Lab: Linear Regression

Gavin Simpson

Feb 2017

Summary

In this lab you will be introduced to fitting linear regression models using R.

1 Simple linear regression

This is the simplest form of regression with a single response or dependent variable y and one predictor or independent variables x. The linear regression model is defined by:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

where α is a constant, the intercept, β is the slope and ϵ_i is the error component.

In this part of the practical class we will model the response of stomatal density (number of stomata mm^{-2} leaf) of modern leaves of *Salix herbacae* (dwarf willow) in relation to atmospheric CO_2 concentrations (ppm by volume, ppmv). Data for 29 collections of leaves growing at different CO_2 concentrations are in the text file `co2.txt`.

You will perform a simple linear regression using R. Load the data file `co2.csv` into the R object 'stomata'. Note that throughout it is **CO** — **cee-oh** — 2 and not **C0** — **cee-zero** — 2!

```
> stomata <- read.csv(file = "co2.csv")
> colnames(stomata)
> str(stomata)
```

The data contain 29 observations of two variables; `co2` is the CO_2 concentration and `sdensity` is the stomatal density. The first thing we should do is plot the data:

```
> co2.exp <- expression(CO[2])
> plot(sdensity ~ co2, data = stomata, xlab = co2.exp, ylab = "Stomatal density")
```

`expression()` creates a valid R expression out of its arguments which R then interprets rather than literally reproducing the contents as labels. So instead of the label being printed as CO[2], R interprets the brackets as meaning display this as subscript and formats the text accordingly. R has very powerful capabilities for displaying complex mathematical notation in plots. The code used to format this in a plot borrows heavily from LaTeX; see `?plotmath` for more information.

Linear models in R are specified using the `lm()` function and a standard formula notation that is used in a range of other R base functions and in many add-on packages:

$$response \sim predictor_1 + predictor_2 \ldots + predictor_i$$

To create a linear model of stomatal density as a function of CO_2 type the following:

```
> sdens.lm <- lm(sdensity ~ co2, data = stomata)
> summary(sdens.lm)

Call:
lm(formula = sdensity ~ co2, data = stomata)

Residuals:
    Min      1Q  Median      3Q     Max
-33.278  -8.566  -1.153  10.901  34.255

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) 294.9904    33.2477   8.873 1.73e-09 ***
co2          -0.6467     0.1153  -5.608 5.99e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.14 on 27 degrees of freedom
Multiple R-squared:  0.538,        Adjusted R-squared:  0.5209
F-statistic: 31.44 on 1 and 27 DF,  p-value: 5.985e-06
```

We then display the ANOVA table for the regression model we just specified. We used the `data` argument of `resid()` to tell R where to find the variables `sdensity` and `co2` instead of attaching the co2 data frame to the search path. This ensures that the variables being used are the ones you actually specify. The regression summary provides a succinct description of the model specified:

1. The call, which describes the model being summarised,
2. A description of the distribution of the residuals that allows one to quickly evaluate if these are normally distributed,
3. Regression coefficients, standard errors and their associated t-values and significance levels. The stars provide a quick visual guide to the level of significance achieved by each coefficient,
4. Finally the residual standard error, the multiple R-squared (R^2), R^2 adjusted for the degrees of freedom, the F statistic for the model and its associated p-value.

To display the fitted line from the regression first re-plot the data (you can use the up-cursor to move back through the commands you have previously typed, so you do not need to retype everything):

```
> plot(sdensity ~ co2, data = stomata, xlab = co2.exp,
+      ylab = "Stomatal density")
> abline(sdens.lm, col = "red")           # plots the fitted line
```

`abline()` is used for drawing lines on plots. It can be used to plot lines that fulfill many functions (we will use it to plot vertical lines on our diagnostics plots later), but when a linear model (of class `lm()`) is used as an argument, `abline()` extracts the slope and the intercept and plots a line accordingly.

