ZunZunSite3

User-Selectable Polynomial

Equation Family: Polynomial

$$z = a + bx^0y^1 + cx^0y^2 + dx^1y^0 + fx^1y^1 + gx^1y^2$$

Tue Mar 27 23:34:07 2018 local server time

Galatians 5:22-23

But the fruit of the Spirit is love, joy, peace, longsuffering, gentleness, goodness, faith, meekness, temperance: against such there is no law.

Read or search the King James Bible online at
http://quod.lib.umich.edu/k/kjv/

Coefficients

z = a + bx^0y^1 + cx^0y^2 + dx^1y^0 + fx^1y^1 + gx^1y^2

Fitting target of lowest sum of squared absolute error = 3.7485802849909211E-02

a = 1.4610559504218952E+00
b = 1.0993977915597843E-01
c = 5.5707457038157646E-05
d = -6.6547586823996904E-02
f = 5.9370452670676590E-04
g = -1.4752241260518328E-06

Coefficient and Fit Statistics

Most statstics from scipy.odr.odrpack and http://www.scipy.org/Cookbook/OLS
LL, AIC and BIC from http://stackoverflow.com/questions/7458391/python-multiple-linear-regression-using-ols-code-with-specific-data

If you entered coefficient bounds. Parameter statistics may
not be valid for parameter values at or near the bounds.

Degrees of freedom (error): 4
Degrees of freedom (regression): 5
Chi-squared: 0.0374858028499
R-squared: 0.999996182751
R-squared adjusted: 0.999991411189
Model F-statistic: 209574.201808
Model F-statistic p-value: 6.37495611855e-11
Model log-likelihood: 13.7425051747
AIC: -1.54850103494
BIC: -1.36694997915
Root Mean Squared Error (RMSE): 0.0612256505477

a = 1.4610559504218952E+00
       std err: 1.46306E-02
       t-stat: 1.20791E+01
       p-stat: 2.69414E-04
       95% confidence intervals: [1.12523E+00, 1.79689E+00]
b = 1.0993977915597843E-01
       std err: 3.37887E-06
       t-stat: 5.98094E+01
       p-stat: 4.68020E-07
       95% confidence intervals: [1.04836E-01, 1.15043E-01]
c = 5.5707457038157646E-05
       std err: 6.76277E-12
       t-stat: 2.14216E+01
       p-stat: 2.80843E-05
       95% confidence intervals: [4.84872E-05, 6.29277E-05]
d = -6.6547586823996904E-02
       std err: 1.24031E-05
       t-stat: -1.88959E+01
       p-stat: 4.61972E-05
       95% confidence intervals: [-7.63257E-02, -5.67695E-02]
f = 5.9370452670676590E-04
       std err: 8.74887E-09
       t-stat: 6.34738E+00
       p-stat: 3.15589E-03
       95% confidence intervals: [3.34009E-04, 8.53400E-04]
g = -1.4752241260518328E-06
       std err: 3.01457E-14
       t-stat: -8.49660E+00
       p-stat: 1.05219E-03
       95% confidence intervals: [-1.95728E-06, -9.93163E-07]

Coefficient Covariance Matrix
[1.56118753e+00 -1.61372032e-02   2.07942938e-05 -3.97521825e-02
   8.37606852e-04 -1.22980087e-06]
[-1.61372032e-02   3.60548783e-04 -5.06825470e-07   5.04895956e-04
  -1.44616350e-05   1.13958675e-08]
[2.07942938e-05 -5.06825470e-07   7.21635024e-10 -6.71670779e-07
   1.99352186e-08 -1.48263872e-11]
[-3.97521825e-02   5.04895956e-04 -6.71670779e-07   1.32349689e-03
  -3.17823858e-05   5.00759295e-08]
[8.37606852e-04 -1.44616350e-05   1.99352186e-08 -3.17823858e-05
   9.33565949e-07 -1.43687574e-09]
[-1.22980087e-06   1.13958675e-08 -1.48263872e-11   5.00759295e-08
  -1.43687574e-09   3.21675915e-12]

