In this article I present two versions of VBA subroutines that select the best polynomial regression models among a larger set of models. In both cases, the subroutines obtain the values of observed variables, apply a number of transformations on each variable, and then use these transformed values to determine which combination of transformations yield the best set of fitted polynomials. In other words, the VBA subroutines go beyond performing simple/typical polynomial regression. This approach allows you to investigate polynomial relations between the transformations of the observed variables! While we are certainly interested in the very best polynomial model, it is a very wise to view the best say 20 models. The reason is that if the user has several sets of similar data (each having its own errors) then a possible best empirical polynomial model will consistently appear in the set of top models. Once you select one or more polynomial models, you can use subroutine MLR, which I present in the first article, to obtain the regression ANOVA table for these selected polynomial models.

This article presents two approaches for obtaining the best (empirical) polynomial models:

- Scheme P1: Applying a combination of powers and the natural logarithm to the observed variables. These transformed variables then participate in the polynomial regression for a specific polynomial order.
- Scheme P2: Is a version of scheme P1 that covers a range of orders.

**The Scheme P1 Approach for Best Polynomial Models**

The best way to present and explain scheme P1 is to show you the worksheet for this approach to getting the best polynomial models. Figures 1 and 2 show the left and right sides, respectively, of a sample scheme P1 worksheet. Notice the following individual cells and columns of cells that provide input for the regression program:

- Cell A2 specifies the polynomial order.
- Cell A4 contains the values for the maximum number of best polynomial models. The current value is 20.
- Cells A6 and A8 specify the shift and scale values for the observed independent variable X. The default settings for the shift and scale values are 0 and 1, respectively.
- Cells A10 and A12 specify the shift and scale values for the observed dependent variable Y. The default settings for the shift and scale values are 0 and 1, respectively.
- Column B has the values of the observed dependent variable Y, starting at cells B2.
- Column C stores the values of the observed independent variable X, starting at cell C2.
- Column D MUST BE empty.
- Column E contains the numerically coded transformations for variable Y. The numerical codes for the transformations are the same ones used in the best multiple regression programs.
- Column F contains the numerically coded transformations for variable X.
- Column G MUST BE empty.

The following worksheet columns provide the regression output generated by running the VBA macro **BestPolyReg**:

- Column H shows the best transformations for variable Y.
- Column I shows the best transformations for variable X.
- Column J displays the values of the F statistic.
- Column K shows the values of the \( R^2 \) statistic.
- Column L displays the values of the intercepts.
- Column M shows the values for regression coefficient for the transformed variable raised to power 1.
- Column N and up displays the values for regression coefficient for the transformed variable raised to power 2 and up, respectively.

The rows that start with the column H represent the record for each model. These rows are sorted in a descending order using the values of the F statistic (in column J).

![Figure 1. The left side of worksheet for scheme P1.](image-url)
Figure 2. The right side of worksheet for scheme P1.

Looking at columns E and F in Figure 1, you see the list of transformations for each variable. The numerically coded transformations represent powers used to raise the variables in order to calculate their transformations. For example, the value of –4 in cell E2 tells the program that one of the transformations for variable Y is 1/Y^4. You will no doubt notice that columns E and F contain the value of 0. The VBA code treats 0 as a special case and applies the natural logarithm to the corresponding variable.

The transformations of the observed variables require that you obey the following simple rules:

- Each variable has its own set of transformations that are independent, in number and sequence, of those used with the other variable.
- The transformation can be integers and non-integers, as well as negative, zero, or positive. The VBA subroutine has an error handler and will catch runtime errors. Any set of transformation that generates a runtime error is out of the contest, so to speak!

Looking at the sample results in Figure 2, the best model is:

Y = 100 + 2 X + 3 X^2 + 0.1 X^3

This model has the F statistic of 3.09E+32 and the R^2 value of 1.

The BestPolyReg VBA Subroutine

Listing 1 shows the source code for subroutine BestPolyReg. This subroutine performs the following general tasks:

- Read the input from the leading columns. This includes the observations for the different variables, the scale and shift values for X and Y, and the lists of numerically coded transformations.
- Start nested loops to transform the X and Y variables. The nested loops perform the nest tasks.
- Prepare the regression variables for the polynomial regression.
- Invoke the Linest function and store its results in variable vRegResultsMat. The subroutine compares the value of the F statistic with the lowest value of F in the Hit Parade list. If the newly calculated F statistic is greater than that lowest F value, the subroutine overwrites the last Hit Parade row with newly calculated data and then invokes Excel’s Sort method to quickly re-sort the Hit Parade list using values of the F statistics.

