How to perform curvilinear regression analysis with R

Axel Drefahl | axeleratio@gmail.com | axeleratio.com

Last updated: March 10, 2019

Summary

Curvilinear regression (CLR) analysis can easily be performed in the R software environment. Here, we demonstrate how to derive a parabolic relationship, a special case of a curvilinear relationship, with a few lines of R programming. Fitting a parabolic model is done in four basic steps, which consist in (1) entering the data or importing the data via CSV file into a data frame structure, (2) calculating and adding the desired powers of the values for the independent variable(s) to the data frame, (3) using function `lm()` to derive the model and (4) reporting results and extracting result values for further analysis.
This document has been made available at
www.axeleratio.com/math/comp/linreg/curvilinreg.pdf

Keywords: Curvilinear regression, polynomial regression, R, free software, programming, statistical computing.

Introduction

Computationally, **curvilinear regression** (**CLR**) analysis is not very different from **multiple linear regression** (**MLR**) analysis. To fit a curvilinear relationship (polynomial relationship), we follow exactly the same procedure as fitting a multiple regression [4]. Let's assume we have a dataset of observed values for an independent variable x and a dependent variable y, for which we found the relationship not adequately represented by a model resulting from simple linear regression (SLR). Then, we may consider curvilinear regression by including powers of x. Treating powers $x, x^2, \ldots, x_p$ like the independent variables in MLR (compare with equation 1 in "MLR with R"), we get:

$$y' = a + \sum_{j=1}^{p} b_j x^j \tag{1}$$

In this equation, a and the b_j's are **regression coefficients** and y' is the **response variable**. This model approach can be extended to the dependence of y on various independent variables—each one included up to a certain power. Here, we consider the special case of **parabolic regression** ($p = 2$). The derivation of a **parabolic model** (also named **quadratic model**) with R is demonstrated by using a published set of sample data. With the R code of this example at hand, R programming to model relationships with multiple variables at various power levels will be computationally straightforward, while interpretation of the obtained results may become more complex.

Hands-on data

We use the data of Example 8.3 in [4] that studies monthly usage of coke as a function of the air/steam ratio for a water-gas plant. The independent variable x is the air/steam ratio (1,000 m^3 air/ton steam) and the dependent variable y is coke efficiency (coke used per 1,000 m^3 of (H_2 + CO) produced). The values are listed in Table 1 in the Appendix and are also available with a CSV file:
www.axeleratio.com/math/comp/linreg/csv/woodward83.csv.

The scatter diagram (y vs. x, Fig. 8.6 in [4]) suggests that there is a relationship, but not a linear one. For the parabolic model, Woodward gives the following calculated values: $a = 280.9$, $b_1 = -323.54$, $b_2 = 112.25$ $s_{b_1} = 81.2$ and $s_{b_2} = 22.3$. We have calculated the response values and residuals, which are given in columns 5 and 6, respectively, in Table 1. The boldface residual entries are the residual minimum of -20.447 and the residual maximum of 30.194.

CLR with `lm()` in R

We use the variable `dataset` to reference the data frame storing the x and y columns of Table 1:

```
> x <- c(2.11, 2.29, 2.32, 2.31, 2.25, 2.22, 2.20, 2.41, 2.19,
+        2.06, 1.99, 1.62, 1.59, 1.70, 1.76, 1.33, 1.23, 1.40,
+        1.38, 1.96, 1.47, 1.42, 1.33, 1.65, 1.26, 1.61, 1.74)
> y <- c(120, 122, 128, 124, 118, 114, 119, 149, 141,
+        86, 78, 31, 51, 72, 51, 53, 50, 34,
+        68, 70, 49, 50, 66, 46, 40, 51, 51)
> dataset <- data.frame(x, y)
```

The same is achieved by importing the values from CSV file `woodward83.csv`:

```
> fcsv <-
+ "http://www.axeleratio.com/math/comp/linreg/csv/woodward83.csv"
> dataset <- read.csv(fcsv, header=TRUE, sep=";")
```

The `month` column in this file containd index i and can be ignored int further treatment. New columns can easily be added to an existing data frame [2]. To derive a parabolic model, we need to add the squared x values:

```
> dataset$xsq <- dataset$x ^ 2
```

The function `lm()` carries out the curvilinear modeling:

```
> curvilm <- lm(y ~ x + xsq, data = dataset)
```

The model results are displayed by calling the `summary()` function:

```
> summary(curvilm)
```

We get:

```
Call:
lm(formula = y ~ x + xsq, data = dataset)

Residuals:
    Min      1Q  Median      3Q     Max
-20.457  -6.827  -3.318   2.905  30.177

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   281.23      71.65   3.925 0.000637 ***
```

```
x           -323.80     81.60  -3.968 0.000571 ***
xsq          112.32     22.39   5.016 3.99e-05 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘’ 1

Residual standard error: 12.27 on 24 degrees of freedom
Multiple R-squared: 0.8962,  Adjusted R-squared: 0.8876
F-statistic: 103.6 on 2 and 24 DF, p-value: 1.558e-12
```