Error Listing

Indep. Data 1	Indep. Data 2	Dependent Data	Predicted	Abs Error	Rel Error
2.10E+01	0.000E+00	0.00E+00	6.3556627118E-02	6.355663E-02	n/a
1.00E+02	3.130E+01	0.00E+00	1.5757330377E-02	1.575733E-02	n/a
3.22E+01	5.000E+00	0.00E+00	-3.6285889727E-02	-3.628589E-02	n/a
3.81E+01	7.800E+00	0.00E+00	-4.0470052648E-02	-4.047005E-02	n/a
2.02E+01	2.519E+02	3.25E+01	3.2475570972E+01	-2.442903E-02	n/a
1.32E+01	3.401E+02	4.47E+01	4.4829648645E+01	1.296486E-01	n/a
1.45E+01	3.535E+02	4.68E+01	4.6691301796E+01	-1.086982E-01	n/a
1.09E+01	3.710E+02	4.94E+01	4.9378597720E+01	-2.140228E-02	n/a
1.65E+01	3.948E+02	5.25E+01	5.2523721546E+01	2.372155E-02	n/a
0.00E+00	6.693E+02	1.00E+02	9.9998601306E+01	-1.398694E-03	n/a

Error Statistics

NOTE: Relative error statistics cannot be compiled, as at least one of
the dependent variable data points contains a value of exactly zero.

Absolute Error	
Minimum:	-1.086982E-01
Maximum:	1.296486E-01
Mean:	1.666668E-11
Std. Error of Mean:	2.040855E-02
Median:	-1.140049E-02
Variance:	3.748580E-03
Standard Deviation:	6.122565E-02
Skew:	4.489939E-01
Kurtosis:	1.583984E-01

Data Statistics

X	Y	Z
Minimum:	0.000000E+00	0.000000E+00	0.000000E+00
Maximum:	1.000000E+02	6.693000E+02	1.000000E+02
Mean:	2.666000E+01	2.424700E+02	3.259000E+01
Std. Error of Mean:	8.822625E+00	7.139117E+01	1.044568E+01
Median:	1.835000E+01	2.960000E+02	3.860000E+01
Variance:	7.005484E+02	4.587029E+04	9.820109E+02
Standard Deviation:	2.646787E+01	2.141735E+02	3.133705E+01
Skew:	1.982230E+00	3.637258E-01	6.016567E-01
Kurtosis:	3.028114E+00	-8.183908E-01	-3.598172E-01

Source Code in C++

```
// To the best of my knowledge this code is correct.
// If you find any errors or problems please contact
// me directly using zunzun@zunzun.com.
//
//      James


#include

// Fitting target: lowest sum of squared absolute error
// Fitting target value = 0.0374858028499

double Polynomial_UserSelectablePolynomial_model(double x_in, double y_in)
{
    double temp;
    temp = 0.0;

    // coefficients
    double a = 1.4610559504218952E+00;
    double b = 1.0993977915597843E-01;
    double c = 5.5707457038157646E-05;
    double d = -6.6547586823996904E-02;
    double f = 5.9370452670676590E-04;
    double g = -1.4752241260518328E-06;

    temp += a;
    temp += b * y_in;
    temp += c * pow(y_in, 2.0);
    temp += d * x_in;
    temp += f * x_in * y_in;
    temp += g * x_in * pow(y_in, 2.0);
    return temp;
}
```

Source Code in Fortran90

```
! To the best of my knowledge this code is correct.
! If you find any errors or problems please contact
! me directly using zunzun@zunzun.com.
!
!      James


! fortran90 has no power function, only an operator, create
! a function for pyeq3 automated source code generation
real function pow(a, b)
real :: a ! input
real :: b ! input
real :: c ! output
c = a**b
end function pow

! Fitting target: lowest sum of squared absolute error
! Fitting target value = 0.0374858028499

real function Polynomial_UserSelectablePolynomial_model(x_in, y_in)
real :: x_in ! input
real :: y_in ! input
 real :: temp ! output

! coefficients
real :: a = 1.4610559504218952E+00
real :: b = 1.0993977915597843E-01
real :: c = 5.5707457038157646E-05
real :: d = -6.6547586823996904E-02
real :: f = 5.9370452670676590E-04
real :: g = -1.4752241260518328E-06

temp = 0.0

temp = temp + a
temp = temp + b * y_in
temp = temp + c * pow(y_in, 2.0)
temp = temp + d * x_in
temp = temp + f * x_in * y_in
temp = temp + g * x_in * pow(y_in, 2.0)

end function Polynomial_UserSelectablePolynomial_model
```