Option Explicit
Option Base 1

Sub BestPolyReg()

Const MAX_ERRORS As Double = 1000000# ' initial max error value

Dim ErrorCounter As Double, MaxErrors As Double, sMaxErr As String
Dim NumIndpVars As Integer ' number of variables
Dim PolyOrder As Integer ' polynomial order
Dim TN As Integer ' total number coefficients
Dim N As Integer ' number of data points
Dim MaxTrans As Integer ' max transformations
Dim MaxResults As Integer ' max results
Dim Col1 As Integer, Col2 As Integer, Col3 As Integer
Dim Col4 As Integer, Col5 As Integer
Dim I As Integer, J As Integer, K As Integer
Dim M1 As Integer, M As Integer
Dim iY As Integer, iX As Integer
Dim TransfMat() As Double
Dim CurrentTransf() As Double, CountTransf() As Integer
Dim NumTransf() As Integer ' number of transformations
Dim Y() As Variant, X() As Variant
Dim Yt() As Variant, Xt() As Variant
Dim Xtp() As Variant
Dim vRegResultsMat As Variant
Dim F As Double, Rsqr As Double, xval As Double
Dim ShiftX As Double, ShiftY As Double
Dim ScaleX As Double, ScaleY As Double

ErrorCounter = 0
MaxErrors = MAX_ERRORS
NumIndpVars = 2
PolyOrder = [A2].Value
MaxResults = [A4].Value
TN = PolyOrder + 1
ShiftX = [A6].Value
ScaleX = [A8].Value
ShiftY = [A10].Value
ScaleY = [A12].Value
Col1 = 2  ' first column of data
Col2 = 5  ' first column of transformations
Col3 = 8  ' first column of best transformations
Col4 = 10 ' first column of results
Col5 = Col4 + TN + 1 ' last column of results

Range(Cells(1 + Col3), Cells(1, 50)).Value = ""
Cells(1, Col3) = "Transf Y"
Cells(1, Col3 + 1) = "Transf X"
Cells(1, Col4) = "F"
Cells(1, Col4 + 1) = "Rsqr"
For I = 0 To PolyOrder
   Cells(1, Col4 + 2 + I) = "A" & I
Next I

Range(Cells(2, Col3), Cells(1 + 2 * MaxResults, Col4 + 3 * 
TN)).Value = ""
Range(Cells(2, Col4), Cells(1 + MaxResults, Col4 + 1 + TN)).Value = 
0
MaxTrans = Range(Cells(2, Col2), Cells(1, 
Col2)).CurrentRegion.Rows.Count - 1
ReDim NumTransf(NumIndpVars), TransfMat(NumIndpVars, MaxTrans), 
CurrentTransf(NumIndpVars)
ReDim CountTransf(NumIndpVars)

For I = 1 To NumIndpVars
   J = 2
   Do While Trim(Cells(J, Col2 + I - 1)) <> ""
      TransfMat(I, J - 1) = Cells(J, Col2 + I - 1)
      J = J + 1
   Loop
   NumTransf(I) = J - 2
Next I

N = Range("B1").CurrentRegion.Rows.Count - 1
Y = Range("B2:B" & N + 1).Value
X = Range(Cells(2, 3), Cells(N + 1, 2 + PolyOrder)).Value
Yt = Range("B2:B" & N + 1).Value
Xt = Range(Cells(2, 3), Cells(N + 1, 2 + PolyOrder)).Value
ReDim Xtp(N, PolyOrder) ' polynomial data matrix

For iY = 1 To NumTransf(1)
    CurrentTransf(1) = TransfMat(1, iY)
    For iX = 1 To NumTransf(2)
        CurrentTransf(2) = TransfMat(2, iX)
    Next iX
On Error GoTo HandleErr

For I = 1 To N
    DoEvents
    If CurrentTransf(1) <> 0 Then
        Yt(I, 1) = (ScaleY * Y(I, 1) + ShiftY) ^ CurrentTransf(1)
    Else
        Yt(I, 1) = Log(ScaleY * Y(I, 1) + ShiftY)
    End If
    If CurrentTransf(2) <> 0 Then
        Xt(I, 1) = (ScaleX * X(I, 1) + ShiftX) ^ CurrentTransf(2)
    Else
        Xt(I, 1) = Log(ScaleX * X(I, 1) + ShiftX)
    End If
    For J = 1 To PolyOrder
        Xtp(I, J) = Xt(I, 1) ^ J
    Next J
Next I