The output format and the displayed descriptors are the same as described for SLR. The only difference is that the "Coefficients" section includes an additional rows to account for x^2. You can individually access these values via the model summary, for which we use the variable `ms`. Then, for example, b_2 and s_{b_2} are obtained as follows:

```
> ms = summary(curvilm)
> b2  = coef(ms)["xsq","Estimate"]
> sb2 = coef(ms)["xsq","Std. Error"]
```

To apply the derived model to a new value $v = 1.72$ for x, we all `predict()` function after putting the new data into data frame `newdata`:

```
> v   <- 1.72
> vsq <- v^2
> newdata <- data.frame(x=c(v), xsq=c(vsq))
> predict(curvilm,newdata)
       1
56.59288
```

The estimated y value matches our control calculations (when rounded to the fourth decimal):

$$1: \quad 281.2303 - 323.7978 \cdot 1.72 - 112.3225 \cdot 1.72 \cdot 1.72 = 56.59297.$$

Note that the numeric precision of the y values in the dataset is 3 or lower.

Conclusion & Outlook

The purpose here was to demonstrate how a parabolic model can be derived with R. A dataset and a corresponding CSV file for testing was provided. No attempt was made to interpret CLR results or to investigate modeling alternatives. You may also want to look at other examples demonstrating curvilinear regression in R [1, 3].

About the author

Axel Drefahl has designed scientific software for chemical property prediction at the Technical University of Munich, Germany, and Stanford University, California. At the Freiberg University of Mining and Technology he developed Monte-Carlo-simulation algorithms to virtually study interactions of functionalized nanoparticles. Axel initiated the CurlySMILES Project for the encoding of complex, annotated molecular structures, polymer systems and nanoarchitectures. His experience and interests include pattern recognition, nanoinformatics, sustainable chemistry and the history (and future) of science. Off-line, Axel enjoys the outdoors, nature studies and photography. Back online, he shares his findings and impressions on TrailingAhead, Latintos, Explore Reno-Tahoe and other sites.

Literature & Links

[1] Michy Alice. Fitting polynomial regression in r. `https://datascienceplus.com/fitting-polynomial-regression-r/`. Accessed: 2019-03-10.

[2] Sharon Machlis at ComputerWorld. 4 data wrangling tasks in R for advanced beginners: Learn how to add columns, get summaries, sort your results and reshape your data. `https://www.computerworld.com/article/2486425/`. Accessed: 2019-03-06.

[3] David Lillis. R is not so hard! a tutorial, part 4: Fitting a qudratic model. `https://www.theanalysisfactor.com/r-tutorial-4/`. Accessed: 2019-03-10.

[4] R. H. Woodward. Multiple and curvilinear regression. In O. L. Davies and P. L. Goldsmith, editors, *Statistical Methods in Research and Production*, chapter 8, pages 237–303. Longman, London and New York, 4 edition, 1984.

Appendix

The dataset of observed values used in this document are from Table 8.3 in [4]. These values are given in Table 1 along with response values and residuals calculated with equation 1 using the regression coefficients obtained by R computation. The residuals are calculated as $e_i = y_i - y'_i$. The minimum and maximum residuals appear in boldface type.

Table 1: Dataset with observed and fitted values, and residuals (see section "Hands-on data").

i	x_i	x_i^2	y_i	y'_i	e_i
1	2.11	4.4521	120	98.072	21.928
2	2.29	5.2441	122	128.745	-6.745
3	2.32	5.3824	128	134.565	-6.565
4	2.31	5.3361	124	132.603	-8.603
5	2.25	5.0625	118	121.300	-3.300
6	2.22	4.9284	114	115.952	-1.952
7	2.20	4.8400	119	112.499	6.501
8	2.41	5.8081	149	153.238	-4.238
9	2.19	4.7961	141	110.806	**30.194**
10	2.06	4.2436	86	90.843	-4.843
11	1.99	3.9601	78	81.666	-3.666
12	1.62	2.6244	31	51.447	**-20.447**
13	1.59	2.5281	51	50.344	0.656
14	1.70	2.8900	72	55.375	16.625
15	1.76	3.0976	51	59.264	-8.264
16	1.33	1.7689	53	49.259	3.741
17	1.23	1.5129	50	52.885	-2.885
18	1.40	1.9600	34	48.057	-14.057
19	1.38	1.9044	68	48.288	19.712
20	1.96	3.8416	70	78.071	-8.071
21	1.47	2.1609	49	47.956	1.044
22	1.42	2.0164	50	47.916	2.084
23	1.33	1.7689	66	49.259	16.741
24	1.65	2.7225	46	52.751	-6.751
25	1.26	1.5876	40	51.561	-11.561
26	1.61	2.5921	51	51.057	-0.057
27	1.74	3.0276	51	57.878	-6.878