

Best Rational Heteronomial Model Selection]

by
Namir Clement Shammass

Contents

Introduction	1
The Case of Two Independent Variables.....	5
The Case of Three Independent Variables	23
About the ZIP File for this study	48

Introduction

Polynomials are popular constructs in math, calculus and regression analysis. Other popular constructs in regression analysis belong to multiple linear(ized) models. A simple example of a multiple regression model with two independent variables is:

$$Y = a + b_1 * X_1 + b_2 * X_2 \quad (1)$$

I will call the above model a linear *heteronomial*. It is linear because all the variables are in linear form. A simple example of a nonlinear heteronomial is:

$$Y = a + b_1 / X_1 + b_2 * X_2^2 \quad (2)$$

The above model has a linear value for the dependent variable, raises variable X_1 to the power of -1, and raises variable X_2 to power 2.

The general form of a heteronomial is:

$$Y^p = a + b_1 * X_1^{px1} + b_2 * X_2^{px2} + \dots + X_n^{pxn} \quad (3)$$

The above heteronomial shows that each variable in the model can have a power other than 1. The powers can be negative, zero (special code for using the function $\ln(x)$), and positive. The non-zero powers can be integers or floating-point numbers. Using floating point negative powers requires the data to be positive.

I can generalize equation (3) further by using the following form:

$$f(Y, yp) = a + b_1 * f(X_1, px1) + b_2 * f(X_2, px2) + \dots + b_n * f(X_n, pxn) \quad (4)$$

$$\begin{aligned} \text{Where } f(x, p) &= x^p \text{ when } p \neq 0 \\ &= \ln(x) \text{ when } p = 0 \end{aligned} \quad (5)$$

This study will focus on heteronomials with the above kind of power transformations. Using other functions like trigonometric and hyperbolic functions requires a more elaborate power-coding scheme to map special powers to special functions. The power management scheme uses distinct initial powers, power increments, and final powers for each variable.

Please note that the total number of models tested is the product of the total number of transformations for each variable. The total number of models can easily increase to very high values with the increase in the number of transformations for each variable.

Padé polynomials take the ratio of two polynomials. The general form of a Padé polynomial is:

$$Y = (a + b_1 * X + b_2 * X^2 + \dots + b_m * X^m) / (1 + c_1 * X + c_2 * X^2 + \dots + c_n * X^n) \quad (6)$$

Notice that the intercept of the denominator polynomial is always 1. The Padé polynomials have orders of m and n.

Now let me introduce you to **rational heteronomials**. They are to heteronomials what Pade polynomials are to regular polynomials. The general form of rational heteronomials is:

$$\begin{aligned} Y^{py} &= (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \dots + b_n * X_n^{px1n}) / \\ &\quad (1 + c_1 * X_1^{px21} + c_2 * X_2^{px22} + \dots + c_n * X_n^{px2n}) \end{aligned} \quad (7)$$

Each variable in equation (7) has its own power. In fact, equation (7) shows the general form of *symmetrical rational heteronomials* that have the same independent variables appearing in the numerator and denominator, albeit being raised to different powers. The minimal condition for rational heteronomials is that

at least one independent variable must appear in both numerator and denominator.

A few general examples of non-symmetrical rational heteronomials appear next:

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \dots + b_n * X_n^{px1n}) / (1 + c_1 * X_1^{px21} + c_2 * X_2^{px22} + \dots + c_m * X_m^{px2m}) \quad (7b)$$

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \dots + b_n * X_n^{px1n}) / (1 + c_1 * X_1^{px21} + c_2 * X_3^{px23} + \dots + c_m * X_m^{px2m}) \quad (7c)$$

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \dots + b_n * X_n^{px1n}) / (1 + c_1 * X_1^{px21} + c_2 * X_5^{px25} + \dots + c_m * X_m^{px2m}) \quad (7d)$$

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_5^{px15} + \dots + b_n * X_n^{px1n}) / (1 + c_1 * X_1^{px21} + c_2 * X_3^{px23} + \dots + c_m * X_m^{px2m}) \quad (7e)$$

Selecting the best combination of variables in the numerator and denominator can be very tedious and more complicated than determining the best orders n and m for a Pade polynomial. For example, given two independent variables, one must search through the best model in the following sets of general models:

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px22}) / (1 + c_1 * X_1^{px21} + c_2 * X_2^{px22}) \quad (8a)$$

$$Y^{py} = (a + b_1 * X_1^{px11}) / (1 + c_1 * X_1^{px21} + c_2 * X_2^{px22}) \quad (8b)$$

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px22}) / (1 + c_1 * X_1^{px21}) \quad (8c)$$

$$Y^{py} = (a + b_1 * X_2^{px12}) / (1 + c_1 * X_1^{px21} + c_2 * X_2^{px22}) \quad (8b)$$

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px22}) / (1 + c_1 * X_2^{px22}) \quad (8c)$$

So, two independent variables will require you to find the best fit in five sets of rational heteronomials! The case of three independent variables will require you to find the best fit in 13 sets of rational heteronomials. In the case of 4 independent variables, one must find the best fit in 29 sets of rational heteronomials. It's easy to

see how the number of sets keep climbing as the number of independent variables increases! The search will require several model selection functions that test various kinds of models. For example, in the above case of two independent variables, we need three model selection functions to determine the best of 5 sets of models.

This study looks at the selection of the best type of rational heteronomials models as shown in equations (8a) to (8c). Moreover, I will present programs that work with three independent variables in models that are supersets of those in equations (8a) to (8c). I will use MATLAB and Excel data files. The program uses MATLAB code.

The following table lists the model selection functions and the configurations of the independent variables appearing in the numerator and denominator of the rational heteronomials.

Model Selection Function	Variables in the Numerator	Variables in the Denominator	Number of Cases
fx422	X1and X2	X1 and X2	1
fx421	X1and X2	X1 or X2	2
fx412	X1 or X2	X1 and X2	2

Table 1. Case for two independent variables.

The above table shows that the model selection function fx422() needs to be called once. By contrast, the functions fx421() and fx412() need to be called twice—once for variable X1 and once for variable X2.

In the case of three independent variables the following table lists the model selection functions and the configurations of the independent variables appearing in the numerator and denominator of the rational heteronomials.

