HP 39gII Regression: Part IV
Least-Squares Relative Error Regression

Namir Shammas

Introduction

Ordinary least-squares (OLS) errors regression minimizes the sum of squared errors for the regression models. These regression models include all kinds of linear and nonlinear equations that describe the relationship between a dependent variable and one or more independent variables. Conceptually, OLS regression treats all errors, in the dependent variable, as having an equal weight, concern, and importance. There are cases where you need to reduce the relative (or percentage) errors. Consider the example where you are measuring distance versus time to calculate the speed. You have distance readings between 10 ft, and 100 ft in increments of 5 feet. Repeated measurements tell you that you get an error of $\pm$0.5 ft for each reading. Of course, an error of $\pm$0.5 ft for a 10 ft reading is more serious than for that for the higher readings. You like to perform a curve fit that minimizes the least-squares *relative* errors so that you regression model gives better prediction for smaller distance values. This is where least-squares relative error (LSRE) regression is relevant. The reality of statistical calculations has favored OLS regression far more than LSRE regression, since the OLS equations are simpler to derive. This is somewhat unfortunate, since it does not give researchers a fair choice between OLS and LSRE regression analysis tools. This article looks at LSRE regression using the HP 39gII calculator.

In this article I present three HP 39gII functions that are the RLSE versions of OLS regression functions that I presented in part I of this series. The two flavors of regression functions have the same parameter lists and returned results. These similarities make the RLSE functions easy to use once you have become familiar with their OLS counterpart.

☞ In this series of articles, I use the term *regression model* to mean the equation that is used in the regression calculations to describe the relationship between a dependent variable and one or more independent variables.

LSRE Regression 101

Consider the simplest linear regression model:

$y = A + B x$

Where A and B are the intercept and slope, respectively. Variables x and y are the independent and the dependent variables, respectively. The following equation calculates the error for the linear regression:

$OE_i = y_i – A – B x_i$

In the case of least-squares relative errors, the relative error is:

$RE_i = OE_i / y_i = (y_i – A – B x_i) / y_i = (1– A/y_i – B x_i/y_i)$

☞ The RLSE regression cannot have zeros as values for the dependent variable or its transformation. Such values generate runtime divide-by zero errors when calculating the least-squares relative errors.

OLS regression minimizes the sum of squared errors as expressed in the following equation:

$L_{OLS} = \sum(y_i - A - B\ x_i)^2$

Whereas LSRE regression minimizes the sum of squared relative errors as expressed in the following equation:

$L_{LSRE} = \sum(1 - A/y_i - B\ x_i/y_i)^2$

To obtain the values of A and B for OLS regression, you perform the following steps:

1. Expand the summation of L_{OLS}.
2. Derive L_{OLS} with respect to A and with respect to B to obtain two linear equations.
3. Solve the two linear equations to evaluate A and B using various statistical summations of x and y.

The above steps yield the following commonly known equations for the slope and intercept:

$B = [n \cdot \sum x_i y_i - \sum x_i \cdot \sum y_i] / [\ n \cdot \sum x_i\text{^}2 - (\sum x_i)\text{^}2]$

$A = (\sum y_i - B \cdot \sum x_i) / n$

Where n is the number of observations. In the case of LSRE regression we follow a similar procedure to calculate A and B from the equation for L_{LSRE}. The equations that calculate B and A are:

$B = b1 / b2$

$b1 = \sum(x_i/y_i) \cdot \sum(1/y_i\text{^}2) - \sum(x_i/y_i\text{^}2) \cdot \sum(1/y_i)$

$b2 = \sum((x_i/y_i)\text{^}2) \cdot \sum(1/y_i\text{^}2) - [\sum(x_i/y_i\text{^}2)]\ \text{^}2$

$A = [\sum(1/y_i) - B \cdot \sum(x_i/y_i\text{^}2)]\ / \sum(1/y_i\text{^}2)$

The above equations are quite different from their OLS counterpart. Also notice that calculating the RLSE slope and intercept does not involve the number of observations.

