Table of Contents


```
function [Km,Vmax] = project_function(time, enzymeData);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
%   This program estimates the Michaelis-Menten parameters, Km and
 Vmax, for
% a given enzyme's data.
%
% Function Call
%   [Km, Vmax] = project_function(time, substate_data);
%
% Input Arguments
%   time: the time variable for each given data set.
%   enzymeData: First row is the inital concentrations of the
 substrates for
% each given test. The rest of the rows are the data points for each
 test; each
% test in a serparate column.
%
% Output Arguments
%   Km: Outputs the estimated Km value for the enzyme
%   Vmax: Outputs the estimated Vmax value for the enzyme
%
% Assignment Information
%   Assignment:     M02, Problem 1
%   Team member:    Nic Ballesteros, nballes@purdue.edu
%   Team member:    Annabelle Johnson, john245@purdue.edu
%   Team member:    Alan Camacho, @purdue.edu
%   Team ID:        005-19
%   Academic Integrity:
%     [x] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

```
% Each test is stored in the following manner
% to get the data for a test the following command is used :
%     test(test#).data;
% to get the duplicate data for a test the following command is used:
%     test(test#).dupData;
% to get the concentation of the test use:
%     test(test#).concentation;

%Gets rid of times where reactions are no longer occuring
for i = 1:10
  test(i).data = rmmissing(enzymeData(2:end, i)); %get all not NaN
 values in each col for each test
  test(i).dataSize = size(test(i).data); %Determines the number
 of seconds that pass before the reaction stops for each initial
 substrate concentration
  test(i).time = time(1:test(i).dataSize(1)); %Creates a matrix of
 times where the reaction was occuring for each initial substrate
 concentration
  %store the duplicate data
  test(i).dupData = rmmissing(enzymeData(2:end, i + 10)); %get all not
 NaN values in each col for each duplicate test
  test(i).dupDataSize = size(test(i).dupData); %Determines the number
 of seconds that pass before the duplicate reaction stops for each
 initial substrate concentration
  test(i).dupTime = time(1:test(i).dupDataSize(1)); %Creates a matrix
 of times where the duplicate reaction was occuring for each initial
 substrate concentration
  %store the concentation
  test(i).concentation = enzymeData(1, i); %Creates a matrix of
 substrate concentrations
end;

mmData = zeros(20, 2); %Michaelis-Menten data that will eventually be
 plotted
```

```
Not enough input arguments.

Error in project_function (line 45)
  test(i).data = rmmissing(enzymeData(2:end, i)); %get all not NaN
 values in each col for each test
```

CALCULATIONS

```
% ----------------
% find the v0 data
```

```
% ----------------

for i = 1:10
  test(i).v0 = [0 0];
  x = test(i).time;
  y = test(i).data;

  x(1) = []; %had a divide by zero error
  y(1) = []; %to line up both vectors

  x = x(1:500); %linearize only the first 500 values
  y = y(1:500); %linearize only the first 500 values

  y = x ./ y; %linearize the product data

  %find line best fit
  xline = mean(x);
  yline = mean(y);
  xyline = mean(x .* y);

  a = (xline * yline - xyline) / (xline ^ 2 - mean(x .^ 2));
  b = yline - a * xline;
  %done with best fit

  a = 1 / a;
  b = b * a;

  %make dataset off of modeled line
  xDataPoints = 1:500;
  yDataPoints = (a * xDataPoints) ./ (b + xDataPoints);

  %use line to find initial velocity
  test(i).v0(1) = (yDataPoints(2) - yDataPoints(1)) / (xDataPoints(2)
 - xDataPoints(1));

  %add it to the Michaelis-Menten dataset
  mmData(2 * i - 1, 1) = test(i).concentation;
  mmData(2 * i - 1, 2) = test(i).v0(1);

  %do the same thing as above but for the duplicate data
  x = test(i).dupTime;
  y = test(i).dupData;

  x(1) = []; %had a divide by zero error
  y(1) = []; %to line up both vectors

  x = x(1:500); %linearize only the first 500 values
  y = y(1:500); %linearize only the first 500 values

  y = x ./ y; %linearize the product data


  %find the line best fit
  xline = mean(x);
```

```
  yline = mean(y);
  xyline = mean(x .* y);

  a = (xline * yline - xyline) / (xline ^ 2 - mean(x .^ 2));
  b = yline - a * xline;

  a = 1 / a;
  b = b * a;

  %use the model to make a dataset
  xDataPoints = 1:500;
  yDataPoints = (a * xDataPoints) ./ (b + xDataPoints);

  %use data set to find the inital velocity
  test(i).v0(2) = (yDataPoints(2) - yDataPoints(1)) / (xDataPoints(2)
 - xDataPoints(1));

  %add the inital velocity to the Michaelis-Menten dataset
  mmData(2 * i, 1) = test(i).concentation;
  mmData(2 * i, 2) = test(i).v0(2);
end;

%--------------------------------------
%implementing Hanes-Woolf Linearization
%--------------------------------------

data(:, 1) = mmData(:,1);
data(:, 2) = mmData(:, 1) ./ mmData(:, 2);

%data = rmoutliers(data);

Y = mmData(:, 1) ./ mmData(:, 2);
X = mmData(:, 1);

Xline = mean(X);
Yline = mean(Y);
XYline = mean(X .* Y);

a = (Xline * Yline - XYline) / (Xline ^ 2 - mean(X .^ 2));
b = Yline - a * Xline;

fx = X * a + b; %Hanes-Woolf Line

Vmax = 1 / a; %calculate Vmax from Hanes-Woolf
Km = b * Vmax; %calculate Km from Hanes-Woolf

% make a dataset that follows the model
numberOfDataPoints = 100;
seperation = (2000 - 3.75) / numberOfDataPoints;
xmodel = 3.75:seperation:2000;
MichaelisModel = Vmax * xmodel ./ (Km + xmodel);
```

FORMATTED TEXT/FIGURE DISPLAYS

```
%Plots the Calculated Reaction Velocities against the Model Reaction
%Velocities
 figure;
 subplot(2,1,1);
 plot(mmData(:,1), mmData(:, 2), 'ko'); %Calculated Reaction
 Velocities
 title('Reaction Velocity as Initial [S] changes');
 xlabel('Initial Substrate Concentration [S] (uM)');
 ylabel('Reaction Velocity (uM/s)');
 hold on;
 plot(xmodel, MichaelisModel, 'r--'); %Michealis Model curve
 legend('Calculated Reaction Velocities','Michaelis
 Model','location','best');

 subplot(2,1,2);
 plot(X,Y, 'ro');
 hold on;
 plot(X, fx, 'b-');
 xlabel('Initial Substrate Concentration (uM)');
 ylabel('Velocity / [S]');
 title('Hanes-Woolf Linearization');
 legend('Linarized Velocity Data', 'Best Fit
 Line', 'location', 'best');
```

COMMAND WINDOW OUTPUT

```
fprintf("Vmax: %.3f\n", Vmax);
fprintf("Km: %.3f\n", Km);
```

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The program we are submitting is our own original work.

Published with MATLAB® R2019b