Exercises: Week 5. Regression

Bob O'Hara

4 February 2018

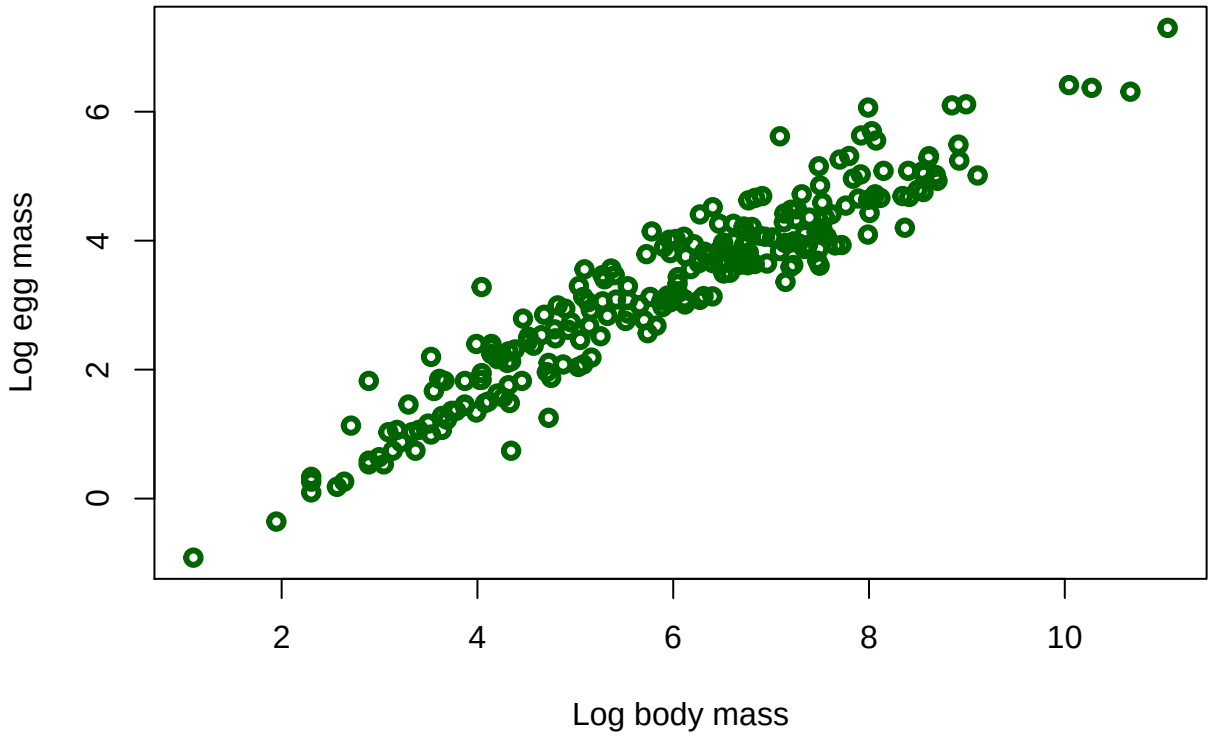
Code Workthrough

This is the analysis of the data from the lectures, on the sizes of birds and their eggs. It presents the code to run regressions and look at the results.

First, we read in the data (with read.csv()),and then plot it (the par() function sets some parameters for the plot: in this case the margin sizes). You will have to download the data from Blackboard (unless I can get access to the wiki!), and then put it in the working directory (use getwd() in R to check where that is, and setwd() if you want to change it).

```
BirdEggs <- read.csv(file="BirdEggs.csv", stringsAsFactors = FALSE)

par(mar=c(4.1,4.1,1,1))
plot(BirdEggs$logFemaleMass, BirdEggs$logEggMass,
     xlab="Log body mass", ylab="Log egg mass",
     main="", lwd=3, col="darkgreen")
```



We should look at this to see if (a) a linear model makes sense, and (b) if there are likely to be any problems (e.g. there might be a couple of species with egg sizes that are too large or too small, but nothing severe). We can make a mental note to check these later.