' perform the regression calculations
vRegResultsMat = Application.WorksheetFunction.LinEst(Yt, Xtp, True, True)

Rsqr = vRegResultsMat(3, 1)
F = vRegResultsMat(4, 1)
' check if F > F of last result
If F > Cells(MaxResults + 1, Col4) Then
    M1 = MaxResults + 1
    ' write new results to row M
    Cells(M1, Col4) = F
    Cells(M1, Col4 + 1) = Rsqr
    For I = 1 To TN
        Cells(M1, Col4 + I + 1) = vRegResultsMat(1, TN - I + 1)
    Next I
For I = 1 To NumIndpVars
    Cells(M1, Col3 + I - 1) = CurrentTransf(I)
Next I

Range(Cells(2, Col3), Cells(MaxResults + 1, Col5)).Select
Range(Cells(2, Col3), Cells(MaxResults + 1, Col5)).Sort
Key1:=Range(Cells(2, Col4), Cells(MaxResults + 1, Col4)),
Order1:=xlDescending

End If ' If F > Cells(MaxResults + 1, Col3)

GoTo Here

HandleErr:
    ErrorCounter = ErrorCounter + 1
    If ErrorCounter > MaxErrors Then
        If MsgBox("Reached maximum error limits of " & ErrorCounter & vbCrLf & _
        "Want to stop the process?", vbYesNo + vbQuestion, "Confirmation requested") = vbYes Then
            Exit Sub
        Else
            sMaxErr = InputBox("Update maximum number of errors", "Max Errors Input", MaxErrors)
            If Trim(sMaxErr) = "" Then
                MsgBox "User canceled calculations process", vbOKOnly + vbInformation, "End of Process"
                Exit Sub
            End If
            MaxErrors = CDbl(sMaxErr)
            ErrorCounter = 0
        End If
    End If
Next iX
Next iY

MsgBox "Done", vbOKOnly + vbInformation, "Success!"
End Sub

Listing 1. The BestPolyReg subroutine.
Handling Runtime Error in Scheme P1, and P2

The subroutines **BestPolyReg** and **BestPolRegSet** (which I will introduce later in this article) have an error handler and an error counting scheme. The constant **MAX_ERRORS** represents the initial value for the maximum number of occurring runtime errors. The subroutine assigns the value of this constant to the variable **MaxErrs**. The subroutine also uses the variable **ErrorCounter** to count the number of occurring runtime errors. When the value in variable **ErrorCounter** exceeds that in variable **MaxErrs**, the subroutine displays a message box telling you that the number of occurring runtime errors has reached its maximum limit. This message should not convey a mandatory end to the calculations. The message box has a **Yes** and **No** buttons allowing you to end the program by clicking the **Yes** button. If you click the **No** button, the subroutine displays an input box prompting you to enter a new maximum error limit. The input box shows the current maximum error limit as the default value. You can alter the maximum error limit or simply press the **OK** button to accept it and proceed with the statistical calculations. In this case, the subroutine resets the value in **ErrorCounter** to 0. You can also stop the subroutine altogether by clicking the **Cancel** button. Thus, the subroutine offers you two points of exit.

The Scheme P2 Approach for Best Polynomial Models

Scheme P2 allows you to find the best polynomial order (in a range of orders that you specify) and the best power/logarithm transformations for each polynomial. Unlike, scheme P1, scheme P2 works on multiple worksheets. Therefore you will need a separate workbook for each set of data when using scheme P2. The first sheet in the workbook specifies the polynomial order, the maximum number of results for that worksheet, the source data, and transformation lists. All subsequent sheets need only to specify the polynomial order and the maximum number of results for that worksheet. The VBA subroutine copies the source data and transformation lists from the first worksheet to all other worksheets.

Figures 3 and 4 show the left and right sides, respectively, of a sample scheme P2 worksheet. This is the first sheet that also includes the source data, and transformation lists. Notice the following individual cells and columns of cells that provide input for the regression program:

- Cell A2 contains the polynomial order.
- Cell A4 contains the values for the maximum number of best models. The current value is 20.
- Column B has the values of the observed dependent variable Y, starting at cell B2.
- Column C stores the values of the observed independent variable X, starting at cell C2.
- **Column D MUST BE empty.**
- Column E contained the numerically coded transformations for variable Y.
- Column F contained the numerically coded transformations for variable X.
- **Column G MUST BE empty.**
The following worksheet columns provide the regression output generated by running the VBA macro FindBestPolyReg. You need to prepare the headers manually:

- Column H shows the best transformations for variable Y.
- Column I shows the best transformations for variable X.
- Column J displays the values of the F statistic.
- Column K shows the values of the $R^2$ statistic
- Column L displays the values of the intercepts.
- Column M shows the values for regression coefficient for the transformed variable raised to power 1.
- Columns N and up show values for regression coefficient for the transformed variable raised to power 2 and up, respectively.