In matrix form, you can calculate the OLS regression coefficients vector, **b**, for a linearized regression model using the following equation:

$\mathbf{b} = (\mathbf{X}^T\ \mathbf{X})^{-1}\ (\mathbf{X}^T\ \mathbf{y})$

Where matrix **X** has multiple columns that represent values for the independent variables and/or their transformations. These transformations include the natural logarithm, reciprocal, and square, just to name a few. The first column in matrix **X** is typically filled with the constant 1 to calculate a constant for the fitted regression model. The column vector **y** contains the values of the dependent variable or its transformations.

In the case of LSRE regression the matrix form for solving vector **b** is:

$\mathbf{b} = (\mathbf{X}^T\ \mathbf{D}^2\ \mathbf{X})^{-1}\ (\mathbf{X}^T\ \mathbf{D}^2\ \mathbf{y})$

Where **D** is a square matrix with zeros except for diagonal elements having $1/y_i$ values.

To calculate the goodness of fit for the LSRE regression, you obtain the value for the coefficient of determination using the following equation:

$$R^2_{LSRE} = \frac{\sum_1^n((\hat{y}_i - \bar{y})/y_i)^2}{\sum_1^n((y_i - \bar{y})/y_i)^2}$$

Where $\hat{y}_i$ is the predicted value of y at the values of the independent variables used in the calculations, $\bar{y}$ is the average value of y, and y_i is the values of y entering in the regression calculations. Compare the above equation with its OLS regression counterpart:

$$R^2_{OLSE} = \frac{\sum_1^n(\hat{y}_i - \bar{y})^2}{\sum_1^n(y_i - \bar{y})^2}$$

A Calculation Trick

You may have noticed that I jumped from the basic equations for calculating a simple LSRE linear regression model to presenting the matrix form that describes the LSRE calculations for a wide variety of regression models. I could have gradually presented more advanced LSRE regression equations. For example, I could have listed the equations for an LSRE quadratic polynomial regression and also for the simplest LSRE multiple regression that involves two dependent variables. This process would have added several pages filled with equations. Instead I chose to present a calculation trick where we can use the OLS regression calculations to calculate the coefficients for LSRE regression models. The trick relies on looking at the expression for L_{LSRE} and viewing it differently--as if it were L_{OLS}:

$L_{OLS} = \sum(1 - A/y_i - B\ x_i/y_i)^2$

The above equation indicates that we are fitting the following OLS regression model:

$1 = A/y + B\ x/y$

Or,

$\alpha = A/y + B\ x/y$

The above (very strange) equation is an OLS multiple regression model with the following features:

- The values for the (new) dependent variable (call it α) are always 1.
- The model is a linearized multiple regression. It uses the independent variables x and y in the transformations 1/y and x/y. Notice that I have changed the status of the original dependent variable y, temporarily making it an independent variable.
- The regression model has NO constant term.

Likewise, if we have a quadratic polynomial regression model:

$y = A + B\ x + C\ x^2$

Our trick yields the following equation that can use OLS regression calculations:

$1 = A/y + B\ x/y + C\ x^2/y$

In the case of a simple multiple linear regression model:

$Y = A + B\ x + C\ z$

Our trick yields the following equation that can use OLS regression calculations:

1 = A/y + B x/y + C z/y

I have offered you the above examples so that you can see the pattern used in the trick that allows us to apply OLS regression calculations with any RLSE regression model. Notice that in all of the above examples (and for any other regression model), each right-hand-side term in the regression model is divided by the dependent variable y. Also notice that each left-hand-side is the new variable α, whose values are systematically equal to 1.

The above features allow us to use OLS regression calculation tools, such as the function **LSQ** on the HP 39gII. However, we *MUST* observe the following simple calculation rules.

- The matrix **X** has no column populated by 1s, since the regression calculations yields NO constant term.
- Each column in matrix **X** has a term from the modified multiple regression model.
- The column vector **y** now represents the new variable α and is populated with 1s.

To obtain the LSRE version for the coefficient of determination, we must use the definition for that statistic that I showed you earlier. We must also observe the following rules:

- The values of $\hat{y}_i$ used to calculate the LSRE coefficient of determination must be based on the *original* regression model. This is the model that shows variable y as the dependent variable.
- If you transformed the observed values of y, then you need to work with the transformed values and not the original observed ones. Likewise, the values for $\hat{y}_i$ must be for the transformed values of y.
- We cannot use the formula for the OLS coefficient of determination with LSRE regression models, because each version is defined differently. That would be like mixing apples and oranges!