Now we fit the model. The R function to fit a regression model is lm(). The main part of this is the formula, which uses a notation that is more convenient than writing out the equations. If we want to regress Y against X (i.e. fit the model $Y_i = \alpha + \beta X_i + \varepsilon_i$), then we write this as Y ~ X. This will get more complicated later (e.g. if we want to regress Y against X and Z then the model is written Y ~ X + Z).

```
mod <- lm(logEggMass ~ logFemaleMass, data=BirdEggs)
```

We now have an R object, mod, which has enough information about the model. if you want to, you can use str(mod) to see what it contains, but you usually shouldnt't have to go there.

So, what is the model? We can get the coefficients with the coef() function, and the residual variance ($\hat{\sigma}^2$) with sigma():

```
round(coef(mod), 2)
```

```
##   (Intercept) logFemaleMass
##         -1.25          0.76
```

```
round(sigma(mod), 2)
```

```
## [1] 0.46
```

I have used round() to give the results to 2 decimal places: if you report results to 7 decimal places (for example), it is harder for the reader to read and interpret the results. This is especially bad if the reader is marking your work. So, always think about not presenting too maky significant figures! Anyway, the model is

$$\mu_i = -1.25 + 0.76 x_i + \varepsilon_i$$

$$y_i \sim N(\mu_i, 0.46)$$

We can also get confidence intervals with confint(), which show that, for example, the slope is fairly well estimated (i.e. the confidence interval is not too wide)

```
round(confint(mod), 2)
```

```
##                2.5 % 97.5 %
## (Intercept)    -1.45  -1.05
## logFemaleMass   0.73   0.79
```

That was (hopefully) not painful. But it is obviously not enough. We can get a summary of the model with the, um, summary() function:

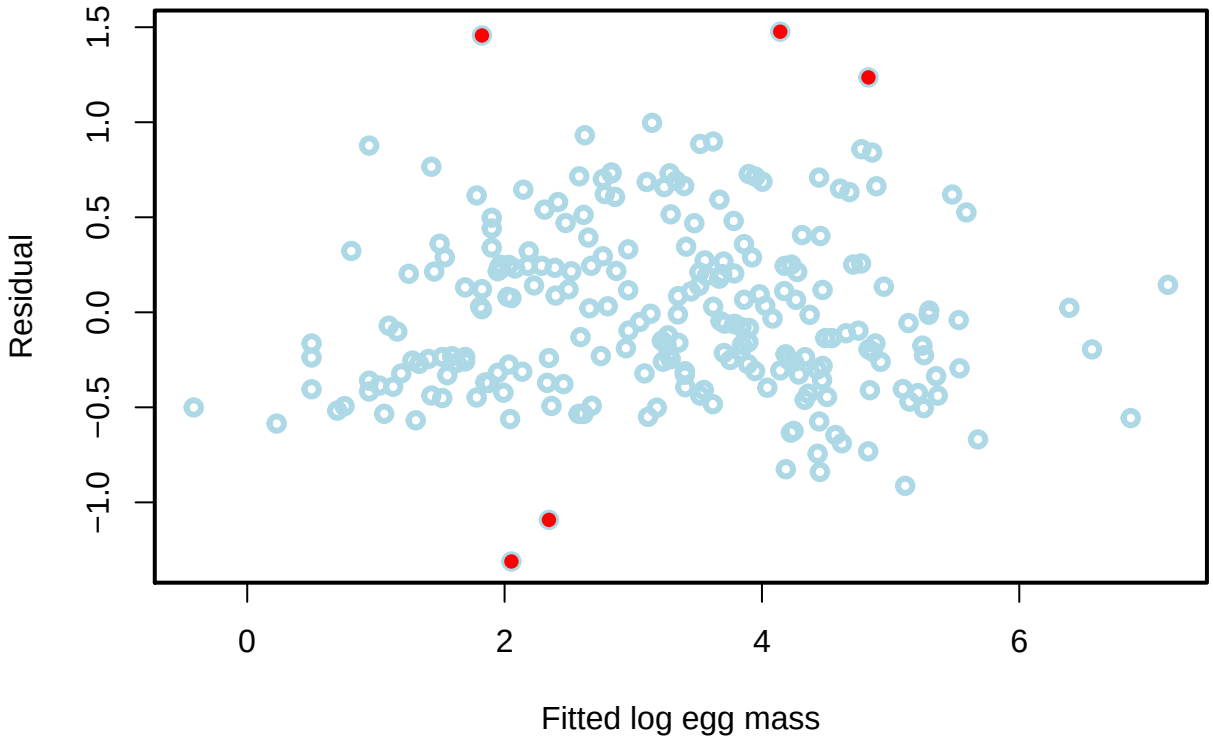
```
summary(mod)
```

```
##
## Call:
## lm(formula = logEggMass ~ logFemaleMass, data = BirdEggs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31116 -0.32185 -0.06145  0.25111  1.47630
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.25139    0.10174  -12.30   <2e-16 ***
## logFemaleMass   0.76074    0.01645   46.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4554 on 235 degrees of freedom
## Multiple R-squared:  0.9009, Adjusted R-squared:  0.9005
## F-statistic:  2137 on 1 and 235 DF,  p-value: < 2.2e-16
```