Model Selection Function	Variables in the Numerator	Variables in the Denominator	Number of Cases
fx633	X1, X2and X3	X1, X2and X3	1
fx632	X1, X2 and X3	Any two variables of X1,X2, and X3	3
fx631	X1, X2 and X3	X1, or X2 or X3	3

fx623	Any two variables of X1,X2, and X3	X1, X2 and X3	3
fx613	X1, or X2 or X3	X1, X2 and X3	3

Table 2. Case for three independent variables.

The above table shows that the model selection function fx633() needs to be called once. By contrast, the functions fx632() and fx623() need to be called three times—each time with a different combination of two variables. The functions fx631() and fx613() also need to be called three times—each time with a different single independent variable. The cases in figures 1 and 2 assume that **at least** the numerator or the denominator part of the rational heteronomials has terms for the entire set of independent variables. This assumption keeps the total number of different model types a bit under control. **Keep the above tables in mind!**

The Case of Two Independent Variables

Let's look at the case of two independent variables as shown in table (1). The Excel file go44.xlsx is the one I use to handle modeling with the scheme outlined by table 1. The workbook has the following sheets:

- The **Data** sheet has the following columns (see Figure 1):
 - Column A contains the values for the dependent variable Y. The column has the header Y in the first row.
 - Column B, C, and D contain the values for the three independent variables. The columns have the headers X1, X2, and X3 in the first row.
- The **T422** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.1):
 - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
 - Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
 - Column C has the header X2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the numerator part of the rational heteronomial.

- Column D has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
- Column E has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
- Sheet **T4211** contains information for the transformation ranges. It has the following columns (see Figure 2.2):
 - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
 - Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
 - Column C has the header X2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the numerator part of the rational heteronomial.
 - Column D has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
- Sheet **T4212** contains information for the transformation ranges. It has the following columns (see Figure 2.3):
 - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
 - Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
 - Column C has the header X2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the numerator part of the rational heteronomial.
 - Column D has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
- The **T4121** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.4):

- Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
- Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
- Column C has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
- Column D has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
- The **T4122** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.5):
 - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
 - Column B has the header X2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the numerator part of the rational heteronomial.
 - Column C has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
 - Column D has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.

Y	X1	X2
12.18213323	1	55
12.54696677	2	67
8.430565344	3	41
6.604571883	4	29
6.00516746	5	25
5.482942824	6	21
5.63198061	7	22
6.274235797	8	27
4.363787374	9	10
8.4397117	10	45
4.759636712	11	12
5.167160174	12	15
5.364859283	13	16
8.9211312	14	48
5.077018806	15	11

Figure 1. The **Data** sheet in file go44.xlsx.

Y	X1	X2	YX1	YX2
0	0	0	0	0
1	1	1	1	1
3	3	3	3	3

Figure 2.1. The **T422** sheet in file go44.xlsxt.

Y	X1	X2	YX1
0	0	0	0
1	1	1	1
3	3	3	3

Figure 2.2. The **T4211** sheet in file go44.xlsxt.

Y	X1	X2	YX2
0	0	0	0
1	1	1	1
3	3	3	3

Figure 2. The **T4212** sheet in file go44.xlsxt.

Y	X1	YX1	YX2
0	0	0	0
1	1	1	1
3	3	3	3

Figure 2.3. The **T4121** sheet in file go44.xlsxt.

Y	X2	YX1	YX2
0	0	0	0
1	1	1	1
3	3	3	3

Figure 2.4. The **T4122** sheet in file go44.xlsxt.

Here is the MATLAB code for function f422():

```
function [bestAdjR2,S] = fx422(ydata,xdata,xyTransf)
%FX422 Summary of this function goes here
% Detailed explanation goes here
S.model = "(X1+X2)/(X1+X2)";
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
yx1Pwrs = xyTransf(1:3,4);
yx2Pwrs = xyTransf(1:3,5);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end
bestAdjR2 = 0;
yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
```

```

x1Cols = x1Cols + 1;
x1 = x1Buffer(:,x1Cols);

x2Cols = 0 ;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2 = x2Buffer(:,x2Cols);

    yx1Cols = 0 ;
    for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
        yx1Cols = yx1Cols + 1;
        yx1 = yx1Buffer(:,yx1Cols);

        yx1 = -1 .* y .* yx1;

        yx2Cols = 0 ;
        for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
            yx2Cols = yx2Cols + 1;
            yx2 = yx2Buffer(:,yx2Cols);

            yx2 = -1 .* y .* yx2;
            X = [x1 x2 yx1 yx2];

            mdl = fitlm(X,y);

            if mdl.Rsquared.Adjusted >= bestAdjR2
                bestAdjR2 = mdl.Rsquared.Adjusted;
                S.bestAdjR2 = bestAdjR2;
                S.bestYpwr = yPwr;
                S.bestX1pwr = x1Pwr;
                S.bestX2pwr = x2Pwr;
                S.bestYX1pwr = yx1Pwr;
                S.bestYX2pwr = yx2Pwr;
                S.bestmdl = mdl;
            end
        end
    end
end
end
end
end
end

end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The function f422() has the following input parameters:

- The parameter ydata passes a column vector for the Y values.

- The parameter `xdata` passes a matrix with three columns for the independent variables.
- The parameter `xyTransf` passes the matrix containing the transformation range information.

The function returns two parameters—the best adjusted R-square statistic and the structure containing information for the best model obtained by the function.

The function performs the following general tasks:

- Determine the number of data points (i.e. rows) in the variable `ydata`.
- Initialize the buffer matrices `yBuffer`, `x1Buffer`, and `x2Buffer`. The program calculates once the various transformations for the different variables and stores them in these buffers. The program recalls specific columns in each buffer as needed.
- Initialize the best adjusted R-square value.
- Use four loops to calculate the data in the four matrix buffers used.
- Start the main process using four nested loops to access the transformed values of Y and X from the buffers.
- Create the regression matrix X and the vector column y needed to perform the multiple linearized regression using the MATLAB function `fitlm()`.
- Determine if the new model object has a better adjusted R-square value than the one store in variable `bestAdjR2`. If this condition is true, the function stores the adjusted R-square value, the powers for X and Y, and the model object in structure `S`.

Here is the MATLAB code for function `f421()`:

```
function [bestAdjR2,S] = fx421(ydata,xdata,x3Idx,xyTransf)
%FX421 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat("(X1+X2)/(X",num2str(x3Idx),")");
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
% x3Idx should be 1 or 2
x3data = xdata(:,x3Idx);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
yx1Pwrs = xyTransf(1:3,4);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
```

```

yx1Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x3data,yx1Pwr);
end

bestAdjR2 = 0;
yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0 ;
        for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
            x2Cols = x2Cols + 1;
            x2 = x2Buffer(:,x2Cols);

            yx1Cols = 0 ;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                X = [x1 x2 yx1];

                mdl = fitlm(X,y);

                if mdl.Rsquared.Adjusted >= bestAdjR2
                    bestAdjR2 = mdl.Rsquared.Adjusted;
                    S.bestAdjR2 = bestAdjR2;
                    S.bestYpwr = yPwr;
                end
            end
        end
    end
end

```

```

        S.bestX1pwr = x1Pwr;
        S.bestX2pwr = x2Pwr;
        S.bestYX1pwr = yx1Pwr;
        S.bestxIdx = x3Idx;
        S.bestmdl = mdl;
    end
end
end
end
end

end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The code for function f421() is a subset of that of function f422() as it works with three terms. The different variables, arrays, matrices, and loops are organized to handle the two terms in numerator part and one term in the denominator part.