Source Code in Java

```
// To the best of my knowledge this code is correct.
// If you find any errors or problems please contact
// me directly using zunzun@zunzun.com.
//
//      James


import java.lang.Math;

// Fitting target: lowest sum of squared absolute error
// Fitting target value = 0.0374858028499

class Polynomial_UserSelectablePolynomial
{
    double Polynomial_UserSelectablePolynomial_model(double x_in, double y_in)
    {
        double temp;
        temp = 0.0;

        // coefficients
        double a = 1.4610559504218952E+00;
        double b = 1.0993977915597843E-01;
        double c = 5.5707457038157646E-05;
        double d = -6.6547586823996904E-02;
        double f = 5.9370452670676590E-04;
        double g = -1.4752241260518328E-06;

        temp += a;
        temp += b * y_in;
        temp += c * Math.pow(y_in, 2.0);
        temp += d * x_in;
        temp += f * x_in * y_in;
        temp += g * x_in * Math.pow(y_in, 2.0);
        return temp;
    }
}
```

Source Code in Julia

```
# To the best of my knowledge this code is correct.
# If you find any errors or problems please contact
# me directly using zunzun@zunzun.com.
#
#      James


# julia has no power function, only an operator, create
# a function for pyeq3 automated source code generation
pow(x,y) = x ^ y

# Fitting target: lowest sum of squared absolute error
# Fitting target value = 0.0374858028499

function Polynomial_UserSelectablePolynomial_model(x_in, y_in)
    temp = 0.0

    # coefficients
    a = 1.4610559504218952E+00
    b = 1.0993977915597843E-01
    c = 5.5707457038157646E-05
    d = -6.6547586823996904E-02
    f = 5.9370452670676590E-04
    g = -1.4752241260518328E-06

    temp += a
    temp += b * y_in
    temp += c * pow(y_in, 2.0)
    temp += d * x_in
    temp += f * x_in * y_in
    temp += g * x_in * pow(y_in, 2.0)

end
```

Source Code in JavaScript

```
// To the best of my knowledge this code is correct.
// If you find any errors or problems please contact
// me directly using zunzun@zunzun.com.
//
//      James


// Fitting target: lowest sum of squared absolute error
// Fitting target value = 0.0374858028499

function Polynomial_UserSelectablePolynomial_model(x_in, y_in)
    {
    var temp;
    temp = 0.0;

    // coefficients
    var a = 1.4610559504218952E+00;
    var b = 1.0993977915597843E-01;
    var c = 5.5707457038157646E-05;
    var d = -6.6547586823996904E-02;
    var f = 5.9370452670676590E-04;
    var g = -1.4752241260518328E-06;

    temp += a;
    temp += b * y_in;
    temp += c * Math.pow(y_in, 2.0);
    temp += d * x_in;
    temp += f * x_in * y_in;
    temp += g * x_in * Math.pow(y_in, 2.0);
    return temp;
}
```

Source Code in Python

```
# To the best of my knowledge this code is correct.
# If you find any errors or problems please contact
# me directly using zunzun@zunzun.com.
#
#      James


import math

# Fitting target: lowest sum of squared absolute error
# Fitting target value = 0.0374858028499

def Polynomial_UserSelectablePolynomial_model(x_in, y_in):
    temp = 0.0

    # coefficients
    a = 1.4610559504218952E+00
    b = 1.0993977915597843E-01
    c = 5.5707457038157646E-05
    d = -6.6547586823996904E-02
    f = 5.9370452670676590E-04
    g = -1.4752241260518328E-06

    temp += a
    temp += b * y_in
    temp += c * math.pow(y_in, 2.0)
    temp += d * x_in
    temp += f * x_in * y_in
    temp += g * x_in * math.pow(y_in, 2.0)
    return temp
```