General RLSE Regression

Let's start with the general case of simple multiple regression, with nor transformation of variables. I introduced you, in part I, the function **MLR2** to calculate the coefficient of determinations and the regression coefficients. Table 1 presents function **RErMLR2** which is the RLSE counterpart of function **MLR2**. The function **RErMLR2** has the following parameters:

- The parameter **MatX** which represents the matrix **X**.
- The parameter **VectY** which represents the vector **y**.

The function returns a list containing the coefficient of determination and the regression coefficients (as a column matrix).

Statement
`EXPORT RErMLR2(MatX,VectY)`
`BEGIN`
`  LOCAL i,j,y,Rsqr;`
`  LOCAL lstDimX,NumRows,NumCols;`
`  LOCAL YMean,Sum1,Sum2,Yhat;`
`  LOCAL TMatX,VectOnes,RegCoeff;`

Statement
`// calculate the number of observations and variables`
`lstDimX:=SIZE(MatX);`
`NumRows:=lstDimX(1);`
`NumCols:=lstDimX(2);`
`// create transformation matrices`
`TMatX:=MAKEMAT(1,NumRows,NumCols+1);`
`VectOnes:=MAKEMAT(1,NumRows,1);`
`FOR i FROM 1 TO NumRows DO`
`y:=VectY(i,1);`
`TMatX(i,1):=1/y;`
`FOR j FROM 1 TO NumCols DO`
`TMatX(i,j+1):=MatX(i,j)/y;`
`END;`
`END;`
`// calculate the regression coefficients`
`RegCoeff:=LSQ(TMatX,VectOnes);`
`// calculate ymean`
`Sum1:=0;`
`FOR i FROM 1 TO NumRows DO`
`y:=VectY(i,1);`
`Sum1:=Sum1+y;`
`END;`
`YMean:=Sum1/NumRows;`
`// calculate the coefficient of determination`
`Sum1:=0;`
`Sum2:=0;`
`FOR i FROM 1 TO NumRows DO`
`y:=VectY(i,1);`
`Yhat:=RegCoeff(1,1);`
`FOR j FROM 1 TO NumCols DO`
`Yhat:=Yhat+RegCoeff(j+1,1)*MatX(i,j);`
`END;`
`Sum1:=Sum1+((Yhat-YMean)/y)^2;`
`Sum2:=Sum2+((y-YMean)/y)^2;`
`END;`
`Rsqr:=Sum1/Sum2;`
`// return the results as a list`
`RETURN {Rsqr,RegCoeff};`
`END;`

Table 1 – The source code for function RErMLR2.

Since the topic of LSRE regression is not as popular as OLS regression, I would like to point out the following aspects of the source code in Table 1:

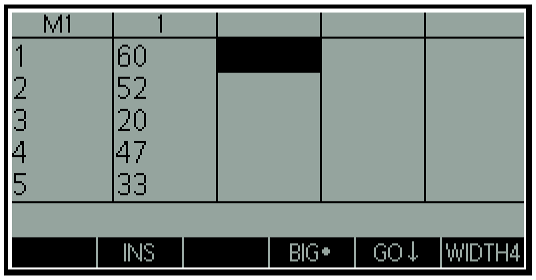
- The function creates the column vector **VectOnes** using the **MAKEMAT** function. This task fills the vector **VectOnes** with 1s. These values remain unchanged during the function execution because they represent the values of the variable α.
- The values for the first column of matrix **MatX** are 1/y instead of 1 (as is the case in OLS regression).
- The values for the remaining columns of matrix **MatX** as x_n divided by y.
- The loop that calculates the LSRE coefficient of determination uses values for $\hat{y}_i$ calculated as $a_1 + a_2\ x_1 + a_3\ x_2 + \ldots + a_{m+1}\ x_m$. The column matrix variable **RegCoeff** contains the regression coefficients a_1, a_2, a_3, …, and a_{m+1}.
- The function calculates the LSRE coefficient of determination using the definition that I presented for that statistic.

Let's use function **RErMLR2** with the sample data in Table 2. This is the same data I used to test function **MLR2** in part I. I am reusing the same data so I can compare the results of functions **RErMLR2** and **MLR2**.

x_1	x_2	x_3	y
7	25	6	60
1	29	15	52
11	56	8	20
11	31	8	47
7	52	6	33

Table 2 – Sample data.