Here is the MATLAB code for function f412():

```

function [bestAdjR2,S] = fx412(ydata,xdata,x3Idx,xyTransf)
%FX412 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat(" (X",num2str(x3Idx),") / (X1+X2) ");
maxrows = length(ydata);
yx1data = xdata(:,1);
yx2data = xdata(:,2);
% x3Idx should be 1 or 2
x1data = xdata(:,x3Idx);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
yx1Pwrs = xyTransf(1:3,3);
yx2Pwrs = xyTransf(1:3,4);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

```

```

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(yx1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(yx2data,yx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        yx1Cols = 0 ;
        for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
            yx1Cols = yx1Cols + 1;
            yx1 = yx1Buffer(:,yx1Cols);

            yx1 = -1 .* y .* yx1;

            yx2Cols = 0 ;
            for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                yx2Cols = yx2Cols + 1;
                yx2 = yx2Buffer(:,yx2Cols);

                yx2 = -1 .* y .* yx2;

            X = [x1 yx1 yx2];
            mdl = fitlm(X,y);

            if mdl.Rsquared.Adjusted >= bestAdjR2
                bestAdjR2 = mdl.Rsquared.Adjusted;
                S.bestAdjR2 = bestAdjR2;
                S.bestYpwr = yPwr;
                S.bestX1pwr = x1Pwr;
                S.bestYX1pwr = yx1Pwr;
                S.bestYX2pwr = yx2Pwr;
                S.bestxIdx = x3Idx;
                S.bestmdl = mdl;
            end
        end
    end
end

```

```

        end
    end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The code for function f412() is a subset of that of function f422() as it works with three terms. The different variables, arrays, matrices, and loops are organized to handle the one term in numerator part and two terms in the denominator part.

Here is the code for the test program go44.m:

```

clc
clear all

zBest = 0;

filename = "Go44.xlsx";
xyData = readmatrix(filename,'Sheet','Data');
ydata =xyData(:,1);
xdata = xyData(:,2:end);

xyTransf = readmatrix(filename,'Sheet','T422');
[R422,S422] = fx422(ydata,xdata,xyTransf)
S422.bestmdl
if R422 > zBest
    zBest = R422;
    bestS = S422;
end

xyTransf = readmatrix(filename,'Sheet','T4211');
[R4211,S4211] = fx421(ydata,xdata,1,xyTransf)
S4211.bestmdl
if R4211 > zBest
    zBest = R4211;
    bestS = S4211;
end

xyTransf = readmatrix(filename,'Sheet','T4212');
[R4212,S4212] = fx421(ydata,xdata,2,xyTransf)
S4212.bestmdl
if R4212 > zBest

```



```

    zBest = R4212;
    bestS = S4212;
end

xyTransf = readmatrix(filename, 'Sheet', 'T4121');
[R4121, S4121] = fx412(ydata, xdata, 1, xyTransf)
S4121.bestmdl
if R4121 > zBest
    zBest = R4121;
    bestS = S4121;
end

xyTransf = readmatrix(filename, 'Sheet', 'T4122');
[R4122, S4122] = fx412(ydata, xdata, 2, xyTransf)
S4122.bestmdl
if R4122 > zBest
    zBest = R4122;
    bestS = S4122;
end

fprintf("\n===== \n\n")
fprintf("----- Best Model ----- \n\n");
bestS
bestS.bestmdl

```

The code for the test program performs the following tasks:

1. Initialize the best adjusted R-square value stored in variable **zBest**.
2. Read the data matrix in sheet **Data** of file **go44.xlsx**. This task stores the input matrix in variable **xyData**,
3. Copy the first column of matrix **xyData** into the column vector **ydata**.
4. Copy the rest of the columns in matrix **xyData** into the matrix **xdata**.
5. Read the transformation in sheet **T422** into variable **xyTransf**.
6. Call function **fx422()** with arguments **ydata**, **xdata**, and **xyTransf**. This task stores the results into variable **R422** and structure **S422**, and displays them.
7. Display the best model stored in object **S422.bestmdl**.
8. If **R422** is greater than variable **zBest**, update **zBest** with the value in **R422** and copy structure **S422** into **bestS**.
9. Repeat steps 5 to 8 by using the transformations in sheet **T4211** and calling function **f421()**. This set of tasks works with the results into variable **R4211** and structure **S4211**.
10. Repeat steps 5 to 8 by using the transformations in sheet **T4212** and calling function **f421()**. This set of tasks works with the results into variable **R4212** and structure **S4212**.
11. Repeat steps 5 to 8 by using the transformations in sheet **T4121** and calling function **f412()**. This set of tasks works with the results into variable **R4121** and structure **S4121**.

12.Repeat steps 5 to 8 by using the transformations in sheet **T4122** and calling function f412(). This set of tasks works with the results into variable R4122 and structure S4122.

13. Display the very best model object stored in structure bestS.

14.Display the best regression results object stored in object bestS.bestmdl.

Open the Excel file go44.xlsx before you run the MATLAB program. If needed, enter or edit the data in sheets **Data** and various transformation sheets. Once you are done you **must close the Excel file** before you run the MATLAB program. Examine the results in the MATLAB console. Here is the console output:

```
R422 =

    1

S422 =

struct with fields:

    model: "(X1+X2) / (X1+X2) "
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:

               Estimate      SE      tStat      pValue
    _____
(Intercept)         5         0         Inf         0
x1                   1         0         Inf         0
x2                   1         0         Inf         0
x3                   1         0         Inf         0
x4                   1         0         Inf         0

Number of observations: 15, Error degrees of freedom: 10
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.58e+29, p-value = 6.03e-144