The Call is just the function that is being summarised. The Residuals is a summary of the residuals. The next part - the coefficients - is more interesting. It gives the coefficients and their standard errors, plus a test that can give you some nice stars. The final block has a few useful statistics: the residual standard error, and the R^2, so we can see that logFemaleMass explains about 90% of the variation in the data: this is a good model, as it explains most of the variation. In practice, the summary() is usually one of the first things to look at, as it tells you the parameter values (so we can eyeball them to see which are large), and a summary of fit, through the R^2.

Now we have the model, and an idea of what it looks like, we can check how well it fits. First, plot the residuals against the fitted values:

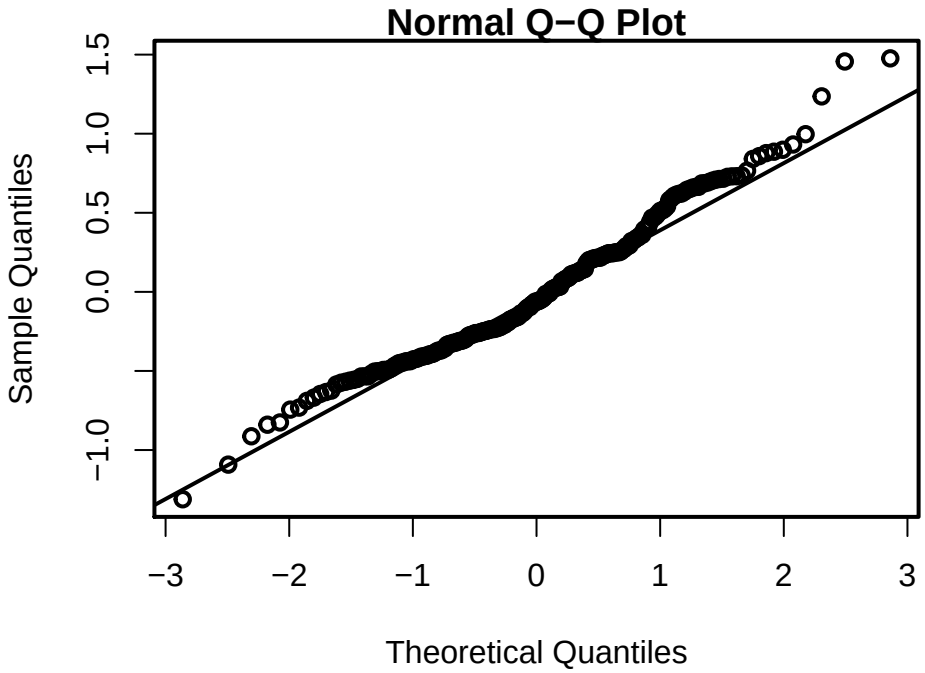
```
# highlight if the residual is above 1.2 or less than -1.0
HighlightPoints <- resid(mod) > 1.2 | resid(mod) < -1
par(mar=c(4.1,4.1,1,1), lwd=2)
plot(fitted(mod), resid(mod),
     xlab="Fitted log egg mass", ylab="Residual",
     main="", lwd=3, col="lightblue")
points(fitted(mod)[HighlightPoints], resid(mod)[HighlightPoints], col=2, pch=16)
```