Source Code in C#

```
// To the best of my knowledge this code is correct.
// If you find any errors or problems please contact
// me directly using zunzun@zunzun.com.
//
//      James


using System;

// Fitting target: lowest sum of squared absolute error
// Fitting target value = 0.0374858028499

class Polynomial_UserSelectablePolynomial
{
    double Polynomial_UserSelectablePolynomial_model(double x_in, double y_in)
    {
        double temp;
        temp = 0.0;

        // coefficients
        double a = 1.4610559504218952E+00;
        double b = 1.0993977915597843E-01;
        double c = 5.5707457038157646E-05;
        double d = -6.6547586823996904E-02;
        double f = 5.9370452670676590E-04;
        double g = -1.4752241260518328E-06;

        temp += a;
        temp += b * y_in;
        temp += c * Math.Pow(y_in, 2.0);
        temp += d * x_in;
        temp += f * x_in * y_in;
        temp += g * x_in * Math.Pow(y_in, 2.0);
        return temp;
    }
}
```

Source Code in SCILAB

```
// To the best of my knowledge this code is correct.
// If you find any errors or problems please contact
// me directly using zunzun@zunzun.com.
//
//      James


// Fitting target: lowest sum of squared absolute error
// Fitting target value = 0.0374858028499

function z = Polynomial_UserSelectablePolynomial_model(x_in, y_in)
    temp = 0.0;

    // coefficients
    a = 1.4610559504218952E+00;
    b = 1.0993977915597843E-01;
    c = 5.5707457038157646E-05;
    d = -6.6547586823996904E-02;
    f = 5.9370452670676590E-04;
    g = -1.4752241260518328E-06;

    temp = temp + a;
    temp = temp + b * y_in;
    temp = temp + c * power(y_in, 2.0);
    temp = temp + d * x_in;
    temp = temp + f * x_in * y_in;
    temp = temp + g * x_in * power(y_in, 2.0);

    z = temp;
endfunction
```

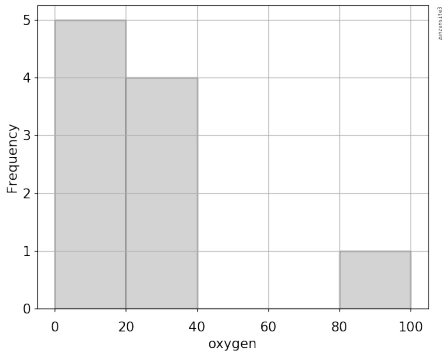
Source Code in MATLAB

```
% To the best of my knowledge this code is correct.
% If you find any errors or problems please contact
% me directly using zunzun@zunzun.com.
%
%      James


% Fitting target: lowest sum of squared absolute error
% Fitting target value = 0.0374858028499

function z = Polynomial_UserSelectablePolynomial_model(x_in, y_in)
    temp = 0.0;

    % coefficients
    a = 1.4610559504218952E+00;
    b = 1.0993977915597843E-01;
    c = 5.5707457038157646E-05;
    d = -6.6547586823996904E-02;
    f = 5.9370452670676590E-04;
    g = -1.4752241260518328E-06;

    temp = temp + a;
    temp = temp + b .* y_in;
    temp = temp + c .* power(y_in, 2.0);
    temp = temp + d .* x_in;
    temp = temp + f .* x_in .* y_in;
    temp = temp + g .* x_in .* power(y_in, 2.0);

    z = temp;
```