R4211 =
```

```

0.9996

S4211 =

  struct with fields:

    model: "(X1+X2)/(X1)"
    bestAdjR2: 0.9996
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestYX1pwr: 0
    bestxIdx: 1
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
  y ~ 1 + x1 + x2 + x3

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	2.739	0.050011	54.768	9.2719e-15
x1	0.21837	0.0065372	33.405	2.0677e-12
x2	0.16764	0.0011004	152.35	1.2206e-19
x3	0.20849	0.0046891	44.461	9.0993e-14

```

Number of observations: 15, Error degrees of freedom: 11
Root Mean Squared Error: 0.0487
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.31e+04, p-value = 8.25e-20

R4212 =

0.9999

S4212 =

  struct with fields:

    model: "(X1+X2)/(X2)"
    bestAdjR2: 0.9999
    bestYpwr: 2
    bestX1pwr: 3
    bestX2pwr: 3
    bestYX1pwr: 0
    bestxIdx: 2
    bestmdl: [1×1 LinearModel]

```

```
ans =
```

```
Linear regression model:
  y ~ 1 + x1 + x2 + x3
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	7.5919	0.30924	24.55	5.8699e-11
x1	0.00052818	0.00012078	4.3732	0.0011116
x2	-6.1908e-05	5.8641e-06	-10.557	4.2893e-07
x3	-0.25411	0.002606	-97.508	1.6483e-17

```
Number of observations: 15, Error degrees of freedom: 11
```

```
Root Mean Squared Error: 0.455
```

```
R-squared: 1, Adjusted R-Squared: 1
```

```
F-statistic vs. constant model: 4.33e+04, p-value = 1.13e-22
```

```
R4121 =
```

```
0.9997
```

```
S4121 =
```

```
struct with fields:
```

```
    model: "(X1)/(X1+X2)"
  bestAdjR2: 0.9997
    bestYpwr: 1
    bestX1pwr: 2
  bestYX1pwr: 0
  bestYX2pwr: 0
    bestIdx: 1
    bestmdl: [1×1 LinearModel]
```

```
ans =
```

```
Linear regression model:
  y ~ 1 + x1 + x2 + x3
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	2.5296	0.040359	62.677	2.1119e-15
x1	0.0029321	0.00024964	11.745	1.4497e-07
x2	0.040829	0.0032262	12.655	6.7218e-08
x3	-0.19736	0.0010131	-194.81	8.1766e-21

```
Number of observations: 15, Error degrees of freedom: 11
```

```

Root Mean Squared Error: 0.0447
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.55e+04, p-value = 3.21e-20

R4122 =

    0.9998

S4122 =

    struct with fields:

        model: "(X2)/(X1+X2)"
        bestAdjR2: 0.9998
        bestYpwr: 3
        bestX1pwr: 3
        bestYX1pwr: 0
        bestYX2pwr: 0
        bestxIdx: 2
        bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
    y ~ 1 + x1 + x2 + x3

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	29.353	4.747	6.1835	6.8741e-05
x1	-0.0013387	0.00017453	-7.6707	9.7197e-06
x2	-0.040404	0.0076949	-5.2508	0.00027226
x3	-0.27647	0.005167	-53.507	1.1969e-14

```

Number of observations: 15, Error degrees of freedom: 11
Root Mean Squared Error: 7.68
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 2.87e+04, p-value = 1.09e-21

=====

----- Best Model -----

bestS =

    struct with fields:

        model: "(X1+X2)/(X1+X2)"
        bestAdjR2: 1
        bestYpwr: 1
        bestX1pwr: 1

```

```

bestX2pwr: 1
bestYX1pwr: 0
bestYX2pwr: 0
bestmdl: [1x1 LinearModel]

ans =

Linear regression model:
y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	5	0	Inf	0
x1	1	0	Inf	0
x2	1	0	Inf	0
x3	1	0	Inf	0
x4	1	0	Inf	0

```

Number of observations: 15, Error degrees of freedom: 10
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.58e+29, p-value = 6.03e-144


```

The console output shows the regression table including the regression coefficients, their standard errors, the student-t values, and the p-Values. The latter values can shed valuable insight as to the eligibility of a variable/term to be part of the model. The best model is:

$$Y = (5 + X_1 + X_2) / (1 + \ln(X_1) + \ln(X_2))$$

The constant 1 in the denominator part is NOT calculated. It is an implicit part of the definition of the rational heteronomial. This fact is true for ALL rational heteronomials.

The MATLAB code contains the function `fx()` that transforms the value of a variable using a power value. The current implementation of function `fx()` handle negative, zero, and positive powers. Notice that the power zero causes function `fx()` to return the natural logarithm value of x .

 Should you desire to use other functions for transformations, then you need to code function `fx()` to detect the supplied power values and use them to evaluate a special function. For example, if you are using the range of -3 to 3

in increments of 1 for regular powers, you can use the powers of 4 and 5 to evaluate, for example, the sine and cosine functions. The function `fx()` should use separate if statements to detect the value of 4 and 5 and return `sin(x)` and `cos(x)`, respectively. The code for function `fx()` would look like:

```
function y = fx(x,pwr):
    if pwr == 4
        y = sin(x)
        return;
    if pwr == 5
        y = cos(x);
        return;
    if pwr > 0:
        y = x.^pwr;
    elif pwr < 0:
        y = 1./x.^abs(pwr);
    else:
        y = log(x);
end
```

As shown above, function `fx()` should detect the special coded powers first.

The Case of Three Independent Variables

In this section, I present functions to select the best model types based on Table 2. The code for these functions handles three independent variables. The functions listed in Table 2 are similar supersets of the ones in Table 1. In addition, Table 2 shows five model selection functions versus three in Table 1, Table 1 also indicates that the test program should make 13 calls to the various model selection functions in Table 2. Since the functions in Table 2 handle more variables than the functions in Table 1, they work with more arrays, matrices, loops, and bigger regression matrices.

Figure 3 shows the sheet **Data**. Figures 4.x shows the various transformation sheets **T6xxx**.

Y	X1	X2	X3
12.02632766	1	55	45
10.62359111	2	67	21
10.59115171	3	41	55
10.37472138	4	29	65
7.964882578	5	25	41
6.627857312	6	21	29
7.193269104	7	22	35
7.473510245	8	27	34
4.751543108	9	10	15
9.980700421	10	45	50
5.083537163	11	12	16
6.217860959	12	15	27
5.709990411	13	16	19
9.418861198	14	48	38
5.005324196	15	11	12

Figure 3. The **Data** sheet in file *go66.xlsx*.

