Multiple Regression Using Excel Linest Function

By Namir C. Shammas

This is the first article in a series of articles that discuss using the Excel **Linest** function, in VBA subroutines, to support various kinds of multiple and polynomial regression calculations. This article discusses the following topics:

- Basic linearized regression approach.
- The results of the **Linest** function.
- Using the **Linest** function and variant variables to perform customized multiple regression.
- Creating and display the regression ANOVA table.

Basic Linearized Regression Approach

When you perform regression calculations you have the following ingredients:

- A set of observed values for variables. These variables include the dependent variable and at least one independent variable. Multiple independent variables are very common in multiple regression.
- One or more regression models (i.e. equations) that describes the relationship between the dependent variable and the independent variables.
- Transformed regression variables that are calculated based on the observed variables and the given regression model(s). The calculation of the regression coefficients of the model are based on the transformed variables.

In the case of simple linear models, the observed variables are also the regression variables. By contrast, in the case of linearized models, the regression variables have their values calculated based on the observed variables. Depending on the regression model, the regression variables can be straightforward transformation of corresponding observed variables (i.e. one-to-one relation). More complex regression model create regression variables based on the mathematical expressions that involve multiple observed variables.

Here are examples of simple multiple regression models:

$Y = a_0 + a_1 X_1 + a_2 X_2$

$Y = a_0 + a_1 X_1 + a_2 X_2 + a_3 X_3$

The variables Y, X_1, X_2, and X_3 are the observed variables and are also the regression variables.

Here are examples of regression models where some of the variables are linearized. The linearization uses a simple one-to-one transformation:

$1/Y = a_0 + a_1\ 1/X_1 + a_2\ 1/X_2$

$Y = a_0 + a_1 \ln(X_1) + a_2\ 1/X_2 + a_3 \sqrt{X_3}$

$1/Y = a_0 + a_1\ 1/X_1 + a_2\ 1/X_2\text{^}2 + a_3\ X_3\text{^}3$

Notice that the models are linearized by applying transformations to the various observed variables. For example, the first model has the observed variables Y, X_1 and X_2. The model has the regression variables of 1/Y, $1/X_1$, and $1/X_2$. The above examples show simple linearized terms.

Here are examples for more elaborate linearized regression models:

$Y = a_0 + a_1\ 1/(2\ X_1\ X_2 + 1) + a_2\ 1/(X_1{*}X_2)$

$Y = a_0 + a_1 \ln(1 + X_1 + X_2) + a_2 \ln(X_2)/(3X_2 + 1) + a_3 \sqrt{(X_1\text{^}2 + X_3\text{^}2)}$

$1/Y = a_0 + a_1\ 1/(X_1\text{^}2 + 1) + a_2\ X_1/X_2\text{^}2 + a_3\ X_1\ X_2\ X_3\text{^}3$

$Y = a_0 + a_1 X + a_2 X^2 + a_3 X^3$

The above sample regression models show several terms that are expressions containing one or more variables. The number of regression variables may or may not equal the number of observed variables. This equality (or inequality) is not an issue. For example, the first model has the observed variables Y, X_1 and X_2. The model has the regression variables of Y, $1/(2\ X_1\ X_2 + 1)$, and $1//(X_1{*}X_2)$. These regression models can still benefit from the function **Linest**, because they are linearized multiple regression models. The last model has the observed variables X and Y, and the regression variables Y, X, X^2, and X^3. This polynomial model is typical of polynomials that have a single independent variable.

It is very important to make a distinction between the observed variables (dependent and independent) and the regression variables. The two types of variables *can* be the same, but this equivalence serves to support only simple multiple regression models. The regression coefficients are based directly on the values of the regression variables.

The Results of the Linest Function

The **Linest** function is a powerful Excel function that returns a matrix of results. The arguments for the function **Linest** are data ranges. When calling the function in VBA you can pass the ranges as such or as variant-type variables. The function's results include the regression coefficients, the standard errors for the regression coefficients, and other statistics that allow you to obtain the regression ANOVA table. You can display the results of function **Linest** as an array of cells, or store its results in a variant-typed variable. You can then access various elements of that variable to obtain the different values returned by the function **Linest**.