Here fitted(mod) gets the fitted values, and resid(mod) gets the residuals. We can see that the model looks OK: there may be some downward curvature, and there are a few points at the bottom and top that might be a problem: these have been highlighted in red (note how I did this: HighlightPoints is a logical variable that say whether the points should be highlighted. When I plot them, fitted(mod)[HighlightPoints] only plots the points where HighlightPoints is TRUE. If you want to play with coding R, try to work out how to colour them without this extra function, i.e. by using different coloiurts for different points in the plot()). Overall, I think this would be fine.

We can look at a normal probability plot:

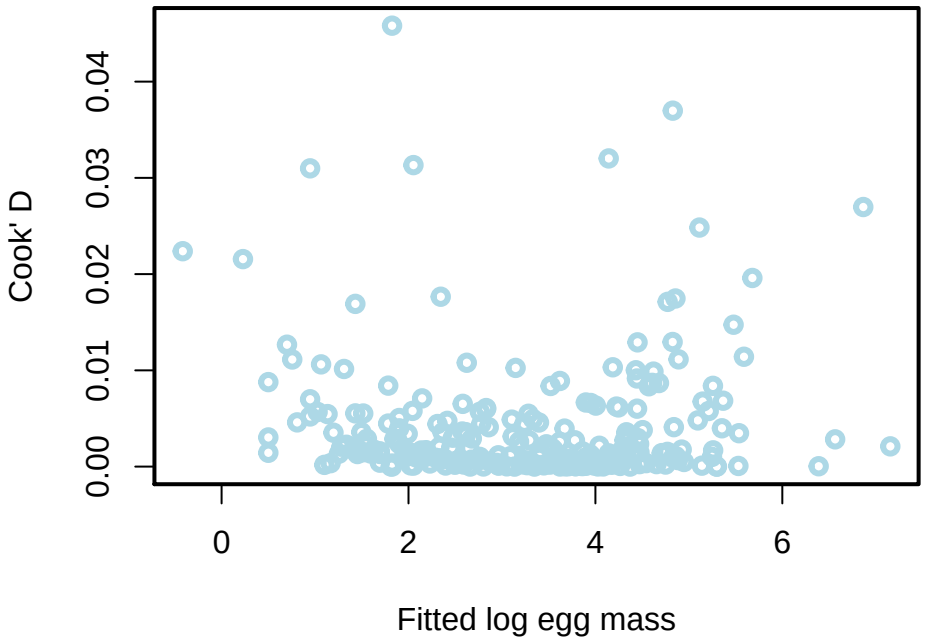
```
par(mar=c(4.1,4.1,1,1), lwd=2)
qqnorm(resid(mod))
qqline(resid(mod))
```



The second line adds a line, so we can see what we would expect. This does not look perfect: it's reasonably straight, but there might be a couple of outliers at the top.

We can also look to see if there are any influential observations, by plotting Cook' D:

```
par(mar=c(4.1,4.1,1,1), lwd=2)
plot(fitted(mod), cooks.distance(mod),
     xlab="Fitted log egg mass", ylab="Cook' D",
     main="", lwd=3, col="lightblue")
```



Basically, all of the values are small, so there is nothing influential.