Y	X1	X2	X3	YX1	YX2	YX3
0	0	0	0	0	0	0
1	1	1	1	1	1	1
3	3	3	3	3	3	3

Figure 4.1. The **T633** sheet in file *go66.xlsx*.

Y	X1	X2	X3	YX1	YX2
0	0	0	0	0	0
1	1	1	1	1	1
3	3	3	3	3	3

Figure 4.2. The **T632** sheet in file *go66.xlsx*.

Y	X1	X2	YX1	YX2	YX3
0	0	0	0	0	0
1	1	1	1	1	1
3	3	3	3	3	3

Figure 4.3. The **T623** sheet in file *go66.xlsx*.

Y	X1	X2	X3	YX1
0	0	0	0	0
1	1	1	1	1
3	3	3	3	3

Figure 4.51. The **T631** sheet in file *go66.xlsx*.

Y	X2	YX1	YX2	YX3
0	0	0	0	0
1	1	1	1	1
3	3	3	3	3

Figure 4.6. The **T613** sheet in file *go66.xlsx*.

Here is the MATLAB code for function f633():

```
function [bestAdjR2,S] = fx633(ydata,xdata,xyTransf)
%FX633 Summary of this function goes here
% Detailed explanation goes here
S.model = "(X1+X2+X3)/(X1+X2+X3)";
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
x3data = xdata(:,3);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
x3Pwrs = xyTransf(1:3,4);
yx1Pwrs = xyTransf(1:3,5);
yx2Pwrs = xyTransf(1:3,6);
yx3Pwrs = xyTransf(1:3,7);
yBuffer = zeros(maxrows,1);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
x3Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yx3Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

x3Cols = 0;
for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)
    x3Cols = x3Cols + 1;
    x3Buffer(:,x3Cols)= fx(x3data,x3Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
```

```

for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end

yx3Cols = 0;
for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)
    yx3Cols = yx3Cols + 1;
    yx3Buffer(:,yx3Cols)= fx(x3data,yx3Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0 ;
        for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
            x2Cols = x2Cols + 1;
            x2 = x2Buffer(:,x2Cols);

            x3Cols = 0 ;
            for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)
                x3Cols = x3Cols + 1;
                x3 = x3Buffer(:,x3Cols);

                yx1Cols = 0 ;
                for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                    yx1Cols = yx1Cols + 1;
                    yx1 = yx1Buffer(:,yx1Cols);

                    yx1 = -1 .* y .* yx1;

                    yx2Cols = 0 ;
                    for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                        yx2Cols = yx2Cols + 1;
                        yx2 = yx2Buffer(:,yx2Cols);

                        yx2 = -1 .* y .* yx2;

                        yx3Cols = 0 ;
                        for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)
                            yx3Cols = yx3Cols + 1;
                            yx3 = yx3Buffer(:,yx3Cols);

                            yx3 = -1 .* y .* yx3;

                            X = [x1 x2 x3 yx1 yx2 yx3];

                            mdl = fitlm(X,y);

```

```

        if mdl.Rsquared.Adjusted >= bestAdjR2
            bestAdjR2 = mdl.Rsquared.Adjusted;
            S.bestAdjR2 = bestAdjR2;
            S.bestYpwr = yPwr;
            S.bestX1pwr = x1Pwr;
            S.bestX2pwr = x2Pwr;
            S.bestX3pwr = x3Pwr;
            S.bestYX1pwr = yx1Pwr;
            S.bestYX2pwr = yx2Pwr;
            S.bestYX3pwr = yx3Pwr;
            S.bestmdl = mdl;
        end
    end
end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

Here is the MATLAB code for function f632():

```

function [bestAdjR2,S] = fx632(ydata,xdata,x1Idx,x2Idx,xyTransf)
%FX632 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat("(X1+X2+X3)/(X", num2str(x1Idx),"X", num2str(x2Idx),"");
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
x3data = xdata(:,3);
yx1data = xdata(:,x1Idx);
yx2data = xdata(:,x2Idx);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
x3Pwrs = xyTransf(1:3,4);
yx1Pwrs = xyTransf(1:3,5);
yx2Pwrs = xyTransf(1:3,6);
yBuffer = zeros(maxrows,1);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
x3Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);

```

```

yx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

x3Cols = 0;
for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)
    x3Cols = x3Cols + 1;
    x3Buffer(:,x3Cols)= fx(x3data,x3Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(yx1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(yx2data,yx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0 ;
        for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
            x2Cols = x2Cols + 1;
            x2 = x2Buffer(:,x2Cols);

            x3Cols = 0 ;
            for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)

```

```

x3Cols = x3Cols + 1;
x3 = x3Buffer(:,x3Cols);

yx1Cols = 0 ;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1 = yx1Buffer(:,yx1Cols);

    yx1 = -1 .* y .* yx1;

    yx2Cols = 0 ;
    for yx2Pwr =yx2Pwrs(2):yx2Pwrs(2):yx2Pwrs(3)
        yx2Cols = yx2Cols + 1;
        yx2 = yx2Buffer(:,yx2Cols);

        yx2 = -1 .* y .* yx2;

    X = [x1 x2 x3 yx1 yx2];

    mdl = fitlm(X,y);

    if mdl.Rsquared.Adjusted >= bestAdjR2
        bestAdjR2 = mdl.Rsquared.Adjusted;
        S.bestAdjR2 = bestAdjR2;
        S.bestYpwr = yPwr;
        S.bestX1pwr = x1Pwr;
        S.bestX2pwr = x2Pwr;
        S.bestX3pwr = x3Pwr;
        S.bestYX1pwr = yx1Pwr;
        S.bestYX2pwr = yx2Pwr;
        S.bestx1Idx = x1Idx;
        S.bestx2Idx = x2Idx;
        S.bestmdl = mdl;
    end
end
end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

Notice that function f632() has the additional input parameters x1Idx and x2Idx to select the two independent variables that appear in the denominator part.

