Outlier Detection and Removal

```
function [X, outliers_idx] = outliers(X0, num_outliers)
% format: [X, outliers_idx] = outliers(X0, num_outliers)
% Summary: outliers are detected using Thompson Tau method and turned
 to
% NaN in the output.
%
% Function Decription:
% This function accepts a vector or matrix and detects the outlier
 values
% in the vector/matrix using Thopson's Tau method, which is based on
 the
% absolute deviation of each record from the mean of the entire
% vector/matrix, and fills the outliers with NaNs in the returned
 output.
% The magnitude of Thompson's Tau value corresponding to the number of
% records in the input vector (m) or matrix (m*n) to the Standard
 Deviation
% (std) of the input vector/matrix is the rule to decide if any record
 is
% in the outliers. This means that the outliers in a curvilinear
 series or
% a curve fit may not be detected by this function as long as all the
% values are within the acceptable range from the mean (i.e.,
 tau*std).
% The mean, standard deviation and the magnitude of Thompson's Tau
% (tau*std) are calculated again after removal of each outlier. If the
% input is matrix, it will be converted to a vector before detecting
 the
% outliers, however, the output will be a matrix with the same m*n
% dimensions as input. Indexes of the outliers also will be returned,
 where
% if the input was a vector, the index vector also will be a vector,
% however, if the input was a matrix, outlier indexes will be returned
 in a
% two-column matrix showing i,j indexes of the outliers (see examples
% below). Maximum number of outliers to be removed is given by the
 second
% input (optional) argument (i.e., num_outliers). This is to avoid
% shrinking data when there is many points that could be treated as
% outliers by the Thomson's Tau function. If the num_outliers is not
% provided, only one outlier will be removed by default, even though
 there
% might be more than one obvious outliers in the input data. However,
 a
% large number for num_outliers input argument won't make any impact
 on the
% result in case is no (or less than num_outleirs) outliers in the
 input
% data.
%
```

```
% Please use my other function called regoutliers (also uploaded in
% Matlab File Exchange) if you are dealing with the outliers in fitted
% data.
%
% --Inputs:
%   X0: input vector or matrix which contains outliers
%   num_outliers: number of outliers that should be removed from the
 input
%   vector/matrix (default is 1)
%
% --Outputs:
%   X: output vector/matrix with outliers (if any detected) turned to
 NaN
%   outliers_idx: the index(es) of any detected outliers, the more
 extreme
%   outliers will be detected first, so the first index refers to the
%   most extreme outlier and so forth
%
% --Theory of Thompson Tau method:
%   http://www.mne.psu.edu/me345/Lectures/Outliers.pdf
%   http://www.jstor.org/stable/2345543 (Thompson, 1985)
%
% --Note: this function is an improvement based on Vince Petaccio,
 2009:
% http://www.mathworks.com/matlabcentral/fileexchange/24885-remove-
outliers
% --Improvements:
%  1. Handling NaNs in inputs
%  2. Number of outliers to be removed is restricted to a user defined
%  maximum to avoid uncontrolled shrinking of input dataset
%  3. Filling outliers by NaNs to preserve original dimensions of the
 input
%  vector/matrix; this is crucial when the input variable is supposed
 to be
%  used with another variable with the same size (e.g., for plotting,
%  regression calculations, etc.)
%  4. Indexes of the outliers that have been detected and removed are
%  returned so that the user knows which records have been removed,
 and
%  since the indexes are ordered from the most extreme (negative or
%  positive) to less extreme outliers, user will know which point was
 in
%  the farthest outliers.
%  5. Syntax and algorithm has been significantly improved, this
 includes
%  the logic for detection of the outliers from the upper and lower
 limits.
%  Logic to detect an outlier is solely based on the absolute distance
 of
%  each record from the central point rather than detecting the
 outliers
%  sequentially, which was the case in Vince Petaccio, 2009, where
 outliers
```

```
%  were detected and removed by order of one from the upper and the
 next
%  from the lower extremes. This code first arranges the extreme
 values
%  (upper or lower) to one side of the sorted vector based on the
 absolute
%  distance from the center (while preserving the original arrangement
 in
%  the input vector) then removes the bottom line element if it meets
%  outlier conditions. This process continues until num_outliers is
%  reached.
%  6. This function is enhanced to handle both vectors and matrices.
%  7. Valuable feedback from the user community (especially a user
 under
%  the name of John D'Errico) helped to detect and fix some issues in
 the
%  algorithm, which were related to exceptions involved in detecting
%  special types of outliers (please refer to the comments section).
 These
%  issues are now fixed. However, this code won't be able to find
 outliers
%  in curvilinear fitted data (which was one of the issues raised).
 This is
%  because the underlying logic to detect the outliers in (modified)
%  Thompson's Tau method is deviation from the mean. Check the
 references
%  given above or a good statistical reference if you are not very
%  familiar with the concept of outliers removal. One thing you should
%  know is that no outliers is absolutely an outlier, it is always a
%  relative, and you might consider a point an outlier depending what
%  criteria you have defined for your data analysis. You may have some
 luck
%  dealing with fitted outliers using my other code called
 'regoutliers'
%  also submitted to Matlab File Exchange.
%
%  And finally, this code is open source, feel free to make
 improvements
%  and post it to the Matlab File Exchange. If you do so, please don't
%  forget to reference my code (Mathworks asks which previous
 submission
%  inspired your code).
