How to perform multiple linear regression analysis with R

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

Last updated: March 10, 2019

Summary

The freely available R software environment is a user-friendly place to work on linear modeling tasks. Here, we demonstrate how basic parameters of multiple linear regression (MLR) models are obtained with R by only writing a few lines of code. An example dataset with calculated results is provided to get a hands-on start with MLR analysis. We show that fitting a linear model (or rejecting it) for given variables with sample data can be done in three basic steps: (1) importing the data via CSV file, (2) using function `lm()` to derive the model and (3) reporting results or extracting result values for further analysis.
This document has been made available at
www.axeleratio.com/math/comp/linreg/multilinreg.pdf
and a brief introduction was posted to the axeleratio blog.

Keywords: Linear regression, R, free software, programming, statistical computing.

Introduction

Multiple linear regression (**MLR**) is a method to test and establish **linear relationships** between one **dependent variable** and two or more

independent variables. We consider a dataset of observed values represented in a table containing n rows. Each row has one value for the dependent variable y and p values for the independent variables x_j; with $j = 1, \ldots, p$. Then, MLR treatment results into an empirical relation of the form

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

In this equation, a and the b_j's are **regression coefficients** and y' is the **response variable**. If $p = 1$, we have the special case of simple linear regression (SLR), for which Python/NumPy and R coding is introduced elsewhere [2, 3]. Applying equation 1 to the observations (actuals), we obtain response values y'_i with $i = 1, ..., n$. The residuals, which measure the deviation between corresponding fitted and observed values, are calculated as $e_i = y_i - y'_i$.

Employing R's `lm()` function [7, 8], we demonstrate in the following R-driven computation how to derive regression coefficients and associated statistical descriptor by using a published set of sample data.

Hands-on data

We use the data of Example 8.2 in [9] that were obtained by examining the packing conditions of ammonium sulfate. The flow rate of ammonium sulfate—an inorganic salt—was determined by letting defined quantities of salt flow through a small funnel. The flow rate is the dependent variable y, depending on salt characteristics such as moisture content, crystal shape anf impurities. In the example, the independent variables are: x_1 = initial moisture content in units of 0.01 %, x_2 = length/breadth ratio for crystals and x_3 = percent impurity in units of 0.01 %. Their values are listed in Table 1 in the Appendix. Also, the response values (calculated with equation 1) and the residuals are given there.

A CSV file with the data is ready for downloading:
www.axeleratio.com/math/comp/linreg/csv/woodward82.csv.

The regression coefficients are $a = 6.737$, $b_1 = -0.04882$, $b_2 = -0.56877$ and $b_3 = -0.16545$ (page 252 in [9]). The estimated residual variance is 0.7164, giving a residual standard error of $s = 0.8464$. The standard errors of the regression coefficients are $s_{b_1} = 0.0280$, $s_{b_2} = 0.2531$ and $s_{b_3} = 0.0509$ (page 254 in [9]).

MLR with `lm()` in R

The data can directly be loaded from the CSV file `woodward82.csv` at the *Axeleratio* location and structured into a data frame we name `woodward82`:

```
> fcsv <-
+ "http://www.axeleratio.com/math/comp/linreg/csv/woodward82.csv"
> woodward82 <- read.csv(fcsv, header=TRUE, sep=";")
```

The function `lm()` carries out the model building:

```
> linearmodel <- lm(y ~ x1 + x2 + x3, data = woodward82)
```

The model building instruction is `y ~ x1 + x2 + x3`. Testing models with less than the three independent variables could be done in the same way; using the symbolic model description `y ~ ~x1 + x2`, for example, to compute the regression coefficients and statistical descriptors for a model with two instead of three x variables. The `summary` function instructs R to display the results in the `linearmodel` object:

```
> summary(linearmodel)
```

The output for the model derived with three x variables is shown in the following:

```
Call:
lm(formula = y ~ x1 + x2 + x3, data = woodward82)

Residuals:
     Min       1Q   Median       3Q      Max
-1.78941 -0.63034 0.08036 0.48404 1.42551

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.73730   0.66697 10.101 4.89e-13 ***
x1          -0.04882   0.02798 -1.745  0.0880 .
x2          -0.56877   0.25306 -2.248  0.0297 *
x3          -0.16545   0.05087 -3.253  0.0022 **
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘’ 1

Residual standard error: 0.8464 on 44 degrees of freedom
Multiple R-squared: 0.5749,  Adjusted R-squared: 0.546
F-statistic: 19.84 on 3 and 44 DF, p-value: 2.777e-08
```

The output format and the displayed descriptors are the same as described for simple linear regression. The only difference is that the "Coefficients" section includes additional rows to account for multiple variables instead of a single one. You can individually access these values via the model summary, for which we use the variable `ms`. Then, for example, b_3 and s_{b_3} are obtained as follows:

```
> ms = summary(linearmodel)
> b3 = coef(ms)["x3","Estimate"]
> sb3 = coef(ms)["x3","Std. Error"]
```

Finally, we apply the model to two triples with new x values, (14, 2.6, 0) and (20, 2.1, 3). We put the values into a new data frame and use the `predict()` function:

```
> newdata <- data.frame(x1=c(14,20), x2=c(2.6,2.1), x3=c(0,3))
> predict(linearmodel,newdata)
       1        2
4.575021 4.070125
```

The two estimated y values are displayed in column 1 and 2 and match our control calculations when rounded to the fifth decimal:

$$1: \quad 6.73730 - 0.04882 \cdot 14 - 0.56877 \cdot 2.6 - 0.16545 \cdot 0 = 4.57502$$

$$2: \quad 6.73730 - 0.04882 \cdot 20 - 0.56877 \cdot 2.1 - 0.16545 \cdot 3 = 4.07013.$$

Conclusion & Outlook

The purpose here was to demonstrate how a MLR task can be approached with R. A dataset and a corresponding CSV file for testing was provided. No attempt was made to interpret MLR results or to investigate modeling alternatives.