Here is the MATLAB code for function f631():

```
function [bestAdjR2,S] = fx631(ydata,xdata,x1Idx,xyTransf)
%FX631 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat("(X1+X2+X3)/(X", num2str(x1Idx),")");
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
x3data = xdata(:,3);
yx1data = xdata(:,x1Idx);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
x3Pwrs = xyTransf(1:3,4);
yx1Pwrs = xyTransf(1:3,5);
yBuffer = zeros(maxrows,1);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
x3Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

x3Cols = 0;
for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)
    x3Cols = x3Cols + 1;
    x3Buffer(:,x3Cols)= fx(x3data,x3Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(yx1data,yx1Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
```

```

yCols = yCols + 1;
y = yBuffer(:,yCols);

x1Cols = 0 ;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1 = x1Buffer(:,x1Cols);

    x2Cols = 0 ;
    for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
        x2Cols = x2Cols + 1;
        x2 = x2Buffer(:,x2Cols);

        x3Cols = 0 ;
        for x3Pwr =x3Pwrs(1):x3Pwrs(2):x3Pwrs(3)
            x3Cols = x3Cols + 1;
            x3 = x3Buffer(:,x3Cols);

            yx1Cols = 0 ;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                X = [x1 x2 x3 yx1];

                mdl = fitlm(X,y);

                if mdl.Rsquared.Adjusted >= bestAdjR2
                    bestAdjR2 = mdl.Rsquared.Adjusted;
                    S.bestAdjR2 = bestAdjR2;
                    S.bestYpwr = yPwr;
                    S.bestX1pwr = x1Pwr;
                    S.bestX2pwr = x2Pwr;
                    S.bestX3pwr = x3Pwr;
                    S.bestYX1pwr = yx1Pwr;
                    S.bestx1Idx = x1Idx;
                    S.bestmdl = mdl;
                end
            end
        end
    end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```


Notice that function `f631()` has the additional input parameter `x1Idx` to select the independent variable that appears in the denominator part.

Here is the MATLAB code for function `f623()`:

```
function [bestAdjR2,S] = fx623(ydata,xdata,x1Idx,x2Idx,xyTransf)
%FX623 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat(" (X", num2str(x1Idx), "+X", num2str(x2Idx), ") / (X1+X2+X3) ");
maxrows = length(ydata);
x1data = xdata(:,x1Idx);
x2data = xdata(:,x2Idx);
yx1data = xdata(:,1);
yx2data = xdata(:,2);
yx3data = xdata(:,3);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
yx1Pwrs = xyTransf(1:3,4);
yx2Pwrs = xyTransf(1:3,5);
yx3Pwrs = xyTransf(1:3,6);
yBuffer = zeros(maxrows,1);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yx3Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(yx1data,yx1Pwr);
end

yx2Cols = 0;
```

```

for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(yx2data,yx2Pwr);
end

yx3Cols = 0;
for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)
    yx3Cols = yx3Cols + 1;
    yx3Buffer(:,yx3Cols)= fx(x3data,yx3Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0 ;
        for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
            x2Cols = x2Cols + 1;
            x2 = x2Buffer(:,x2Cols);

            yx1Cols = 0 ;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                yx2Cols = 0 ;
                for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                    yx2Cols = yx2Cols + 1;
                    yx2 = yx2Buffer(:,yx2Cols);

                    yx2 = -1 .* y .* yx2;

                    yx3Cols = 0 ;
                    for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)
                        yx3Cols = yx3Cols + 1;
                        yx3 = yx3Buffer(:,yx3Cols);

                        yx3 = -1 .* y .* yx3;

                        X = [x1 x2 yx1 yx2 yx3];

                        mdl = fitlm(X,y);

                        if mdl.Rsquared.Adjusted >= bestAdjR2
                            bestAdjR2 = mdl.Rsquared.Adjusted;
                            S.bestAdjR2 = bestAdjR2;
                            S.bestYpwr = yPwr;

```

Notice that function `f623()` has the additional input parameters `x1Idx` and `x2Idx` to select the two independent variables that appear in the numerator part.

```
function [bestAdjR2,S] = fx613(ydata,xdata,x1Idx,xyTransf)
%FX613 Summary of this function goes here
% Detailed explanation goes here
S.model = strcat("X", num2str(x1Idx), ")/(X1+X2+X3)");
maxrows = length(ydata);
yx1data = xdata(:,1);
yx2data = xdata(:,2);
yx3data = xdata(:,3);
x1data = xdata(:,x1Idx);
yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
yx1Pwrs = xyTransf(1:3,3);
yx2Pwrs = xyTransf(1:3,4);
yx3Pwrs = xyTransf(1:3,5);
yBuffer = zeros(maxrows,1);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yx3Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr = yPwrs(1):yPwrs(2):yPwrs(3)
```

```

    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(yx1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(yx2data,yx2Pwr);
end

yx3Cols = 0;
for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)
    yx3Cols = yx3Cols + 1;
    yx3Buffer(:,yx3Cols)= fx(yx3data,yx3Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0 ;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        yx1Cols = 0 ;
        for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
            yx1Cols = yx1Cols + 1;
            yx1 = yx1Buffer(:,yx1Cols);

            yx1 = -1 .* y .* yx1;

            yx2Cols = 0 ;
            for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                yx2Cols = yx2Cols + 1;
                yx2 = yx2Buffer(:,yx2Cols);

                yx2 = -1 .* y .* yx2;

                yx3Cols = 0;
                for yx3Pwr =yx3Pwrs(1):yx3Pwrs(2):yx3Pwrs(3)

```

```

        yx3Cols = yx3Cols + 1;
        yx3 = yx3Buffer(:,yx3Cols);

        yx3 = -1 .* y .* yx3;

        X = [x1 yx1 yx2 yx3];

        mdl = fitlm(X,y);

        if mdl.Rsquared.Adjusted >= bestAdjR2
            bestAdjR2 = mdl.Rsquared.Adjusted;
            S.bestAdjR2 = bestAdjR2;
            S.bestYpwr = yPwr;
            S.bestYX1pwr = yx1Pwr;
            S.bestYX2pwr = yx2Pwr;
            S.bestYX3pwr = yx3Pwr;
            S.bestX1pwr = x1Pwr;
            S.bestxIdx = x1Idx;
            S.bestmdl = mdl;
        end
    end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

Notice that function f613() has the additional input parameter x1Idx to select the independent variable that appears in the numerator part.