R has a plot.lm() function for linear models that plots a few useful plots, so you can use ‘ `r plot(mod,`

which=1:6)‘ to see them all (which=1:6 controls which plots you see: some are more useful than others). When we do this, we see that R marks out a few points that might be worth taking another look at: points 24, 25, 71, 232. In the interst of completeness we can look at these, but first add the residuals to the data:

```
BirdEggs$Resid <- resid(mod)
BirdEggs[c(24, 25, 71, 232),]
```

```
##               species logEggMass logFemaleMass     Resid
## 24  Apteryx australis  6.0637852      7.991254  1.235944
## 25     Apteryx owenii  5.6185876      7.090077  1.476304
## 71  Cuculus saturatus  0.7419373      4.343805 -1.311159
## 232         Tyto alba  3.2809112      4.043051  1.456609
```

The first two (24 & 25) are Kiwis, which we know have large eggs. The third is a cuckoo, which has a smaller egg size than expected (there could be a biological explanation for this: see https://doi.org/10.1093/beheco/arg104 for much more). The last is the barn owl, which is a bit surprising. We can take a look at species with a similar size:

```
# AboutSameSizeAsBarmOwl is logical= it is TRUE if logFemaleMass > 4.0 AND logFemaleMass < 4.1
AboutSameSizeAsBarmOwl <- BirdEggs$logFemaleMass < 4.1 & BirdEggs$logFemaleMass > 4.0
BirdEggs[AboutSameSizeAsBarmOwl,]
```

```
##                    species logEggMass logFemaleMass      Resid
## 53   Caprimulgus vociferus   1.945910      4.043051  0.1216082
## 65  Coccyzus erythropalmus   1.840550      4.025352  0.0297124
## 120        Merops apiaster   1.840550      4.043051  0.0162477
## 169  Picoides albolarvatus   1.481605      4.077537 -0.3689323
## 232              Tyto alba   3.280911      4.043051  1.4566093
```

Note first that two have exactly the same size as a barn owl (can you work out why?). One of these is the European bee-eater, the other is a whip-poor-will. The other species are a woodpecker and a cuckoo. if you are a birder, you might be surprised at this: barn owls certainly look much bigger! According to Wikipedia, barn owls weigh 430 – 620 g, so they are about 10 times heavier. We can check to see what would happen if we used a better estimate of its size (500g, which is about right) using predict(). We need to give R a data frame with some new data in it:

```
BarnOwl <- data.frame(logFemaleMass=log(500))
predict(mod, newdata=BarnOwl, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 3.476283 2.577195 4.375371
```

So, the log egg mass from the data (3.28) is close to the fitted value. This is, then, presumably a typo (it is actually in the table I got this from: I did check!), and should be corrected. As far as the modelling goes, the other species are fine: they are not so extreme that they will affect the model, and they seem to be “real” values.

Problem 1

We know that larger mammals tend to have bigger brains, but also that humans have larger brains than expected. We argue that this is because we are more intelligent than other animals. If that is so, then we might expect a similar pattern to emerge in birds, where parrots and corvids are more intelligent than other birds.

We can dowmload the data from the web, or from Blackboard. (if you download it from blackboard,

```
BirdBrains <- read.csv('BirdBrains.csv')
```

to get it in to R, but). If the link below does not work, go here: http://onlinelibrary.wiley.com/doi/10.1002/ece3.2961/full#ece32961-sup-0001 and use the link for ece32961-sup-0001-TableS1.xlsx (i.e. copy the link, and use that as the Link variable). Note that in the un-transformed data, the brain and body mass are both reported in grams

```
# the gdata package has the read.xls function, which can read Excel files.
#    Just treat read.xls like read.csv.
library(gdata)
Link <- "http://onlinelibrary.wiley.com/store/10.1002/ece3.2961/asset/supinfo/ece32961-sup-0001-TableS1
BirdBrains <- read.xls(Link, stringsAsFactors=FALSE)
names(BirdBrains) <- gsub("\\.\\..*", "", names(BirdBrains)) # tidy up names
BirdBrains$logBodyMass <- log(BirdBrains$Body.mass)
BirdBrains$logBrainMass <- log(BirdBrains$Brain.mass)
```

We want to look at the relationship between the log of body size and log of brain size. Adapt the code above to answer the following questions: 1. What is the relationship between log of body size and log of brain size? What is the effect of increasing body size? Does brain size increase proportionally (e.g. if you doble body size, does brtain size double?)? 2. How good is the model fit? Does it look like there any problems with the fit, e.g. outliers, curavture in the data? 3. For the parrots (Order Psitacciformes), predict what their brain size would be if they were normal birds. Compare the predicted and real values - does the distribution of their brain sizes look hte same as other birds?