%
% --Examples:
% -Example 1. Vector input:
%  X0=[2.0, 3.0, -50.5, 4.0, 109.0, 6.0]
%  [X, outliers_idx] = outliers(X0, 2) %call function with vector
 input
%
%  X =
%      2     3   NaN     4   NaN     6
%
%  outliers_idx =
%      5     3
```

```
%
% -Example 2. Matrix input:
%  X0= [2.0,   3.0,   -50.5,    4.0,  109.0,   6.0;
%      5.3,   7.0,    80.0,    2.0,   NaN,    1.0;
%      5.1,   2.7,     3.8,    2.0,   3.5,    21.0]
%  [X, outliers_idx] = outliers(X0, 4) %call function with matrix
 input
%
%  X =
%      2       3       NaN     4     NaN     6
%      5.3     7       NaN     2     NaN     1
%      5.1     2.7     3.8     2     3.5     NaN
%
%  outliers_idx =
%    %(i)   (J)    %annotated
%      1     5
%      2     3
%      1     3
%      3     6
%
% First version: 06 June 2012
% Enhanced to handle matrix: 09 June 2012
% Issues fixed: 10 Sep. 2014
% email: sohrabinia.m@gmail.com
%---------------------------------------------------------------------------

% Initializations:
outliers_idx=[]; %if no outliers has been found, return empty matrix
 to
% avoid problems in indexing based on outliers_idx
X=[];
if nargin<1
    disp('Error! at least one input argument is necessary');
    return;
elseif nargin<2
    num_outliers=1;
end

% if the input is a line vector, transpose it to column vector:
vec=0;
[rows,cols]=size(X0);
if rows==1     %row vector
   X0=X0';     %transposed
   vec=1;
elseif cols==1 %column vector
   vec=2;
end
[rows,cols]=size(X0); %update [rows cols] after transpose

X0=X0(:); %convert matrix to vector
n1=length(X0); %Determine the number of samples in datain

if n1 < 3
    display(['Error! There must be at least 3 samples in the' ...
        ' dataset in order to use this function.']);
    return
end

X=X0; %keep original vector

%Sort the input data vector so that removing the extreme values
 becomes an
%arbitrary task. Store indexes too, to be able to recreate the data to
 the
%original order after removing the outliers. Also note that NaNs are
%considered maximum by sort function:
[X, idx]=sort(X);
X(isnan(X))=[]; %remove NaNs before calculations
n=length(X); %length after removal of NaNs
nns=n1-n; %NaN elements gathered at the end by sort with default mode


stDev= std(X); %calculate stDev, standard deviation of input vector
xbar = mean(X);%calculate the sample mean

% tau is a vector containing values for Thompson's Tau:
tau =     [1.150; 1.393; 1.572; 1.656; 1.711; 1.749; 1.777; 1.798;
 1.815;
    1.829; 1.840; 1.849; 1.858; 1.865; 1.871; 1.876; 1.881; 1.885;
 1.889;
    1.893; 1.896; 1.899; 1.902; 1.904; 1.906; 1.908; 1.910; 1.911;
 1.913;
    1.914; 1.916; 1.917; 1.919; 1.920; 1.921; 1.922; 1.923; 1.924];

% Determine the value of stDev times Tau
if n > length(tau)
    tauS=1.960*stDev; %For n > 40
else
    tauS=tau(n)*stDev; %For samples of size 3 < n < 40
end

%tauS,stDev,xbar
%tauS=tauS*(n-1)/(sqrt(n)*sqrt(n-2+tauS*tauS))

% Compare the values of extreme high/low data points with tauS:
i=1;
olfirst=0;
ollast=length(idx)+1;
while num_outliers > 0
 if abs(X(1)-xbar)> abs(X(end)-xbar)
   ol=abs(X(1)-xbar);
   beg0=2;
   end0=length(X);
   olfirst=olfirst+1;
 else
   ol=abs(X(end)-xbar);
   beg0=1;
   end0=length(X)-1;
```

```
    ollast=ollast-1;
  end
  if ol > tauS
     X=X(beg0:end0);
     n=length(X);
     if beg0==2
         outliers_idx(i)=idx(olfirst);
     else
         outliers_idx(i)=idx(ollast-nns);
     end
     X0(outliers_idx(i))=NaN;
     i=i+1;

     % Determine the NEW value of S times tau
     stDev=std(X);
     xbar=mean(X);
     if n > length(tau)
         tauS=1.960*stDev; %For n > 40
     else
         tauS=tau(n)*stDev; %For samples of size 3 < n < 40
     end
  end
  beg0=1;
  end0=length(X);
  num_outliers=num_outliers-1; %reduce requested num_outliers by 1
end %end of while

% transform vector to matrix and return X as final output with the
 same
% dimensions as input but outliers turned to NaNs; also, convert
 outlier
% indexes from vector to matrix form, leave vector form if the input
 was a
% vector:
if vec==1
    X=X0'; %row vector after removal of outliers
elseif vec==2
    X=X0;  %column vector after removal of outliers
    outliers_idx=outliers_idx'; %indexes of column vector outliers
else
    matidx=zeros(length(X0),2);
    X=nan(rows,cols);
    i2=1;
    for j=1:cols
        for i=1:rows
            X(i,j)=X0(i+(j-1)*rows); %matrix after removal of outliers
            matidx(i2,1)=i;
            matidx(i2,2)=j;
            i2=i2+1;
        end
    end
    if isempty(outliers_idx)==0
        outliers_idx=matidx(outliers_idx,:); %indexes of matrix
 outliers
```

```
    end
end %end of transformations

end %end of outliers function
```

Published with MATLAB® R2016a