In addition to the textbook already cited, chapter 4 and 5 in [4] provide an excellent introduction to MLR by showing in detail how to calculate regression coefficients and statistical descriptors. Further, online resources are available that address various aspects of R-based MLR analysis [1, 5, 6].

If your visual or statistical data analysis suggests a non-linear relationship between the variables of interest, you may want to consider applying curvilinear regression analysis as well.

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] John M. Quick at R-bloggers. R tutorial series: Multiple linear regression. https://www.r-bloggers.com/r-tutorial-series-multiple-linear-regression/. Accessed: 2019-03-04.

[2] Axeleratio Blog. Simple linear regression with Python and R: Getting started. https://axeleratio.blogspot.com/2019/02/simple-linear-regression-with-python.html. Accessed: 2019-03-03.

[3] Axel Drefahl. Simple linear regression with Python and R: three ways to begin with. http://www.axeleratio.com/math/comp/linreg/linregways.pdf. Accessed: 2019-03-03.

[4] A. L. Edwards. *Multiple Rgression and the Analysis of Variance*. W. H. Freeman and Company, New York, 2 edition, 1984.

[5] David J. Lilja. Linear regression using R - an introduction to data. http://www.axeleratio.com/math/comp/linreg/linregways.pdf. Edition 1.1 (April, 2017), accessed: 2019-03-03.

[6] QuickR. Multiple (linear) regression. https://www.statmethods.net/stats/regression.html. Accessed: 2019-03-04.

[7] RDocumentation. lm. https://www.rdocumentation.org/packages/stats/versions/3.5.2/topics/lm. Accessed: 2019-03-04.

[8] R Tutorial. Estimated multiple regression equation. http://www.r-tutor.com/elementary-statistics/multiple-linear-regression/estimated-multiple-regression-equation. Accessed: 2019-03-04.

[9] 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 8.2 in [9]. 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 as shown in section "MLR with (lm) in R." 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	$x_{1,i}$	$x_{2,i}$	$x_{3,i}$	y_i	y'_i	e_i
1	21	2.4	0	5.00	4.34703	0.65297
2	20	2.4	0	4.81	4.39585	0.41415
3	16	2.4	0	4.46	4.59113	-0.13113
4	18	2.5	0	4.81	4.43662	0.37339
5	16	3.2	0	4.46	4.13612	0.32388
6	18	3.1	1	3.85	3.92990	-0.07990
7	12	3.2	1	3.21	4.16595	-0.95595
8	12	2.7	0	3.25	4.61578	-1.36578
9	13	2.7	0	4.55	4.56696	-0.01696
10	13	2.7	0	4.85	4.56696	0.28304
11	17	2.7	0	4.00	4.37168	-0.37168
12	24	2.8	0	3.62	3.97306	-0.35306
13	11	2.5	0	5.15	4.77836	0.37165
14	10	2.6	0	3.76	4.77030	-1.01030
15	17	2.0	0	4.90	4.76982	0.13018

16	14	2.0	0	4.13	4.91628	-0.78628
17	14	2.0	1	5.10	4.75083	0.34917
18	14	1.9	0	5.05	4.97316	0.07684
19	20	2.1	2	4.27	4.23558	0.03442
20	12	1.9	1	4.90	4.90535	-0.00535
21	11	2.0	2	4.55	4.73184	-0.18184
22	10	2.0	7	5.32	3.95341	1.36659
23	10	2.0	2	4.39	4.78066	-0.39066
24	16	2.0	2	4.85	4.48774	0.36226
25	17	2.2	3	4.59	4.15972	0.43028
26	17	2.4	4	5.00	3.88051	1.11949
27	17	2.4	0	3.82	4.54231	-0.72231
28	15	2.4	2	3.68	4.30905	-0.62905
29	17	2.2	3	5.15	4.15972	0.99028
30	21	2.2	4	2.94	3.79899	-0.85899
31	23	2.2	10	3.18	2.70865	0.47135
32	22	2.0	7	2.28	3.36757	-1.08757
33	21	1.9	4	5.00	3.96962	1.03038
34	24	2.1	8	2.43	3.04760	-0.61760
35	37	2.3	14	0	1.30649	-1.30649
36	21	2.4	2	4.10	4.01613	0.08387
37	28	2.4	5	3.70	3.17804	0.52196
38	29	2.4	7	3.36	2.79832	0.56168
39	23	3.6	7	3.79	2.40872	1.38128
40	32	3.3	8	3.40	1.97452	**1.42548**
41	26	3.5	4	1.51	2.81549	-1.30549
42	28	3.5	12	0	1.39425	-1.39425
43	21	3.0	3	1.72	3.50942	**-1.78942**
44	22	3.0	6	2.33	2.96425	-0.63425
45	34	3.0	8	2.38	2.04751	0.33249
46	29	3.5	5	3.68	2.50358	1.17643
47	17	3.5	3	4.20	3.42032	0.77969
48	11	3.2	2	5.00	4.04932	0.95068