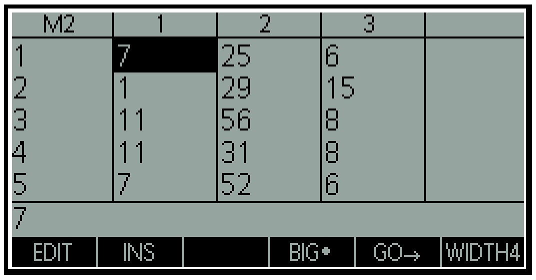
You need to store the values of the dependent variable y as a column in matrix M1 and the values of the dependent variables in matrix M2. Figure 1 shows the contents of matrix M1 which stores the values for the column vector **y**.



M1	1
1	60
2	52
3	20
4	47
5	33

Fig. 1 – The values in matrix M1.

Figure 2 shows the contents of matrix M2 which stores the values for the matrix **X**.



M2	1	2	3
1	7	25	6
2	1	29	15
3	11	56	8
4	11	31	8
5	7	52	6

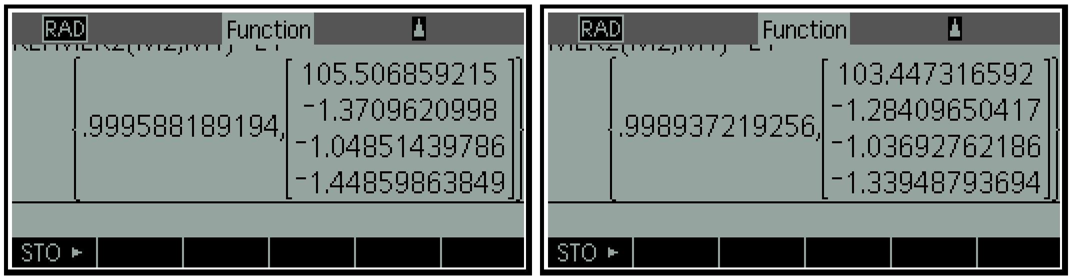
7

Fig. 2 – The values in matrix M2.

Let's use function **RErMLR2** with the data in Table 2. Type the following command:

RErMLR2(M2,M1)→L1

The above command stores the results in list L1 for further examination if so desired. Figure 3 shows the results of executing the function **RErMLR2** along with the output of function **MLR2** using the same data:



```
{.999588189194,[[105.506859215]
                [-1.3709620998]
                [-1.04851439786]
                [-1.44859863849]]}
```

```
{.998937219256,[[103.447316592]
                [-1.28409650417]
                [-1.03692762186]
                [-1.33948793694]]}
```

Fig. 3 – The results of executing function RErMLR2 (left) and function MLR2 (right).

The results in Figure 3 show that the RLSE R^2 is 0.99959. This value is slightly higher than the OLS R^2 which is 0.99894. The RLSE regression model is:

$y = 105.5069 - 1.3710\ x_1 - 1.0485\ x_2 - 1.44860\ x_3$

The regression coefficients for both RLSE and OLS regression are somewhat close to each other, as expected.

LSRE Polynomial Regression

Next we will be working with the LSRE polynomial regression. Table 3 shows the source code for the function **RErPolyReg**. This function has the following parameters:

- The parameter **DSMat** represents the matrix that has the x and y data.
- The parameter **SelXCol** designates the column in matrix **DSMat** which contains the values for x.
- The parameter **SelYCol** designates the column in matrix **DSMat** which contains the values for y.
- The parameter **Order** selects the order for the polynomial regression. If you pass an argument of 1 to this parameter, the function **LSREPolyReg** performs an LSRE linear regression. Passing values of 2 and 3 to the parameter **Order** cause the function to fit the data with a quadratic and cubic polynomial, respectively.

The function **RErPolyReg** returns a list that contains the value of the LSRE coefficient of determination and a column matrix containing the regression coefficients. I recommend that you store the results of calling function **RErPolyReg** in a list so that you can further examine and/or use the results.