To draw the 95% confidence limits and prediction intervals we need to make use of the function `predict()`. Enter the following code chunk:

```
> pred.frame <- data.frame(co2 = seq(min(stomata$co2), max(stomata$co2), length = 50))
> pp <- predict(sdens.lm, int = "p", newdata = pred.frame)
> pc <- predict(sdens.lm, int = "c", newdata = pred.frame)
> plot(sdensity ~ co2, data = stomata, xlab = co2.exp,
+      ylab = "Stomatal density", ylim = range(stomata$sdensity, pp))
> matlines(pred.frame$co2, pc, lty = c(1,2,2), col = "blue")
> matlines(pred.frame$co2, pp, lty = c(1,3,3), col = "red")
> identify(stomata$co2, stomata$sdensity)
```

Q and A

1. Label points on the plot which look like potential outliers or which lie outside the prediction intervals. Remember, after executing the `identify()` funtion you need to click on the plot window to label points, then right click the plot window to finish. You cannot use the R console to enter further commands until you do so—as indicated by the there not being a prompt “>” displayed.

This seems much more complicated than it really needs to be! But this illustrates a number of useful R commands. We want to predict stomatal density over the full range of CO_2, not just the discrete CO_2 values in our data. Also, when we plot lines using `plot(x, y, type = "l")` for example, R plots lines between the data points which would result in a cats cradle if the data points are not in order.

First we create a new data frame to hold the CO_2 values we wish to predict stomatal density for. We create a sequence of 50 equally-spaced values ranging from the minimum to the maximum value of CO_2 in our data. We use `predict()` to generate the prediction (`int = "p"`) and confidence (`int = "c"`) values, and instruct R to predict these values for our sequence of CO_2 data. Next comes the familiar plot command where we have added the argument `ylim` to make sure the full range of the prediction interval can be plotted. Finally we use `matlines()` to plot multiple lines from the provided arguments. Effectively we use `matlines()` to plot 3 lines with each call, plotting the confidence interval with dashed blue lines, the prediction interval with dotted red lines and the fitted line (which actually gets plotted twice) in red.‘

1.1 Regression diagnostics

Once we have fitted our linear model, we must examine that model to check for the influence of outliers and whether the assumptions of linear least squares are met by our data. The summary output provided a simple look at the distribution of the residuals from the regression, but we can do much better with some graphical displays.

Firstly, we should plot the residuals of the fitted model to look for patterns in the data or to assess whether the assumption of normally-distributed errors is maintained. The LOWESS smooth line shows the patterns in the data.

```
> plot(fitted(sdens.lm), resid(sdens.lm), main = "Residuals vs Fitted",
+      ylab = "Residuals", xlab = "Fitted values")
> abline(h = 0, lty = "dashed", col = "grey")
> lines(lowess(fitted(sdens.lm), resid(sdens.lm)), col = "red")
```

Another way of visualising whether the residuals of the models follow a normal distribution is to plot a quantile-quantile or Q-Q plot. The `identify()` function allow us to interactively label points in plots. Try the following:

```
> co2.resid <- resid(sdens.lm)
> co2.qq <- qqnorm(co2.resid)
> qqline(co2.resid)
> identify(co2.qq)
```

After executing the `identify()` command above, click with the left mouse button on some of the points in the Q-Q plot. Label those points that deviate somewhat from the 1:1 line in the plot. To finish labelling points press the right mouse button.

The `resid()` function is called an *extractor function* in R. The `lm()` model object contains a list of the residuals. We could access it directly by accessing the various components of the `sdens.lm` object. It is better practice, however, to use extractor functions to get at the parts of larger, more complex objects. Another example of an extractor function is `coef()`, which can be used to extract the regression coefficients from an `lm()` object.

To generate good predictive model that generalises well, it is important to determine whether the model is being unduly influenced by outlier values. We can look at a number of indicators that can allow us detect outliers that have a high degree of influence; hat values, DFBETAS and Cook's distance.

