```
% function simple_linear_LS(X,y,t_value)
%
% This MATLAB function performs the explicit calculations
% required to perform a least squares analysis of data for
% fitting a linear model.  The user inputs the design matrix
% X (which is assumed to have a y-intercept), a column vector
% of measured values y, and the appropriate t values used for the
% confidence interval and the number of degrees of freedom.
%
% K. Beers.
% MIT ChE
% 12/3/2001

function [b,RSS,yhat,yhat_CI,b_CI,sample_var,iflag] = ...
   simple_linear_LS(X,y,t_value);

iflag = 0;


% First, we check the size of the design matrix and response
% vector.
n = size(X,1);
m = size(X,2)-1;
if(n ~= length(y))
   iflag = -1;
   error('dimension of y and X do not agree');
end

% Now, calculate the least squares estimate of b.  A better way to do
% this calculation is to use singular value decomposition, but here we
% use a simple Gaussian elimination to keep things simple.

% First, calculate the matrix X'*X.
XT_X = X'*X;

% Next, esimate the condition number.  A large value signifies the
% need to use singular value decomposition.
kappa = condest(XT_X);
disp(['Condition number (est.) of XT_X = ', num2str(kappa)]);

% Now, calculate b.
b = XT_X\(X'*y);
disp(' '); disp('LS fitted parameters : '); b,

% Calculate the inverse of XT_X for use in developing error bounds.
XT_X_inv = inv(XT_X);


% Calculate model predictions and residual errors.
```

```
yhat = X*b;
residual = y - yhat;
RSS = dot(residual,residual);
disp(' '); disp('RSS = '); RSS,

% estimate the sample variance and sample standard deviation.
disp(' ');
sample_var = RSS/(n-m-1),
sample_std = sqrt(sample_var),

% make a plot of the residuals
figure;
plot(y,residual,'o');
hold on;
x_plot = [min(y); max(y)];
y_plot = [0; 0];
plot(x_plot,y_plot,'-.');
xlabel('y'); ylabel('residual');
title(['Residual errors, s = ',num2str(sample_std)]);


% Now, calculate the confidence interval data.
yhat_CI = zeros(n,2);
for j=1:n
   xj = X(j,:)';
   half_width = t_value*sample_std * ...
      sqrt(xj'*XT_X_inv*xj);
   yhat_CI(j,1) = yhat(j) - half_width;
   yhat_CI(j,2) = yhat(j) + half_width;
end


% Then, calculate confidence intervals on the model
% parameters.
b_CI = zeros(m+1,2);
for j=0:m
   k = j+1;
   uk = zeros(m+1,1);
   uk(k) = 1;
   half_width = t_value*sample_std * ...
      sqrt(uk'*XT_X_inv*uk);
   b_CI(k,1) = b(k) - half_width;
   b_CI(k,2) = b(k) + half_width;
end


% Now make a plot of y vs. each predictor variable.
for ipred = 1:m
   k = ipred+1;
   xk = X(:,k);
   figure;
```

```
    plot(xk,y,'o');
    hold on;
    plot(xk,yhat,'-');
    plot(xk,yhat_CI(:,1),'--');
    plot(xk,yhat_CI(:,2),'--');
    title('Comparison between model fit and data');
    xlabel(['Predictor variable # ', int2str(ipred)]);
    ylabel('y (o) vs. yhat (line) w/ CI');
end

iflag = 1;

return;
```