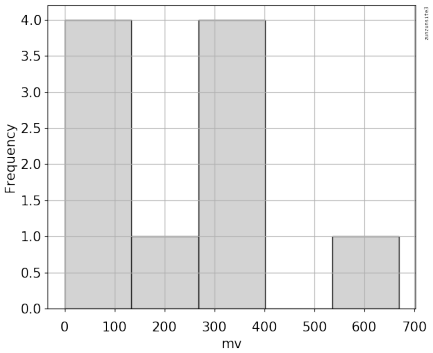
Source Code in VBA

```
' To the best of my knowledge this code is correct.
' If you find any errors or problems please contact
' me directly using zunzun@zunzun.com.
'
'      James

' Fitting target: lowest sum of squared absolute error
' Fitting target value = 0.0374858028499

Public Function Polynomial_UserSelectablePolynomial_model(x_in, y_in)
    temp = 0.0

    ' coefficients
    Const a = 1.4610559504218952E+00
    Const b = 1.0993977915597843E-01
    Const c = 5.5707457038157646E-05
    Const d = -6.6547586823996904E-02
    Const f = 5.9370452670676590E-04
    Const g = -1.4752241260518328E-06

    temp = temp + a
    temp = temp + b * y_in
    temp = temp + c * Application.WorksheetFunction.power(y_in, 2.0)
    temp = temp + d * x_in
    temp = temp + f * x_in * y_in
    temp = temp + g * x_in * Application.WorksheetFunction.power(y_in, 2.0)
    Polynomial_UserSelectablePolynomial_model = temp
End Function
```