A useful guide for identifying outliers is to look for observations that have a hat value that is 2 or 3 times the average hat value. Enter the following commands and identify which leaves lie beyond these thresholds.

```
> plot(hatvalues(sdens.lm), type = "h", main = "Leverage: hat values",
+      xlab = "Observation", ylab = "Leverage")
> abline(h = c(2,3) * 2/29, lty = 2)
> identify(hatvalues(sdens.lm))
```

We use the `hatvalues()` function to calculate the hat values which we then plot as *histogram like* bars using the argument `type = "h"`. `abline()` is used to plot the horizontal lines at the given heights, `h`.

We can also display the DFBETAS for the model. Here, you should identify those observations that lie at a distance from the main cluster of points.

```
> plot(dfbetas(sdens.lm), type = "h", main = "Leverage: Dfbetas",
+      xlab = "Observation", ylab = "Dfbetas")
> abline(h = 0, col = "grey")
> identify(dfbetas(sdens.lm))
```

In a similar theme we can plot the Cook's distances for each observation. Again, identify those leaves that have a Cook's distance greater than the threshold value.

```
> plot(cooks.distance(sdens.lm), type = "h", ylim = c(0, 1),
+      main = "Cook's distance", xlab = "Observation",
+      ylab = "Cook's distance")
> abline(h = 4/27, lty = 2) # 4/(n-k-1) n = 29, k = 1 parameter in model
> identify(cooks.distance(sdens.lm))
```

We use 4/27 from the formulae $4/(n-k-1)$, where $n = 29$, $k = 1$ parameter model.

Q and A

1. By now you should have enough information to answer the following question. Which of the leaves is a potential outlier?

Having done all the hard work of generating regression diagnostic plots in R, it is worth noting that R comes with a plot method for linear models that produces some of the diagnostic plots we produced by hand. The following code produces a 2 x 2 grid of plots on the open device (window), draws the diagnostic plots and then resets the plotting region to 1 x 1.

```
> layout(matrix(1:4, ncol = 2))
> plot(sdens.lm, ask = FALSE)
> layout(1)
```

To find out more about the lower left plot, look at the help for `plot.lm()` by typing `?plot.lm` at the prompt.

In the `co2.txt` data file one of the leaves (leaf 17, CO_2 of 206 ppmv) is a full-glacial age sample matched against the Byrd ice-core CO_2 measurements for 15 800–18 900 years BP. Generate another linear regression model of stomatal density against CO_2 concentration, this time omitting leaf 17 from the analysis. Look to see if the intercepts and slopes differ from those obtained above, and take note of the standard errors of the regression coefficients.

```
> sdens.lm2 <- lm(sdensity ~ co2, data = stomata[-17,])
> summary(sdens.lm2)
```

Refer back to the instructions on how to plot the data with a fitted line if you wish to visually inspect the differences in the two models.

2 Multiple linear regression

So far we have seen how to fit simple linear regression models with a single predictor variable and a single response variable. In this section of the practical we will take a look at multiple regression models in R, where we have a single response variable with two or more predictor variables we wish to model.

The data we'll be using are from Loyn (1987), wherein 56 forest patches in southeastern Victoria, Australia, were selected and the abundance of birds within each patch was related to siz predictor variables:

1. `area`: patch area (ha)
2. `dist2Patch`: distance to the nearest patch (km)
3. `dist2LargePatch`: distance to the nearest large patch (km)
4. `grazing`: grazing stock on a 1–5 ordinal scale (light–heavy)
5. `altitude`: altitude of patch (m)
6. `yearIsolated`: year of isolation

Three of these variables (`area`, `dist2Patch`, and `dist2LargePatch`) are higly skewed and hence we apply a $\log_{10}$ transformation.

You will use R to explore these data and to fit a multiple linear regression model of the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m$$

where $x_1 \ldots x_m$ are our predictor variables and β_0, $\beta_1 \ldots \beta_m$ are parameters that we wish to estimate for combinations of y and x_m.