The rows that start with the column H represent the record for each model. These rows are sorted in a descending order using the values of the F statistic (in column J).

Figure 3. The left side of worksheet for scheme P2.
The data in Figure 3 is based on the following polynomial:

\[ Y = 100 + 2X + 3X^2 + 0.1X^3 \]

The workbook has four worksheets to perform polynomial regression in the orders 1 through 4. When you run the `FindBestPolyReg` subroutine, it selects worksheet Set 3 which has the cubic polynomial. Figures 5 and 6 show the best results of running the `FindBestPolyReg` subroutine.
The best fitted polynomial has the F statistic of 1.238E+31 and the $R^2$ value of 1. The calculations identify the best polynomial as:

$$Y = 100 + 2X + 3X^2 + 0.1X^3$$

### The FindBestPolyReg VBA Subroutine

Listing 2 shows the source code for subroutine **FindBestPolyReg**. This subroutine performs the following general tasks:

- Initializes the index of the best worksheet and F statistics for the best polynomial.
- Loops over the worksheets in the workbook.
  - Select the next worksheet.
    - If the current worksheet is not the first one, copy the data in columns B to columns K from the first worksheet to the current worksheet.
    - Invoke subroutine **BestPolyReg** to find the best transformations for the currently selected polynomial order. This code for the accompanying **BestPolyReg** subroutine is the same one used in scheme P1.
    - Store the best F and the index of the current worksheet if it provides the best F statistic.
- Select the worksheet that has the best polynomial model.
- Display a message box that states the order of the best polynomial model (see Figure 5 for a sample message box).

```vba
Option Explicit
Option Base 1
```
Sub FindBestPolyReg()
    Dim I As Integer, N As Integer
    Dim BestF As Double, BestOrder As Integer, BestSheet As Integer

    BestF = 0
    BestOrder = 0
    N = Application.Worksheets.Count
    For I = 1 To N
        Sheets(I).Select
        If I > 1 Then
            ' copy (X,Y) data and transformations from first sheet
            Sheets(1).Range("B1:K32000").Copy Range("B1:K32000")
        End If
        Call BestPolyReg
        If BestF < Cells(2, 10) Then
            BestF = Cells(2, 10)
            BestOrder = Cells(2, 1)
            BestSheet = I
        End If
    Next I
    Sheets(BestSheet).Select
    MsgBox "Best Polynomial Order is " & BestOrder, vbOKOnly + vbInformation, "Success!"
End Sub

Sub BestPolyReg()

    Const MAX_ERRORS As Double = 1000000# ' initial max error value

    Dim ErrorCounter As Double, MaxErrors As Double, sMaxErr As String
    Dim NumIndpVars As Integer ' number of variables
    Dim PolyOrder As Integer ' polynomial order
    Dim TN As Integer ' total number of coefficients
    Dim N As Integer ' number of data points
    Dim MaxTrans As Integer ' max transformations
    Dim MaxResults As Integer ' max results
    Dim Col1 As Integer, Col2 As Integer, Col3 As Integer
    Dim Col4 As Integer, Col5 As Integer
    Dim I As Integer, J As Integer, K As Integer
    Dim M1 As Integer, M As Integer
    Dim iY As Integer, iX As Integer
    Dim TransfMat() As Double
    Dim CurrentTransf() As Double, CountTransf() As Integer
    Dim NumTransf() As Integer ' number of transformations
    Dim Y() As Variant, X() As Variant
Dim Yt() As Variant, Xt() As Variant
Dim Xtp() As Variant
Dim vRegResultsMat As Variant
Dim F As Double, Rsqr As Double, xval As Double

ErrorCounter = 0
MaxErrors = MAX_ERRORS
NumIndpVars = 2
PolyOrder = [A2].Value
MaxResults = [A4].Value
TN = PolyOrder + 1
Col1 = 2 ' first column of data
Col2 = 5 ' first column of transformations
Col3 = 8 ' first column of best transformations
Col4 = 10 ' first column of results
Col5 = Col4 + TN + 1 ' last column of results

Range(Cells(1 + Col3), Cells(1, 50)).Value = ""
Cells(1, Col3) = "Transf Y"
Cells(1, Col3 + 1) = "Transf X"
Cells(1, Col4) = "F"
Cells(1, Col4 + 1) = "Rsqr"
For I = 0 To PolyOrder
    Cells(1, Col4 + 2 + I) = "A" & I
Next I