Statement
`EXPORT RErPolyReg(DSMat,SelXCol,SelYCol,Order)`
`BEGIN`
`  LOCAL i,j,x,y;`
`  LOCAL lstDimX,NumRows;`
`  LOCAL MatX,VectOnes,RegCoeff;`
`  LOCAL YMean,Sum1,Sum2,Yhat,Rsqr;`
`  // get the number of rows`

Statement
lstDimX:=SIZE(DSMat);
NumRows:=lstDimX(1);
// create the regression matrices
MatX:=MAKEMAT(1,NumRows,Order+1);
VectOnes:=MAKEMAT(1,NumRows,1);
// populate the regression matrices
FOR i FROM 1 TO NumRows DO
y:=DSMat(i,SelYCol);
x:=DSMat(i,SelXCol);
MatX(i,1):=1/y;
FOR j FROM 1 TO Order DO
MatX(i,j+1):=(x^j)/y;
END;
END;
// calculate the regression coefficients
RegCoeff:=LSQ(MatX,VectOnes);
// calculate ymean
Sum1:=0;
FOR i FROM 1 TO NumRows DO
Sum1:=Sum1+DSMat(i,SelYCol);
END;
YMean:=Sum1/NumRows;
// calculate the coefficient of determination
Sum1:=0;
Sum2:=0;
FOR i FROM 1 TO NumRows DO
x:=DSMat(i,SelXCol);
y:=DSMat(i,SelYCol);
Yhat:=RegCoeff(1,1);
FOR j FROM 1 TO Order DO
Yhat:=Yhat+RegCoeff(j+1,1)*x^j;
END;
Sum1:=Sum1+((Yhat-YMean)/y)^2;
Sum2:=Sum2+((y-YMean)/y)^2;
END;
Rsqr:=Sum1/Sum2;
// return the list of results
RETURN {Rsqr,RegCoeff};
END;

Table 3 – The source code for function RErPolyReg.

Since the topic of LSRE polynomial regression is not as popular as OLS polynomial regression, I would like to point out the following aspects of the source code in Table 3:

- The function creates the column vector **VectOnes** using the **MAKEMAT** function. This task fills the vector **VectOnes** with 1s. These values remain unchanged during the function execution because they represent the values of the variable α.
- The values for the first column of matrix **MatX** are 1/y.
- The values for the remaining columns of matrix **MatX** are x, raised to some power, and then the result divided by y.
- The loop that calculates the LSRE coefficient of determination uses values for $\hat{y}_i$ calculated as $a_1 + a_2\ x + a_3\ x^2 + \ldots + a_{m+1}\ x^m$. The column matrix variable **RegCoeff** contains the regression coefficients a_1, a_2, a_3, …, and a_{m+1}.

Let's use the function **RErPolyReg** to fit a cubic polynomial using the data in Table 4. Enter the data in matrix M1.

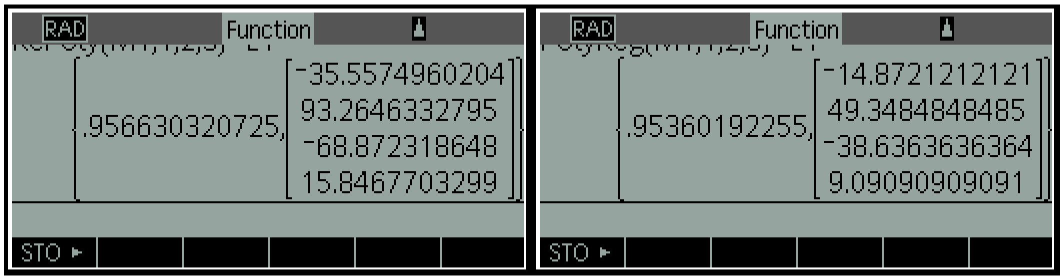
x	*y*
1	5.1
1.1	4.4
1.2	4.6
1.3	4.0
1.4	3.2
1.5	3.2
1.6	2.4
1.7	2.2
1.8	1.3
1.9	2.0

Table 4 – Sample data for a cubic polynomial fit.