Read in the data from `loyn.csv`, display it and plot a scatter-plot matrix of the variables using the following commands.

```
> birds <- read.csv("loyn.csv")
> birds
> plot(birds, gap = 0)
```

It is useful to have a numerical summary of the information shown in a scatter-plot matrix, and it is easy to generate a correlation matrix using the R function `cor()`. `cor()` does not generate p-values or significance values for the correlations however, and the associated function `cor.test()` only works on two vectors, x and y at a time. Our function[1] `corProb()` will generate the lower triangle of a correlation matrix with associated p-values shown in the upper triangle. Load the `corProb()` function and generate the correlation matrix.

```
> source("corProb.R")
```

Then we use it by passing in the `birds` data frame

```
> corProb(birds)

Correlations are shown below the diagonal
P-values of the F-statistics are shown above the diagonal

                abundance area    yearIsolated dist2Patch dist2LargePatch
abundance        1.0000    0.0569  0.0001       0.0798     0.5230
area             0.2560    1.0000  0.9913       0.4267     0.8003
yearIsolated     0.5034   -0.0015  1.0000       0.4061     0.5415
dist2Patch       0.2361    0.1083  0.1132       1.0000     0.0172
dist2LargePatch  0.0872    0.0346 -0.0833       0.3172     1.0000
grazing         -0.6825   -0.3104 -0.6356      -0.2558    -0.0280
altitude         0.3858    0.3878  0.2327      -0.1101    -0.3060
                grazing altitude
abundance        0.0000  0.0033
area             0.0199  0.0032
yearIsolated     0.0000  0.0843
dist2Patch       0.0570  0.4192
dist2LargePatch  0.8376  0.0218
grazing          1.0000  0.0018
altitude        -0.4072  1.0000
```

Multiple linear regression models are fitted in R in the same way as the simple linear regression model you looked at earlier. To fit the full multiple regression model using all seven predictors use the following commands.

```
> birds.lm <- lm(abundance ~ log10(area) + log10(dist2Patch) + log10(dist2LargePatch) +
+                grazing + altitude + yearIsolated, data = birds)
```

Note that the `log10()` parts are applying the transformation to the variables as part of the model fitting. This transformation is part of the model fit now so when predicting for new data those data will automatically get the same transformation applied to them. Look at the model summary:

```
> summary(birds.lm)

Call:
lm(formula = abundance ~ log10(area) + log10(dist2Patch) + log10(dist2LargePatch) +
    grazing + altitude + yearIsolated, data = birds)

Residuals:
     Min       1Q   Median       3Q      Max
-15.6506  -2.9390   0.5289   2.5353  15.2842

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)           -125.69725   91.69228  -1.371   0.1767
log10(area)              7.47023    1.46489   5.099 5.49e-06 ***
log10(dist2Patch)       -0.90696    2.67572  -0.339   0.7361
log10(dist2LargePatch)  -0.64842    2.12270  -0.305   0.7613
grazing                 -1.66774    0.92993  -1.793   0.0791 .
```

[1]The `corProb()` function is based on code published on the R-Help mailing list by Bill Venables which has been modified to provide a print method.

```
altitude                  0.01951    0.02396   0.814   0.4195
yearIsolated              0.07387    0.04520   1.634   0.1086
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.384 on 49 degrees of freedom
Multiple R-squared:  0.6849,        Adjusted R-squared:  0.6464
F-statistic: 17.75 on 6 and 49 DF,  p-value: 8.443e-11
```

Note the values for R^2, adjusted-R^2, F and its p for this, the full model.

Q and A

1. What do these results suggest to you about the relationship between the predictor variables and the abundance of birds in forest patches?

We can also produce an ANOVA table using R's `anova()` function, which shows the marginal effect of adding each variable in turn to the model. Each variable is added to the model in the order that you specified in the right-hand side of the model equation or in the order they appear in the data frame or matrix from which the predictors were derived.

```
> anova(birds.lm)

Analysis of Variance Table

Response: abundance
                       Df Sum Sq Mean Sq F value    Pr(>F)
log10(area)             1 3471.0  3471.0 85.1731 2.695e-12 ***
log10(dist2Patch)       1   65.5    65.5  1.6067 0.2109423
log10(dist2LargePatch)  1  136.5   136.5  3.3500 0.0732901 .
grazing                 1  543.6   543.6 13.3394 0.0006316 ***
altitude                1   15.7    15.7  0.3841 0.5382675
yearIsolated            1  108.8   108.8  2.6705 0.1086351
Residuals              49 1996.9    40.8
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Q and A

1. What do these results suggest to you about the relationship between the predictor variables and the abundance of birds in forest patches?
2. Which variables are candidates for removal?

We can remove seemingly non-informative variables by updating the model fit and indicating which variables are to be removed from the model. Here we remove some variables and then generate the model summary and the ANOVA table coparing the full model (`birds.lm`) with the simplified model (`birds2.lm`)