The **Linest** function returns an irregular r-like shaped matrix of results. Table 1 shows an outline for these results. The VBA subroutine in this article creates and displays an ANOVA regression table using the information in Table 1 and also expanding on it to include confidence intervals and p-values for the regression coefficients.

Slope a_n	Slope a_{n-1}	Slope a_{n-2}	…	Intercept a_0
Standard error for a_n	Standard error for a_{n-1}	Standard error for a_{n-2}	…	Standard error for a_0
R^2	SE(y)			
F statistics	Degrees of freedom			
Regression SS	Residual SS			

Table 1. Outline for the results of function Linest.

The Multiple Regression Worksheet

Figure 1 shows a snapshot for a multiple regression worksheet. Notice the following individual cells and columns of cells that provide input for the regression program:

- Cell A2 contains the number of variables entering the regression. These regression variables can be the values of observed variables and/or their transformations.
- Cell A4 contains the confidence level, as a fraction.
- Cells A5 contains the transformation for the dependent variable Y. The cell stores an expression of Y used to transform the observed values of the dependent variable.
- Cells A6 and below contain expressions for the regression independent variables. These expressions can use the observed independent variables X1, X2, and so on, and even the dependent variable Y. An expression is not limited to using one variable. It can use multiple variables. Examples for expressions are:
    - 1/(X2+1)
    - X1^2
    - (2*X1^2+1)
    - X1*X2
    - LN(X1)/X3
    - LN(2*X1^2+5)/(X^2+1)
    - SQRT(X3^2+1)/(X1^2+1)
- Column B has the values of the observed dependent variable Y, starting in cell B2.
- Column C stores the values of the first observed independent variable, starting at cell C2.
- Columns D and beyond store the values of additional observed independent variables, starting at the columns' second rows.

- The column after comes the column of the last independent variable MUST BE empty. The output appears starting with the subsequent column.

The following worksheet rows provide the regression output generated by running the VBA macro **MLR**:

- The first row of the **Linest** output shows the regression coefficients, starting with the intercept and progressing up. This is in the reverse order that function **Linest** generates.
- The second row **Linest** output shows the standard error for the regression coefficients, starting with the standard error for the intercept and progressing upward.
- The third and fourth rows show the upper and lower confidence values for the regression coefficients.
- The fifth row shows the p-values for the regression coefficients.
- The sixth row shows the values of R^2 and SE(y).
- The seventh row displays the F statistic and the degrees of freedom.
- The eight row shows the regression SS and the residual SS.
- The ninth row displays the total SS and the inverse Student-t statistic.
- The tenth row shows the AICc value and the AICc baseline value (i.e. AICc minus the sum of error term).
- The eleventh row shows my version of the AICc. This version handles perfect fit in a manner that has the logarithm yield a zero.

I use the following equation to calculate the AICc statistic:

$$\text{AIC}_\text{C} = n \ln\left(\sum_1^n (\hat{y}_i - y_i)^2 / n\right) + 2k + (2\,k\,(k+1))/(n-k-1) \qquad (1)$$

I also use the following equation to calculate my own version of AICcs that handles perfect fits:

$$\text{AIC}_\text{CS} = n \ln\left(1 + \sum_1^n (\hat{y}_i - y_i)^2 / n\right) + 2k + (2\,k\,(k+1))/(n-k-1) \qquad (2)$$

The difference between equations 1 and 2 is that when a regression model has perfect fit, the logarithm in equation 1 becomes minus infinity. Any code that calculates AICc has to adjust the value of the sum of the squares (which is zero for perfect fit) to a very small value. This kind of adjustment for AICc gives it large negative values. By contrast, the values for AICcs will not be negative. When the summation term becomes small, the logarithm term approaches zero. The output in Figure 1 also shows an AICc baseline value. The program calculates this value using the following equation:

$$\text{Baseline AIC}_\text{C} = 2k + (2\,k\,(k+1))/(n-k-1) \qquad (3)$$

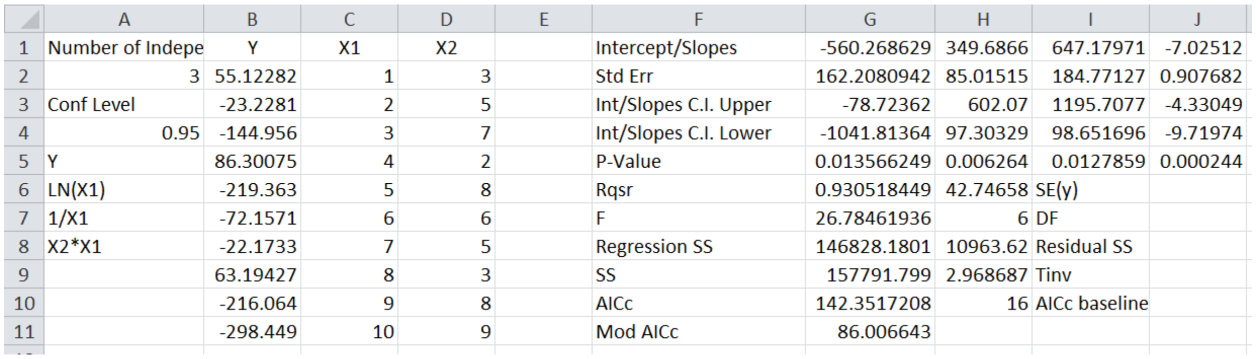
Figure 1 shows how the VBA code performs a multiple regression fit for the following model:

$Y = a_0 + a_1 \ln(X1) + a_2/X1 + a_3\, X1{*}X2$

The observed variables are Y, X1, and X2. The regression variables are Y, ln(X1), 1/X1, and X1*X2. The VBA code yields a regression model with the following regression coefficients (shown rounded to two decimal places):

$$Y = -551.61 + 347.73 \ln(X1) + 631.55/X1 - 7.06\, X1*X2$$

The model has F equal to 26.78 and R^2 equal to 0.9305.



	A	B	C	D	E	F	G	H	I	J
1	Number of Indepe	Y	X1	X2		Intercept/Slopes	-560.268629	349.6866	647.17971	-7.02512
2	3	55.12282	1	3		Std Err	162.2080942	85.01515	184.77127	0.907682
3	Conf Level	-23.2281	2	5		Int/Slopes C.I. Upper	-78.72362	602.07	1195.7077	-4.33049
4	0.95	-144.956	3	7		Int/Slopes C.I. Lower	-1041.81364	97.30329	98.651696	-9.71974
5	Y	86.30075	4	2		P-Value	0.013566249	0.006264	0.0127859	0.000244
6	LN(X1)	-219.363	5	8		Rqsr	0.930518449	42.74658	SE(y)	
7	1/X1	-72.1571	6	6		F	26.78461936	6	DF	
8	X2*X1	-22.1733	7	5		Regression SS	146828.1801	10963.62	Residual SS	
9		63.19427	8	3		SS	157791.799	2.968687	Tinv	
10		-216.064	9	8		AICc	142.3517208	16	AICc baseline	
11		-298.449	10	9		Mod AICc	86.006643			

Figure 1. The multiple regression worksheet.

The MLR VBA Subroutine

Listing 1 shows the source code for function **ExFx** and subroutine **MLR**. The function allows the subroutine to process expressions for the regression variables. As I mentioned earlier, these expression can use various observed variables. The function **ExFx** has the following parameters:

- The parameter **sFx** passes a string that contains an expression used to calculate a value for a regression variable.
- The parameter **X** passes the matrix of values for the observed independent variables.
- The parameter **Y** passes the array of values for the observed dependent variable.
- The parameter **I** passes the row index for an observation.
- The parameter **NumIndpVars** passes the number of independent variables.

The function returns the value for a regression variable based on the values of the observed variables. The function uses a downward-counting **For** loop to replace strings X_n with a corresponding numeric value. This approach helps the function to correctly translate, say variables X19 through X10, and later variable X1, without any mix-up.