To obtain the regression coefficients and coefficient of determination for the cubic polynomial fit using the data in Table 4, execute the following command:

```
RErPolyReg(M1,1,2,3)→L1
```

The first argument of calling function **RErPolyReg** is the matrix M1 which contains the (x, y) data points. The second argument is 1 which tells the function that the data for the independent variable x are in column 1 of M1. The Third argument is 2, which tells the function that the data for the dependent variable y are in column 2 of M1. The last argument is 3, which represent the order of the sought polynomial. I assigned the results to list L1 so that I can examine the results later, if I needed to. Figure 4 shows the output of using function **RErPolyReg** as well as function **PolyReg** (from part I)/.



```
RAD   Function
{.956630320725, [-35.5574960204]
                [93.2646332795 ]
                [-68.872318648 ]
                [15.8467703299 ]}
STO ►
```

```
RAD   Function
{.95360192255, [-14.8721212121]
               [49.3484848485 ]
               [-38.6363636364]
               [9.09090909091 ]}
STO ►
```

Fig. 4 – The results of executing function LSRErPolyReg (left) and function PolyReg (right).

The results show that R^2 is 0.95663 and the fitted polynomial is:

$Y = –35.5575 + 93.2646\ x – 68.87232\ x^2 + 15.84677\ x^3$

The value of R^2 indicates that the polynomial explains about 95% of the variation in the values of y. The value of the RLSE R^2 is slightly higher than of its OLS counterpart. The coefficients of RLSE polynomial regression are quite different in values from their OLS counterpart.

Power Curve Fitting

This section looks at the RLSE version of power fitting. I presented function **PowerFit** in part I. In this section I present, in Table 5, its RLSE counterpart, function **RErPowerFit**. The function **RErPowerFit** has two parameters. The first parameter is **MatX**--the name of the matrix that contains the values for the independent variables. The second parameter is **VectY**—the name of the column vector that contains the values for the dependent variable. The function returns a list containing the RLSE coefficient of determination and the column matrix that stores the regression coefficients.

Statement

```
EXPORT RErPowerFit(MatX,VectY)
BEGIN
  LOCAL i,j,y,LnY;
  LOCAL lstDimX,NumRows,NumCols;
  LOCAL TMatX,VectOnes,RegCoeff,Rsqr;
  LOCAL YMean,Sum1,Sum2,Yhat;

  lstDimX:=SIZE(MatX);
  NumRows:=lstDimX(1);
  NumCols:=lstDimX(2);
  TMatX:=MAKEMAT(1,NumRows,NumCols+1);
  VectOnes:=MAKEMAT(1,NumRows,1);
  Sum1:=0;
  FOR i FROM 1 TO NumRows DO
    LnY:=LN(VectY(i,1));
    Sum1:=Sum1+LnY;
    TMatX(i,1):=1/LnY;
    FOR j FROM 1 TO NumCols DO
      TMatX(i,j+1):=LN(MatX(i,j))/LnY;
    END;
  END;
  // calculate regression coefficients
  RegCoeff:=LSQ(TMatX,VectOnes);
  // calculate mean ln(y)
  YMean:=Sum1/NumRows;

  // calculate coefficient of determination
  Sum1:=0;
  Sum2:=0;
  FOR i FROM 1 TO NumRows DO
    Yhat:=RegCoeff(1,1);
    FOR j FROM 1 TO NumCols DO
      Yhat:=Yhat+RegCoeff(j+1,1)*LN(MatX(i,j));
```

Statement
`    END;`
`    y:=LN(VectY(i,1));`
`    Sum1:=Sum1+((Yhat-YMean)/y)^2;`
`    Sum2:=Sum2+((y-YMean)/y)^2;`
`  END;`
`  Rsqr:=Sum1/Sum2;`
`  RETURN {Rsqr,RegCoeff};`
`END;`

Table 5 – The source code for function RErPowerFit.

To test the function **RErPowerFit** let's used the data in Table 6.

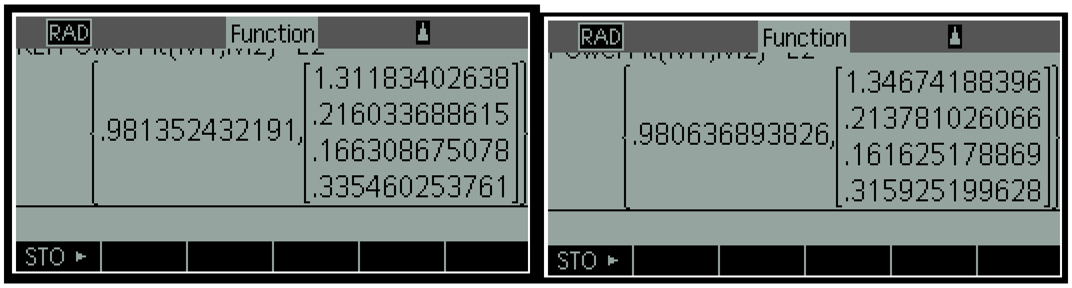
x	*z*	*t*	*y*
1	1	7	7
2	1	5	7.7
3	2	3	7.9
4	2	1	5.3
5	3	2	8.4
6	3	5	11.6
7	4	8	13.6
8	4	9	14.3
9	5	4	12.4
10	5	2	10.6

Table 6 – Sample data for a power fit.