And here is the MATLAB code for the test program go66().

```

clc
clear all

zBest = 0;

filename = "Go66.xlsx";
xyData = readmatrix(filename,'Sheet','Data');
ydata =xyData(:,1);
xdata = xyData(:,2:end);

xyTransf = readmatrix(filename,'Sheet','T633');
[R633,S633] = fx633(ydata,xdata,xyTransf)
S633.bestmdl

```

```

if R633 > zBest
    zBest = R633;
    bestS = S633;
end

xyTransf = readmatrix(filename,'Sheet','T632');
[R6321,S6321] = fx632(ydata,xdata,1,2,xyTransf)
S6321.bestmdl
if R6321 > zBest
    zBest = R6321;
    bestS = S6321;
end

xyTransf = readmatrix(filename,'Sheet','T632');
[R6322,S6322] = fx632(ydata,xdata,1,3,xyTransf)
S6322.bestmdl
if R6322 > zBest
    zBest = R6322;
    bestS = S6322;
end

xyTransf = readmatrix(filename,'Sheet','T632');
[R6323,S6323] = fx632(ydata,xdata,2,3,xyTransf)
S6323.bestmdl
if R6323 > zBest
    zBest = R6323;
    bestS = S6323;
end

xyTransf = readmatrix(filename,'Sheet','T631');
[R6311,S6311] = fx631(ydata,xdata,1,xyTransf)
S6311.bestmdl
if R6311 > zBest
    zBest = R6311;
    bestS = S6311;
end

xyTransf = readmatrix(filename,'Sheet','T631');
[R6312,S6312] = fx631(ydata,xdata,2,xyTransf)
S6312.bestmdl
if R6312 > zBest
    zBest = R6312;
    bestS = S6312;
end

xyTransf = readmatrix(filename,'Sheet','T631');
[R6313,S6313] = fx631(ydata,xdata,3,xyTransf)
S6313.bestmdl
if R6313 > zBest
    zBest = R6313;
    bestS = S6313;
end

xyTransf = readmatrix(filename,'Sheet','T613');
[R6131,S6131] = fx613(ydata,xdata,1,xyTransf)
S6131.bestmdl

```

```

if R6131 > zBest
    zBest = R6131;
    bestS = S6131;
end

xyTransf = readmatrix(filename, 'Sheet', 'T613');
[R6132, S6132] = fx613(ydata, xdata, 2, xyTransf)
S6132.bestmdl
if R6132 > zBest
    zBest = R6132;
    bestS = S6132;
end

xyTransf = readmatrix(filename, 'Sheet', 'T613');
[R6133, S6133] = fx613(ydata, xdata, 3, xyTransf)
S6133.bestmdl
if R6133 > zBest
    zBest = R6133;
    bestS = S6133;
end

fprintf("\n===== \n\n")
fprintf("----- Best Model ----- \n\n");
zBest
bestS
bestS.bestmdl

```

Here is the console output:

```

R633 =

    1

S633 =

    struct with fields:

        model: "(X1+X2+X3) / (X1+X2+X3)"
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestX3pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestYX3pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6

```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	5	0	Inf	0
x1	1	0	Inf	0
x2	1	0	Inf	0
x3	1	0	Inf	0
x4	1	0	Inf	0
x5	1	0	Inf	0
x6	1	0	Inf	0

Number of observations: 15, Error degrees of freedom: 8

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 2.26e+28, p-value = 1.81e-112

R6321 =

0.9999

S6321 =

struct with fields:

```

    model: "(X1+X2+X3) / (X1+X2) "
    bestAdjR2: 0.9999
    bestYpwr: 3
    bestX1pwr: 1
    bestX2pwr: 1
    bestX3pwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 1
    bestx1Idx: 1
    bestx2Idx: 2
    bestmdl: [1×1 LinearModel]
```

ans =

Linear regression model:

y ~ 1 + x1 + x2 + x3 + x4 + x5

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	-14.711	16.067	-0.91562	0.38374
x1	3.9431	0.93859	4.2011	0.0023028
x2	2.9001	0.68983	4.2042	0.0022925
x3	0.084671	0.0066933	12.65	4.9064e-07
x4	0.059569	0.0099111	6.0104	0.00019994
x5	-0.20334	0.0053959	-37.684	3.2379e-11


```

Number of observations: 15, Error degrees of freedom: 9
Root Mean Squared Error: 4.75
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 3.23e+04, p-value = 7.46e-19

```

```
R6322 =
```

```
0.9999
```

```
S6322 =
```

```
struct with fields:
```

```

    model: "(X1+X2+X3)/(X1+X3)"
    bestAdjR2: 0.9999
    bestYpwr: 3
    bestX1pwr: 3
    bestX2pwr: 3
    bestX3pwr: 0
    bestYX1pwr: 0
    bestYX2pwr: 1
    bestx1Idx: 1
    bestx2Idx: 3
    bestmdl: [1×1 LinearModel]

```

```
ans =
```

```
Linear regression model:
```

```
y ~ 1 + x1 + x2 + x3 + x4 + x5
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	-89.484	25.887	-3.4568	0.0071975
x1	0.0065619	0.002032	3.2293	0.010334
x2	0.0012214	3.1873e-05	38.32	2.7871e-11
x3	51.753	8.547	6.0551	0.00018931
x4	0.032369	0.0035895	9.0179	8.4005e-06
x5	-0.21604	0.0018603	-116.13	1.3217e-15

```

Number of observations: 15, Error degrees of freedom: 9
Root Mean Squared Error: 4.92
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 3.01e+04, p-value = 1.02e-18

```

```
R6323 =
```

```
0.9999
```

```
S6323 =
```

```

struct with fields:

    model: "(X1+X2+X3)/(X2+X3)"
    bestAdjR2: 0.9999
    bestYpwr: 3
    bestX1pwr: 3
    bestX2pwr: 2
    bestX3pwr: 3
    bestYX1pwr: 0
    bestYX2pwr: 3
    bestx1Idx: 2
    bestx2Idx: 3
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
y ~ 1 + x1 + x2 + x3 + x4 + x5

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	39.12	6.2722	6.2371	0.00015196
x1	0.0026335	0.0019438	1.3548	0.2085
x2	-0.06473	0.0080867	-8.0045	2.2035e-05
x3	0.0014719	0.00014736	9.9884	3.6128e-06
x4	-0.29323	0.0089158	-32.888	1.0934e-10
x5	7.9777e-05	1.2764e-05	6.2502	0.00014959

```

Number of observations: 15, Error degrees of freedom: 9
Root Mean Squared Error: 5.77
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 2.19e+04, p-value = 4.29e-18

R6311 =

    0.9987

S6311 =

struct with fields:

    model: "(X1+X2+X3)/(X1)"
    bestAdjR2: 0.9987
    bestYpwr: 1
    bestX1pwr: 3
    bestX2pwr: 1
    bestX3pwr: 1
    bestYX1pwr: 0
    bestx1Idx: 1
    bestmdl: [1×1 LinearModel]