The subroutine **MLR** performs the linearized multiple regression. The subroutine reads its input data from the currently selected worksheet. That worksheet shows the input data prepared in the leading columns, as described earlier. The subroutine performs the following general tasks:

- Read the input from the leading columns.
- Transform the observed variables into the regression variables using the expressions in column A.

- Invoke the **Linest** function and store its results in variable **vRegResultsMat**.
- Build the output by either displaying the values in variable **vRegResultsMat**. This task also calculates some results (such as the confident intervals and p-values) based on values in variable **vRegResultsMat** and then displays the results.

```
Option Explicit
Option Base 1

Function ExFx(ByVal sFx As String, ByRef X() As Variant, _
              ByRef Y() As Variant, _
              ByVal I As Integer, ByVal NumIndpVars As Integer) _
              As Double
 Dim J As Integer
 ' replace Xnn starting with the higher indices just in case
 ' there are more than 9 variables.
sFx = UCase(sFx)
For J = NumIndpVars To 1 Step -1
   sfx = Replace(sFx, "X" & J, "(" & X(I, J) & ")")
 Next J
 sFx = Replace(sFx, "Y", "(" & Y(I, 1) & ")")
 ExFx = Evaluate(sFx)
End Function

Sub MLR()
  Dim X() As Variant, Y() As Variant, TX() As Variant
  Dim I As Integer, J As Integer, K As Integer, M As Integer
  Dim N As Integer, J1 As Integer, J2 As Integer, Col As Integer
  Dim NumIndpVars As Integer, NumTrnsX As Integer
  Dim Tstat As Double, ConfidenceLevel As Double, Alpha As Double
  Dim Sum As Double, yhat As Double, AICc As Double
  Dim sfx As String
  Dim vRegResultsMat As Variant
  Dim xval As Double, df As Integer

  NumTrnsX = [A2].Value
  ConfidenceLevel = [A4].Value
  If ConfidenceLevel > 1 Then ConfidenceLevel = ConfidenceLevel / 100
  N = Range("B1").CurrentRegion.Rows.Count - 1
  NumIndpVars = Range("A1").CurrentRegion.Columns.Count - 2
  ReDim TX(N, NumTrnsX), X(NumIndpVars), Y(N)

  Y = Range("B2:B" & N + 1).Value
  X = Range(Cells(2, 3), Cells(N + 1, 2 + NumIndpVars)).Value
  Range(Cells(1, 4 + NumIndpVars), Cells(1000, 100 +
NumIndpVars)).Value = ""
```

```
  sfx = [A5].Value
  For I = 1 To N
    Y(I, 1) = ExFx(sfx, X, Y, I, NumIndpVars)
  Next I

  M = 6
  For J = 1 To NumTrnsX
    sfx = Range("A" & M).Value
   For I = 1 To N
      TX(I, J) = ExFx(sfx, X, Y, I, NumIndpVars)
    Next I
    M = M + 1
  Next J

  vRegResultsMat = Application.WorksheetFunction.LinEst(Y, TX, True,
True)

  Col = NumIndpVars + 4
  ' output tags
  Cells(1, Col) = "Intercept/Slopes"
  Cells(2, Col) = "Std Err"
  Cells(3, Col) = "Int/Slopes C.I. Upper"
  Cells(4, Col) = "Int/Slopes C.I. Lower"
  Cells(5, Col) = "P-Value"
  Cells(6, Col) = "Rqsr"
  Cells(7, Col) = "F"
  Cells(8, Col) = "Regression SS"
  Cells(9, Col) = "SS"
  Cells(10, Col) = "AICc"
  Cells(11, Col) = "Mod AICc"

  Col = Col + 1
  ' display coefficients in reverse order--a0, a1, a2, ..., an
  For J = 0 To NumTrnsX
    J2 = NumTrnsX - J + 1
    Cells(1, Col + J) = vRegResultsMat(1, J2)
    Cells(2, Col + J) = vRegResultsMat(2, J2)
  Next J

  Alpha = 1 - ConfidenceLevel
  Tstat = WorksheetFunction.Tinv(Alpha / 2, N - NumTrnsX - 1)

  ' display confidence intervals for regression coefficients
  ' and p-values
```

```
For I = 0 To NumTrnsX
  J = Col + I
  Cells(3, J) = Cells(1, J) + Tstat * Cells(2, J)
  Cells(4, J) = Cells(1, J) - Tstat * Cells(2, J)
  xval = Abs(Cells(1, J) / Cells(2, J))
  df = vRegResultsMat(4, 2)
  ' display p-value
  Cells(5, J) = WorksheetFunction.T_Dist_2T(xval, df)
Next I

' r^2
Cells(6, Col) = vRegResultsMat(3, 1)
Cells(6, Col + 1) = vRegResultsMat(3, 2)
Cells(6, Col + 2) = "SE(y)"
' F & df
Cells(7, Col) = vRegResultsMat(4, 1)
Cells(7, Col + 1) = vRegResultsMat(4, 2)
Cells(7, Col + 2) = "DF"
' SS
Cells(8, Col) = vRegResultsMat(5, 1)
Cells(8, Col + 1) = vRegResultsMat(5, 2)
Cells(8, Col + 2) = "Residual SS"
Cells(9, Col) = vRegResultsMat(5, 1) + vRegResultsMat(5, 2)
Cells(9, Col + 1) = Tstat
Cells(9, Col + 2) = "Tinv"

' Calculate AICC
Sum = 0
For I = 1 To N
  yhat = vRegResultsMat(1, 1)
  For J = 1 To NumTrnsX
    J2 = NumTrnsX - J + 1
    yhat = yhat + TX(I, J) * vRegResultsMat(1, J2)
  Next J
  Sum = Sum + (Y(I, 1) - yhat) ^ 2
Next I
If Sum < 0.000000000000001 Then Sum = 0.000000000000001
Sum = Sum / N
K = NumTrnsX + 1
AICc = N * Log(Sum) + 2 * K + (2 * K * (K + 1)) / (N - K - 1)
Cells(10, Col) = AICc
Cells(10, Col + 1) = 2 * K + (2 * K * (K + 1)) / (N - K - 1)
Cells(10, Col + 2) = "AICc baseline"
' calculate my version of AICc
Sum = 0
```

```
  For I = 1 To N
    yhat = vRegResultsMat(1, NumTrnsX + 1)
    For J = 1 To NumTrnsX
      J2 = NumTrnsX - J + 1
      yhat = yhat + TX(I, J) * vRegResultsMat(1, J2)
    Next J
    Sum = Sum + (Y(I, 1) - yhat) ^ 2
  Next I
  Sum = 1 + Sum / N
  K = NumTrnsX + 1
  AICc = N * Log(Sum) + 2 * K + (2 * K * (K + 1)) / (N - K - 1)
  Cells(11, Col) = AICc
End Sub
```