Store the values in the first three columns of Table 6 in matrix M1. Store the values of the rightmost column of Table 6 in the matrix M2. To calculate the regression coefficient of a power fit between variables x, z, t, and y, execute the following command:

```
RErPowerFit(M1,M2)→L2
```

Figure 5 shows the results of executing the above command as well as the results from using function **PowerFit** (in Part I).



```
{.981352432191, [[1.31183402638]
                 [.216033688615]
                 [.166308675078]
                 [.335460253761]]}
```

```
{.980636893826, [[1.34674188396]
                 [.213781026066]
                 [.161625178869]
                 [.315925199628]]}
```

Fig. 5 – The results of executing function RErPowerFit (left) and function PowerFit (right).

The results show that R^2 is 0.98135 and the power fit is:

y = 1.31183 + 0.216033 * ln(x) + 0.1663086 * ln(z) + 0.33546 * ln(t)

Or is in the nonlinear form,

y = 3.712962 * (x^ 0.216033)*(z^0.1663086)*(t^0.33546)

The value of R^2 indicates that the power fit explains about 98% of the variation in the values of y. The RLSE R^2 value is slightly higher than its OLS counterparts. The corresponding coefficients of the RLSE and OLS polynomial regression are close to each other, as expected.

Comparing RLSE and OLS Curve Fitting

The three HP 39gII RLSE regression functions that I presented in this article produced better values for the coefficient of determination than their OLS counterparts. In his paper, Tofallis[1] makes reference to the work of Saez and Rittmann[2] that have performed Mote Carlo simulations and found that RLSE regression produced better results than OLS regression models. These researchers found that the RLSE regression models yielded regression coefficients with 90% confidence regions that were approximately centered on the true coefficient values. This was not the case with OLS results.

To Infinity and Beyond!

You have seen the source code for three RLSE regression functions. You should be able to convert other HP 39gII regression functions that I presented in parts II and III into RLSE versions. The tasks include:

- Populating the matrix **X** with the correct values. The values in the first column are always $1/y_i$. The values for the other columns should be calculated as the values of the independent variables (or their transformations) divided by y_i.
- Populating the vector **y** with the constant 1.
- Making sure that you correctly calculate the RSLE version of the coefficient of determination.

Observations and Conclusions

This article discussed least-squares relative errors. The article first presented some theoretical foundation for basic RLSE calculations. The article also showed you how apply a trick to use OLS regression calculations for RLSE regression. You also learned about calculating the RLSE coefficient of determination. The article also presented three RLSE regression functions for the HP 39gII. These functions are the RSLE versions of OLS regression functions that appear in part I of this series.

References

1. Chris Tofallis, *Least Squares Percentage Regression*, Journal Of Modern Applied Statistical Methods, November, 2008, Vol. 7, No. 2, 368-631.

2. Saez and Rittmann, *Model-parameter estimation using least squares*. Water Research, 26(6), 789-796, 1992.

3. Wikipedia article *Coefficient of Determination*.

4. Wikipedia article *Linear Regression*.

5. Wikipedia article *Simple Linear Regression*.

6. Draper and Smith, *Applied Regression Analysis*, Wiley-Interscience; 3rd edition (April 23, 1998)

7. Neter, Kuther, Wasserman, and Nachtsheim, *Applied Linear Statistical Models*, McGraw-Hill/Irwin; 4th edition (February 1, 1996).

8. Fox, *Applied Regression Analysis and Generalized Linear Models*, Sage Publications, Inc; 2nd edition (April 16, 2008).

9. Montgomery, Peck, and Vining, *Introduction to Linear Regression Analysis*, Wiley-Interscience; 4th edition (2006).