Range(Cells(2, Col3), Cells(1 + 2 * MaxResults, Col4 + 3 * TN)).Value = ""
Range(Cells(2, Col4), Cells(1 + MaxResults, Col4 + 1 + TN)).Value = 0
MaxTrans = Range(Cells(2, Col2), Cells(1, Col2)).CurrentRegion.Rows.Count - 1
ReDim NumTransf(NumIndpVars), TransfMat(NumIndpVars, MaxTrans), CurrentTransf(NumIndpVars),
ReDim CountTransf(NumIndpVars)

For I = 1 To NumIndpVars
    J = 2
    Do While Trim(Cells(J, Col2 + I - 1)) <> ""
        TransfMat(I, J - 1) = Cells(J, Col2 + I - 1)
        J = J + 1
    Loop
    NumTransf(I) = J - 2
Next I
N = Range("B1").CurrentRegion.Rows.Count - 1
Y = Range("B2:B" & N + 1).Value
X = Range(Cells(2, 3), Cells(N + 1, 2 + PolyOrder)).Value
Yt = Range("B2:B" & N + 1).Value
Xt = Range(Cells(2, 3), Cells(N + 1, 2 + PolyOrder)).Value
ReDim Xtp(N, PolyOrder) ' polynomial data matrix

For iY = 1 To NumTransf(1)
    CurrentTransf(1) = TransfMat(1, iY)
    For iX = 1 To NumTransf(2)
        CurrentTransf(2) = TransfMat(2, iX)
    On Error GoTo HandleErr
    For I = 1 To N
        DoEvents
        If CurrentTransf(1) <> 0 Then
            Yt(I, 1) = Y(I, 1) ^ CurrentTransf(1)
        Else
            Yt(I, 1) = Log(Y(I, 1))
        End If
        If CurrentTransf(2) <> 0 Then
            Xt(I, 1) = X(I, 1) ^ CurrentTransf(2)
        Else
            Xt(I, 1) = Log(X(I, 1))
        End If
        For J = 1 To PolyOrder
            Xtp(I, J) = Xt(I, 1) ^ J
        Next J
    Next I
    ' perform the regression calculations
    vRegResultsMat = Application.WorksheetFunction.LinEst(Yt, Xtp, True, True)
    Rsqr = vRegResultsMat(3, 1)
    F = vRegResultsMat(4, 1)
    ' check if F > F of last result
    If F > Cells(MaxResults + 1, Col4) Then
        M1 = MaxResults + 1
        ' write new results to row M
Cells(M1, Col4) = F
Cells(M1, Col4 + 1) = Rsqr
For I = 1 To TN
    Cells(M1, Col4 + I + 1) = vRegResultsMat(1, TN - I + 1)
Next I
For I = 1 To NumIndpVars
    Cells(M1, Col3 + I - 1) = CurrentTransf(I)
Next I

Range(Cells(2, Col3), Cells(MaxResults + 1, Col5)).Select
Range(Cells(2, Col3), Cells(MaxResults + 1, Col5)).Sort
Key1:=Range(Cells(2, Col4), Cells(MaxResults + 1, Col4)),
Order1:=xlDescending

End If ' If F > Cells(MaxResults + 1, Col3)

GoTo Here

HandleErr:
    ErrorCounter = ErrorCounter + 1
    If ErrorCounter > MaxErrors Then
        If MsgBox("Reached maximum error limits of " & ErrorCounter & vbCrLf & "Want to stop the process?", vbYesNo + vbQuestion, "Confirmation requested") = vbYes Then
            Exit Sub
        Else
            sMaxErr = InputBox("Update maximum number of errors", "Max Errors Input", MaxErrors)
            If Trim(sMaxErr) = "" Then
                MsgBox "User canceled calculations process", vbOKOnly + vbInformation, "End of Process"
                Exit Sub
            End If
            MaxErrors = CDb1(sMaxErr)
            ErrorCounter = 0
        End If
    End If
    Resume Here
Here:
    Next iX
    Next iY
End Sub

Listing 2. The FindBestPolyReg subroutine.
References

3. Wikipedia article *Coefficient of Determination*.
4. Wikipedia article *Linear Regression*.
5. Wikipedia article *Simple Linear Regression*.
6. Wikipedia article *Akaike information criterion*.

About this Article

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0</td>
<td>12/7/2012</td>
<td>Initial release.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>