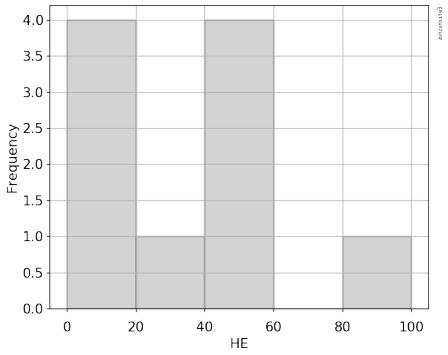
Histogram of oxygen



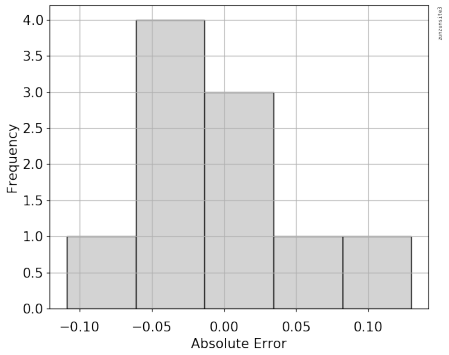
Histogram of mv



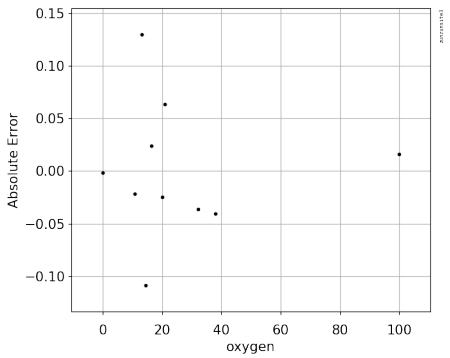
Histogram of HE



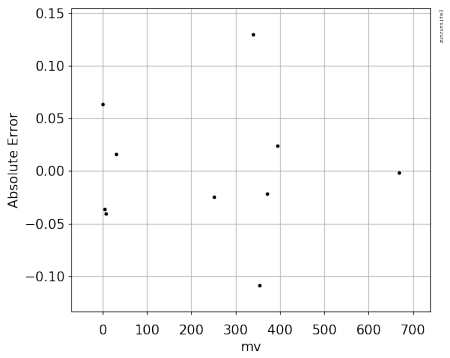
Histogram of Absolute Error



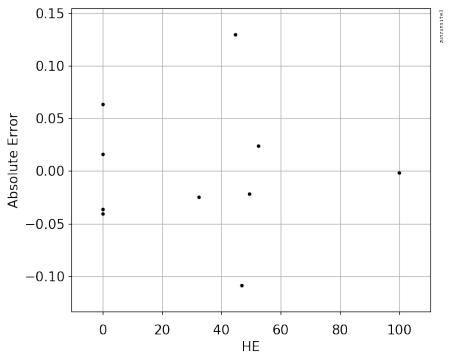
Absolute Error vs. oxygen



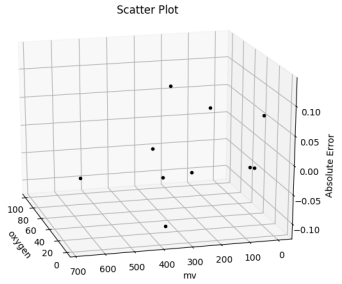
Absolute Error vs. mv



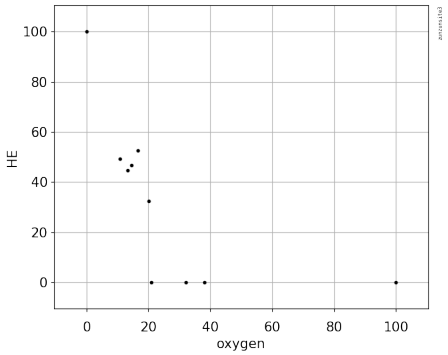
Absolute Error vs. HE



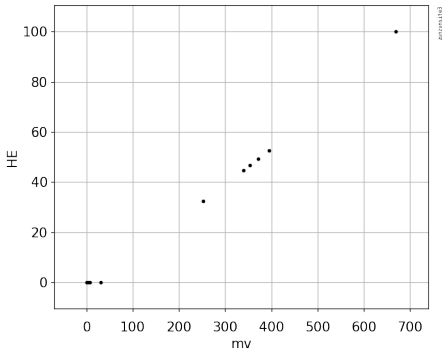
Absolute Error Scatter Plot



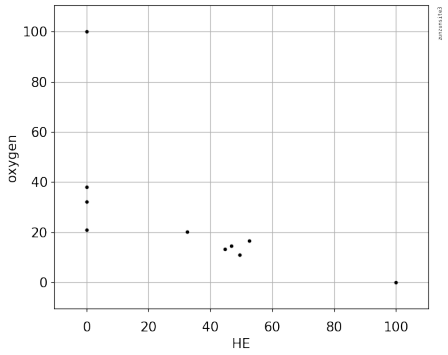
HE vs. oxygen



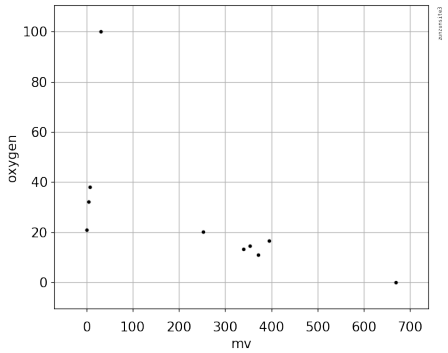
HE vs. mv



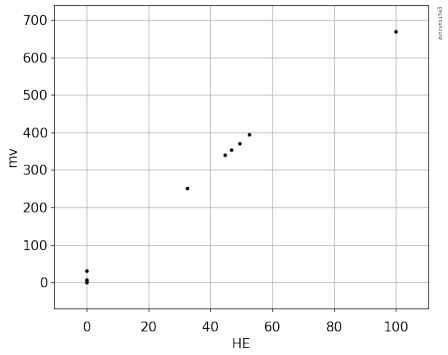
oxygen vs. HE



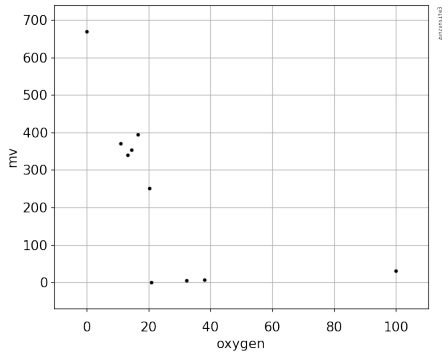
oxygen vs. mv



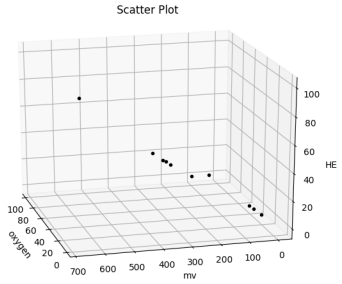
mv vs. HE



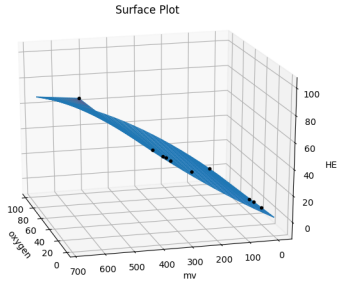
mv vs. oxygen



Scatter Plot



Surface Plot



Contour Plot