```
> birds2.lm <- update(birds.lm, . ~ . - log10(dist2Patch) - log10(dist2LargePatch) -
+                     altitude)
> summary(birds2.lm)

Call:
lm(formula = abundance ~ log10(area) + grazing + yearIsolated,
    data = birds)

Residuals:
     Min       1Q   Median       3Q      Max
-14.5159  -3.8136   0.2027   3.1271  14.5542

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  -134.26065   86.39085  -1.554   0.1262
log10(area)     7.16617    1.27260   5.631 7.32e-07 ***
grazing        -1.90216    0.87447  -2.175   0.0342 *
yearIsolated    0.07835    0.04340   1.805   0.0768 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.311 on 52 degrees of freedom
Multiple R-squared:  0.6732,        Adjusted R-squared:  0.6544
F-statistic: 35.71 on 3 and 52 DF,  p-value: 1.135e-12
```

```
> anova(birds.lm, birds2.lm)

Analysis of Variance Table

Model 1: abundance ~ log10(area) + log10(dist2Patch) + log10(dist2LargePatch) +
    grazing + altitude + yearIsolated
Model 2: abundance ~ log10(area) + grazing + yearIsolated
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     49 1996.8
2     52 2071.1 -3   -74.286 0.6076 0.6132
```

2.1 Interaction terms

Thus far we have only considered the unique contributions of each variable to the modelling of bird abundance. Perhaps the effect on bird abundance of one variable depends on the value(s) of one or more other variables in the model. We can account for such *interactions* between variables via interaction terms. In the next code chunk we add 2-way (2nd order) and the 3-way (3rd-order) interactions between variables in the simplified model.

```
> birds3.lm <- update(birds2.lm, . ~ . + log10(area):grazing + log10(area):yearIsolated +
+                     grazing:yearIsolated + log10(area):grazing:yearIsolated)
> anova(birds2.lm, birds3.lm)

Analysis of Variance Table

Model 1: abundance ~ log10(area) + grazing + yearIsolated
Model 2: abundance ~ log10(area) + grazing + yearIsolated + log10(area):grazing +
    log10(area):yearIsolated + grazing:yearIsolated + log10(area):grazing:yearIsolated
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     52 2071.1
2     48 1542.9  4    528.25 4.1085 0.006076 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Q and A

1. Do we need the 3-way interaction?

What about the 2-way terms?

```
> birds4.lm <- update(birds3.lm, . ~ . - log10(area):grazing:yearIsolated)
> anova(birds3.lm, birds4.lm)

Analysis of Variance Table

Model 1: abundance ~ log10(area) + grazing + yearIsolated + log10(area):grazing +
    log10(area):yearIsolated + grazing:yearIsolated + log10(area):grazing:yearIsolated
Model 2: abundance ~ log10(area) + grazing + yearIsolated + log10(area):grazing +
    log10(area):yearIsolated + grazing:yearIsolated
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     48 1542.9
2     49 1546.4 -1   -3.4888 0.1085 0.7432
```

```
> drop1(birds4.lm)

Single term deletions

Model:
abundance ~ log10(area) + grazing + yearIsolated + log10(area):grazing +
    log10(area):yearIsolated + grazing:yearIsolated
                        Df Sum of Sq    RSS    AIC
<none>                                1546.4 199.83
log10(area):grazing       1    334.74 1881.1 208.80
log10(area):yearIsolated  1    239.74 1786.1 205.90
grazing:yearIsolated      1    272.21 1818.6 206.91
```

Q and A

1. Do we need any of the 2-way interactions?