Listing1. The MLR subroutine.

References

1. Shammas, Namir, *Best Multiple Regression Model Using Excel Linest Function, article on www.namirshammas.com web site.*
2. Shammas, Namir, *Best Polynomial Model Using Excel Linest Function, article on www.namirshammas.com web site.*
3. Wikipedia article *Coefficient of Determination*.
4. Wikipedia article *Linear Regression*.
5. Wikipedia article *Simple Linear Regression*.
6. Wikipedia article *Akaike information criterion*.
7. Draper and Smith, *Applied Regression Analysis*, Wiley-Interscience; 3rd edition (April 23, 1998).
8. Neter, Kuther, Wasserman, and Nachtsheim, *Applied Linear Statistical Models*, McGraw-Hill/Irwin; 4th edition (February 1, 1996).
9. Fox, *Applied Regression Analysis and Generalized Linear Models*, Sage Publications, Inc; 2nd edition (April 16, 2008).
10. Montgomery, Peck, and Vining, *Introduction to Linear Regression Analysis*, Wiley-Interscience; 4th edition (2006).
11. Seber and Lee, *Linear Regression Analysis*, Wiley; 2nd edition (February 5, 2003).

About this Article

Version	*Date*	*Comment*
1.0.0.0	12/7/2012	Initial release.