```

```
ans =
```

```
Linear regression model:
```

```
y ~ 1 + x1 + x2 + x3 + x4
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	3.2923	0.082565	39.875	2.3528e-12
x1	0.00023994	3.2446e-05	7.395	2.3287e-05
x2	0.090457	0.0014997	60.317	3.8126e-14
x3	0.084145	0.0020991	40.087	2.2322e-12
x4	0.080304	0.0054191	14.819	3.929e-08

```
Number of observations: 15, Error degrees of freedom: 10
```

```
Root Mean Squared Error: 0.0862
```

```
R-squared: 0.999, Adjusted R-Squared: 0.999
```

```
F-statistic vs. constant model: 2.71e+03, p-value = 3.99e-15
```

```
R6312 =
```

```
0.9997
```

```
S6312 =
```

```
struct with fields:
```

```
    model: "(X1+X2+X3)/(X2)"
    bestAdjR2: 0.9997
    bestYpwr: 3
    bestX1pwr: 0
    bestX2pwr: 2
    bestX3pwr: 2
    bestYX1pwr: 0
    bestx1Idx: 2
    bestmdl: [1x1 LinearModel]
```

```
ans =
```

```
Linear regression model:
```

```
y ~ 1 + x1 + x2 + x3 + x4
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	82.245	14.346	5.7328	0.00018957
x1	-12.757	5.6833	-2.2447	0.048616
x2	-0.0118	0.0058967	-2.0011	0.073258
x3	0.045072	0.0040616	11.097	6.0748e-07

```

x4          -0.22939    0.0052896    -43.366    1.0211e-12

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 9.07
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.11e+04, p-value = 3.52e-18

R6313 =

    0.9997

S6313 =

struct with fields:

    model: "(X1+X2+X3) / (X3) "
    bestAdjR2: 0.9997
    bestYpwr: 3
    bestX1pwr: 0
    bestX2pwr: 3
    bestX3pwr: 3
    bestYX1pwr: 0
    bestx1Idx: 3
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	104.13	14.497	7.1829	2.9861e-05
x1	-19.765	5.4713	-3.6125	0.0047485
x2	0.00096514	5.756e-05	16.768	1.1935e-08
x3	-0.00011831	7.0318e-05	-1.6825	0.12339
x4	-0.22409	0.0039511	-56.715	7.0466e-14

```

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 9.41
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.03e+04, p-value = 5.07e-18

R6131 =

    0.9995

S6131 =

```

```

struct with fields:

    model: "(X1)/(X1+X2+X3)"
    bestAdjR2: 0.9995
    bestYpwr: 3
    bestYX1pwr: 0
    bestYX2pwr: 3
    bestYX3pwr: 0
    bestX1pwr: 0
    bestxIdx: 1
    bestmdl: [1×1 LinearModel]

ans =

Linear regression model:
y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	129.59	24.795	5.2264	0.00038627
x1	-31.119	10.977	-2.8351	0.017698
x2	-0.033792	0.0080667	-4.189	0.0018608
x3	-8.6915e-07	4.3317e-08	-20.065	2.079e-09
x4	-0.20507	0.0043591	-47.044	4.5399e-13

```

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 11.5
R-squared: 1, Adjusted R-Squared: 0.999
F-statistic vs. constant model: 6.87e+03, p-value = 3.83e-17

R6132 =

    0.9998

S6132 =

struct with fields:

    model: "(X2)/(X1+X2+X3)"
    bestAdjR2: 0.9998
    bestYpwr: 3
    bestYX1pwr: 0
    bestYX2pwr: 1
    bestYX3pwr: 0
    bestX1pwr: 2
    bestxIdx: 2
    bestmdl: [1×1 LinearModel]

ans =

```

Linear regression model:

$$y \sim 1 + x1 + x2 + x3 + x4$$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	50.381	4.0388	12.474	2.0271e-07
x1	0.11959	0.011761	10.169	1.3632e-06
x2	0.048799	0.0065834	7.4125	2.2819e-05
x3	0.002918	0.00083351	3.5009	0.0057183
x4	-0.24233	0.0068636	-35.307	7.8814e-12

Number of observations: 15, Error degrees of freedom: 10

Root Mean Squared Error: 7.32

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 1.7e+04, p-value = 4.12e-19

R6133 =

0.9998

S6133 =

struct with fields:

```

    model: "(X3)/(X1+X2+X3)"
    bestAdjR2: 0.9998
    bestYpwr: 3
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestYX3pwr: 2
    bestX1pwr: 2
    bestxIdx: 3
    bestmdl: [1×1 LinearModel]
```

ans =

Linear regression model:

$$y \sim 1 + x1 + x2 + x3 + x4$$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	51.339	4.8428	10.601	9.2872e-07
x1	0.064273	0.0078443	8.1936	9.5443e-06
x2	0.020099	0.0043634	4.6063	0.00097078
x3	-0.22652	0.0019246	-117.69	4.8093e-17
x4	6.8965e-06	6.5928e-06	1.0461	0.32016

```

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 7.56
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.59e+04, p-value = 5.71e-19

```

```
=====
```

```
----- Best Model -----
```

```
zBest =
```

```
1
```

```
bestS =
```

```
struct with fields:
```

```

    model: "(X1+X2+X3)/(X1+X2+X3)"
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestX3pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestYX3pwr: 0
    bestmdl: [1×1 LinearModel]

```

```
ans =
```

```
Linear regression model:
```

```
y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6
```

```
Estimated Coefficients:
```

	Estimate	SE	tStat	pValue
(Intercept)	5	0	Inf	0
x1	1	0	Inf	0
x2	1	0	Inf	0
x3	1	0	Inf	0
x4	1	0	Inf	0
x5	1	0	Inf	0
x6	1	0	Inf	0

```

Number of observations: 15, Error degrees of freedom: 8
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 2.26e+28, p-value = 1.81e-112

```

The console output shows the regression table including the regression coefficients, their standard errors, the student-t values, and the p-Values. The latter

values can shed valuable insight as to the eligibility of a variable/term to be part of the model. The best model is:

$$Y = (5 + X_1 + X_2 + X_3) / (1 + \ln(X_1) + \ln(X_2) + \ln(X_3))$$

About the ZIP File for this study

You will find the files for the MATLAB code and Excel files in the ZIP file for this project. The ZIP file het.zip contains the PDF version of the document, and the files used in this study.