1,'facecolor','white')
        hold on
        circles(1,3,1,'facecolor','white')
        hold on
        circles(1,1,1,'facecolor','green')
        title('The patient is feeling well')
        hold on
        pause(time/length(A));

        %disp('The patient is feeling well')
    elseif val(p) < -0.3
        circles(1,5,1,'facecolor','red')
        hold on
        circles(1,3,1,'facecolor','white')
        hold on
        circles(1,1,1,'facecolor','white')
        title('Warning: Patient is unwell')
        hold on
        pause(time/length(A));
        %disp('Warning: The patient is feeling uncomfortable')
    else
        circles(1,5,1,'facecolor','white')
        hold on
        circles(1,3,1,'facecolor','yellow')
        hold on
        circles(1,1,1,'facecolor','white')
        title('')
        hold on
        pause(time/length(A));

    end
    p = p+1;
end

figure
plot(q,val)
title('Valence')
ylabel('Intensity')
xlabel('Sample')
grid
```

Code to make coloured circles:

```
function [ h ] = circles(x,y,r,varargin)
% h = circles(x,y,r,varargin) plots circles of radius r at points x and y.
% x, y, and r can be scalars or N-D arrays.
%
% Chad Greene, March 2014. Updated August 2014.
% University of Texas Institute for Geophysics.
%
%% Syntax
%  circles(x,y,r)
%  circles(...,'points',numberOfPoints)
%  circles(...,'rotation',degreesRotation)
%  circles(...,'ColorProperty',ColorValue)
%  circles(...,'LineProperty',LineValue)
%  h = circles(...)
%
%% Description
%
% circles(x,y,r) plots circle(s) of radius or radii r centered at ...
    points given by
% x and y.  Inputs x, y, and r may be any combination of scalar,
% vector, or 2D matrix, but dimensions of all nonscalar inputs must agree.
%
% circles(...,'points',numberOfPoints) allows specification of how many ...
    points to use
% for the outline of each circle. Default value is 1000, but this may be
% increased to increase plotting resolution.  Or you may specify a small
% number (e.g. 4 to plot a square, 5 to plot a pentagon, etc.).
%
% circles(...,'rotation',degreesRotation) rotates the shape by a given
% degreesRotation, which can be a scalar or a matrix. This is useless for
% circles, but may be desired for polygons with a discernible number of ...
    corner points.
%
% circles(...,'ColorProperty',ColorValue) allows declaration of
% 'facecolor' or 'facealpha'
% as name-value pairs. Try declaring any fill property as name-value pairs.
%
% circles(...,'LineProperty',LineValue) allows declaration of 'edgecolor',
% 'linewidth', etc.
%
% h = circles(...) returns the handle(s) h of the plotted object(s).
%
%
%% EXAMPLES:
%
% Example 1:
% circles(5,10,3)
%
% % Example 2:
% x = 2:7;
% y = [5,15,12,25,3,18];
% r = [3 4 5 5 7 3];
% figure
% circles(x,y,r)
%
% % Example 3:
% figure
% circles(1:10,5,2)
%
```

