% simple_LS.m

```
% function [b,bint,r,rint,stats,sample_var,iflag] = ...
%             simple_LS(y,X,alpha,...
%              iplot,plot_type,plot_var,plot_text);
%
% This MATLAB function employs the statistics toolkit
% functions to perform a simple linear least squares
% data fit using multiple regression.  One enters the
% design matrix X, vector of values y, and the alpha
% value for developing the confidence intervals.
%
%
% INPUT :
% =======
% y - the column vector of response variables
% X - the design matrix
% alpha - for 95% confidence interval, use alpha = 0.05
% iplot - if zero, don't make any plots; if 1, make
%      plots of residuals and response vs. predictor
%      variables; if 2, add plot of response vs. plot_var
% plot_type - if 0, use log plots; if 1, semilogx, if 2,
%      semilogy; if 3 loglog plots.  This is only for the
%      final plot using plot_var.
% plot_var - this is a column vector, of the same dimension
%      as y, for use in supplemental plot if iplot=2
% plot_text - this is a structure with three fields.
%   .xlabel = string to be used for labeling x-axis
%   .ylabel = string to be used for labeling y-axis
%   .title = string used to set title of supplemental plot
%
% OUTPUT :
% ========
% b - the least squares fitted parameters
% bint - the confidence interval bounds on each parameter
% r - the vector of residual errors
% rint - the confidence interval bounds on the residual errors
% stats - contains parameters measuring the quality of fit
%      (see help section on MATLAB function regress for
%      further data)
% sample_var - the sample variance of the response data
% iflag - if 1, function exited with sucessful performance
%
% K. Beers
% MIT ChE
% 12/5/2001

function [b,bint,r,rint,stats,sample_var,iflag] = ...
    simple_LS(y,X,alpha,iplot,plot_type,plot_var,plot_text);
```

```
iflag = 0;


% We extract the number of obversations and the number of
% predictor variables (assuming y intercept present).
n = length(y);
ntest = size(X,1);
if(ntest ~= n)
   iflag = -1;
   error('simple_LS: dimensions of X and y do not match');
end
m = size(X,2) - 1;
if(m > n)
   iflag = -2;
   error('simple_LS: insufficient data to perform regression');
end


% The toolkit function regress is called to perform
% the multiple regression.
[b,bint,r,rint,stats] = regress(y,X,alpha);


% We now calculate the residual sum of squarred errors and
% the sample variance and standard deviation.
RSS = dot(r,r);
sample_var = RSS/(n-m-1);
sample_std = sqrt(sample_var);


% We now plot the residuals vs. the reponses and check
% for normality of the errors.
if(iplot)
   figure;
   plot(y,r,'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, RSS = ',num2str(RSS), ...
      ', s = ', num2str(sample_std)]);

   % We make a norm plot of the residuals.  For a normal
   % distribution, this should be a straight line.
   figure;
   normplot(r);

end
```

```
% Using the fitted parameter vector, we calculate
% the model predictions.
yhat = X*b;

% We now make a plot of y vs. each predictor variable.
if(iplot)
   for ipred=1:m
      k = ipred+1;
      xk = X(:,k);
      figure;
      plot(xk,y,'o');
      hold on;
      errorbar(xk,yhat,rint(:,1),rint(:,2));
      xlabel(['Predictor variable # ', int2str(ipred)]);
      ylabel('y (o) vs. yhat (line) w/ CI');
      title('Comparison between model fit and data');
   end
end


% if requested, we make a final plot of the model results.
if(iplot==2)
   if(plot_type == 0)
      func_plot = 'plot';
   elseif(plot_type == 1)
      func_plot = 'semilogx';
   elseif(plot_type == 2)
      func_plot = 'semilogy';
   else
      func_plot = 'loglog';
   end
   figure;
   feval(func_plot,plot_var,y,'o');
   hold on;
   feval(func_plot,plot_var,yhat);
   feval(func_plot,plot_var,yhat+rint(:,1),'-.');
   feval(func_plot,plot_var,yhat+rint(:,2),'-.');
   xlabel(plot_text.xlabel);
   ylabel(plot_text.ylabel);
   title(plot_text.title);
end

iflag = 1;

return;
```