Problem 2

We can simulate some bad data and look at what happens. To simulate some "good" data, we can do this:

```
# define the parameters
alpha <- 5
beta <- 5
sigma <- 1 # standard deviation

# simulate x, 100 samples from a uniform distribution between 0 and 1
x <- runif(100, 0, 1) # random uniform distribution
# calculate E(y) using the defined values of alpha & beta, and the simulated values of x
mu <- alpha + beta*x
# simulate y, with mean mu and standard deviation sigma
y <- rnorm(length(mu), mu, sigma)
```

So, fit the model to the data and take a (quick) look at the model fit. Then, we can simulate "bad" models:

$$y_i = \alpha + \beta x_i^2 + \varepsilon_i$$

$$y_i = (\alpha + \beta x_i^2 + \varepsilon_i)^2$$

For each of these bad models: 1. simulate a data set, by modifying the code above, and then fit a linear model and look at the residuals. 2. Try to find a "good" Box-Cox transformation (https://www.isixsigma.com/tools-templates/normality/making-data-normal-using-box-cox-power-transformation/) on each of these, and see how it performs. Try a few Box-Cox transformations, and look at how they behave: look at the residuals to see how well the model fits to the data.

The MASS package in R has a boxcox() function if you want to find an optimum transformation, but also try a few yourself, so you can see what it happening.

Problem 3

I briefly showed the data on healthcare expenditure and life expectancy. The data is available online (see the link below), and also on Blackboard. Your task is to try to model this using simple linear regression and transformations, so that you can predict what the life expectancy of US citizens should be. So:

- Plot the data, and decide what sort of transformation of the data (x and/or y) might work.
- Try this transformation and fit the model. Have a look at the model and decide if it is reasonable, nad then if the model is fitting well: are there outliers? Are the residuals random or (for example) curved?
- Based on what you have learned in part 2, adjust the model. Do you need a different transformation? Are there outliers that might be making the model worse (and if so, what happens if you remove them)? Try a new model, and again look at how well it fits.
- You might want to try a few models, so record what you did, and then present your final model. I do not expect it to be perfect, so explain any problems you think might remain.

The code for getting the data from the web is a bit fiddly, but this is what real data analysis is like! So it is probably easiest to download it from Blackboard into your working directory, and read it from there:

```
rawdata <- read.csv("LifeExpectancy.csv")
```

Alternatively, if you want to get the data from the web, use this. I have commented out the code to plot the data - you can do that yourselves. You may have to install the RCurl and XML packages.

```
library(RCurl)
```

```
## Loading required package: bitops
```

```
library(XML)
# First we need to extract the table from the webpage. This is a bit tricky.
fileURL1 <- "https://www.ineteconomics.org/perspectives/blog/"
fileURL2 <- "the-link-between-health-spending-and-life-expectancy"
fileURL3 <- "-the-us-is-an-outlier"
fileURL <- paste0(fileURL1, fileURL2, fileURL3)
xData <- getURL(fileURL, ssl.verifyPeer=FALSE)
rawdata <- readHTMLTable(xData, stringsAsFactors=FALSE)[[1]]
# Now we need to format the data extracted from the html page
names(rawdata) <- gsub("\n", "", names(rawdata))
names(rawdata) <- gsub("\r", "", names(rawdata))
names(rawdata) <- gsub("\t", "", names(rawdata))
rawdata$`Life exectancy` <- as.numeric(rawdata$`Life exectancy`)
rawdata$`Health Spending per capita` <-
  as.numeric(gsub(",", "", gsub(" \\$", "",
                                rawdata$`Health Spending per capita`)))
```