```
% % Example 4:
% figure
% circles(5,15,1:5,'facecolor','none')
%
% % Example 5:
% figure
% circles(5,10,3,'facecolor','green')
%
% % Example 6:
% figure
% h = circles(5,10,3,'edgecolor',[.5 .2 .9])
%
% % Example 7:
% lat = repmat((10:-1:1)',1,10);
% lon = repmat(1:10,10,1);
% r = .4;
% figure
% h1 = circles(lon,lat,r,'linewidth',4,'edgecolor','m','facecolor',[.6 ...
    .4 .8]);
% hold on;
% h2 = ...
    circles(1:.5:10,((1:.5:10).^2)/10,.12,'edgecolor','k','facecolor','none');
% axis equal
%
% % Example 8: Circles have corners
% This script approximates circles with 1000 points. If all those points
% are too complex for your Pentium-II, you can reduce the number of points
% used to make each circle.  If 1000 points is not high enough resolution,
% you can increase the number of points.  Or if you'd like to draw
% triangles or squares, or pentagons, you can significantly reduce the
% number of points. Let's try drawing a stop sign:
%
% figure
% h = circles(1,1,10,'points',8,'color','red');
% axis equal
% % and we see that our stop sign needs to be rotated a little bit, so ...
    we'll
% % delete the one we drew and try again:
% delete(h)
% h = circles(1,1,10,'points',8,'color','red','rot',45/2);
% text(1,1,'STOP','fontname','helvetica CY',...
%     'horizontalalignment','center','fontsize',140,...
%     'color','w','fontweight','bold')
%
% figure
% circles([1 3 5],2,1,'points',4,'rot',[0 45 35])
%
%
% TIPS:
% 1. Include the name-value pair 'facecolor','none' to draw outlines
% (non-filled) circles.
%
% 2. Follow the circles command with axis equal to fix distorted circles.
%
% See also: fill, patch, and scatter.

%% Check inputs:

assert(isnumeric(x),'Input x must be numeric.')
assert(isnumeric(y),'Input y must be numeric.')
assert(isnumeric(r),'Input r must be numeric.')

```

```
if ~isscalar(x) && ~isscalar(y)
    assert(numel(x)==numel(y),'If neither x nor y is a scalar, their ...
        dimensions must match.')
end
if ~isscalar(x) && ~isscalar(r)
    assert(numel(x)==numel(r),'If neither x nor r is a scalar, their ...
        dimensions must match.')
end
if ~isscalar(r) && ~isscalar(y)
    assert(numel(r)==numel(y),'If neither y nor r is a scalar, their ...
        dimensions must match.')
end

%% Parse inputs:

% Define number of points per circle:
tmp = ...
    strcmpi(varargin,'points')|strcmpi(varargin,'NOP')|strcmpi(varargin,'corners')|...
    strncmpi(varargin,'vert',4);
if any(tmp)
    NOP = varargin{find(tmp)+1};
    tmp(find(tmp)+1)=1;
    varargin = varargin(~tmp);
else
    NOP = 1000; % 1000 points on periphery by default
end

% Define rotation
tmp = strncmpi(varargin,'rot',3);
if any(tmp)
    rotation = varargin{find(tmp)+1};
    assert(isnumeric(rotation)==1,'Rotation must be numeric.')
    rotation = rotation*pi/180; % converts to radians
    tmp(find(tmp)+1)=1;
    varargin = varargin(~tmp);
else
    rotation = 0; % no rotation by default.
end

% Be forgiving if the user enters "color" instead of "facecolor"
tmp = strcmpi(varargin,'color');
if any(tmp)
    varargin{tmp} = 'facecolor';
end

%% Begin operations:

% Make inputs column vectors:
x = x(:);
y = y(:);
r = r(:);
rotation = rotation(:);

% Determine how many circles to plot:
numcircles = max([length(x) length(y) length(r) length(rotation)]);

% Create redundant arrays to make the plotting loop easy:
if length(x)<numcircles
    x(1:numcircles) = x;
end

if length(y)<numcircles
```

```
    y(1:numcircles) = y;
end

if length(r)<numcircles
    r(1:numcircles) = r;
end

if length(rotation)<numcircles
    rotation(1:numcircles) = rotation;
end

% Define an independent variable for drawing circle(s):
t = 2*pi/NOP*(1:NOP);

% Query original hold state:
holdState = ishold;
hold on;

% Preallocate object handle:
h = NaN(size(x));

% Plot circles singly:
for n = 1:numcircles
    h(n) = fill(x(n)+r(n).*cos(t+rotation(n)), ...
        y(n)+r(n).*sin(t+rotation(n)),'',varargin{:});
end

% Return to original hold state:
if ~holdState
    hold off
end

% Delete object handles if not requested by user:
if nargout==0
    clear h
end

end
```