linear_model_script

Kevin Healy

2023-10-25

When we have 2 or more continuous variables that we often want to know how they are related to one another. For example, we might want to know whether larger values of body mass are associated with longer lifespans.

In this script we will look at some basic correlation tests before running some linear models which consists of fitting lines to two or more continuous variables.

Here we will test the relationship between penguin body mass and flipper length. Our hypothesis will be that larger penguins have longer flippers. The NULL is that there is no difference in flipper length in penguins of different size.

Data

We will need to upload the penguin data set for this so set your directory to the folder with the downloaded dataset called penguin_1_10_2023.csv

```
pen_data <- read.csv("penguin_1_10_2023.csv")
```

Data cleaning

If you look at the dataset you might notice some NA values. These are missing values which were either not recorded or not input in the original dataset. These are normal but we should deal with them before we start our analysis.

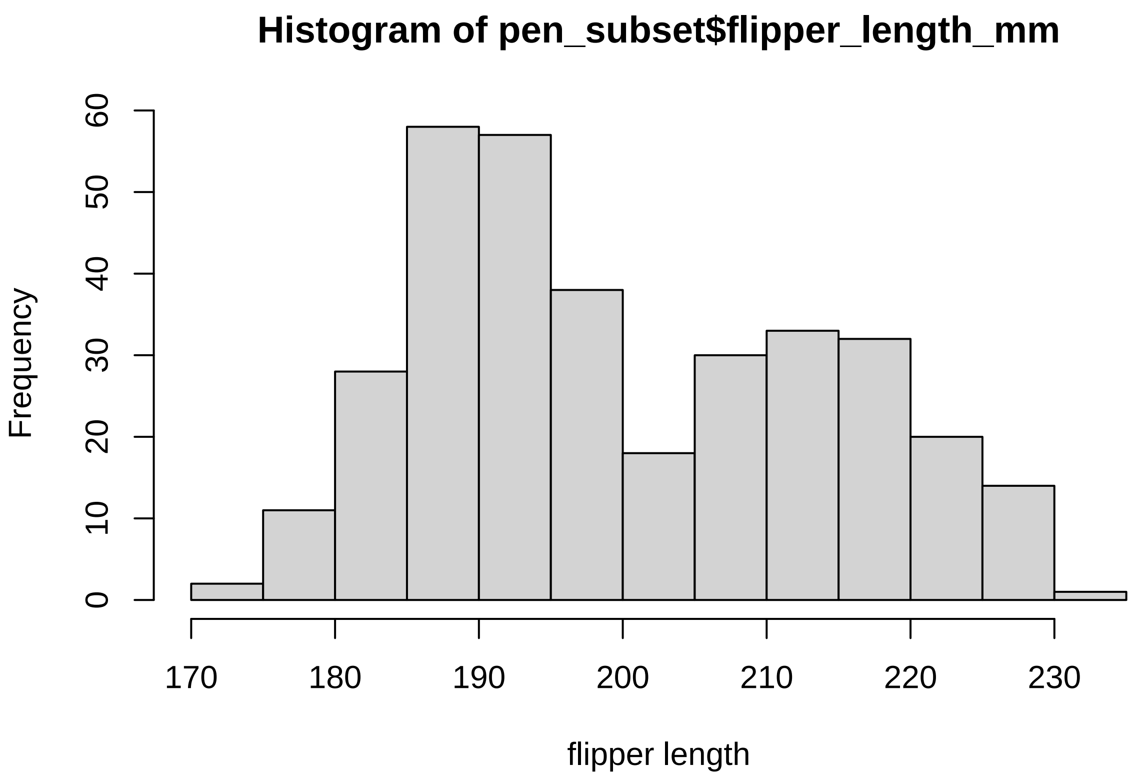
Here we will use the na.omit() function which will drop every row with an NA. We should be careful with this as if a row has an NA for any column it will be dropped even if we don't plan on using the column with the NA. To avoid this we will create a subset of the variables we want in our analysis. In this case lets just use species, flipper_length_mm, body_mass_g and sex. First we will subset and then we will remove all the rows with NAs.

```
#Subset the data
pen_subset <- data.frame(species = pen_data$species,
                         flipper_length_mm = pen_data$flipper_length_mm,
                         body_mass_g = pen_data$body_mass_g,
                         sex = pen_data$sex)

#remove rows with NAs in them
#Notice I overwrote pen_subset after removing the na, this is fine but be
#careful and avoid overwriting the original data you imported.

pen_subset <- na.omit(pen_subset)
```

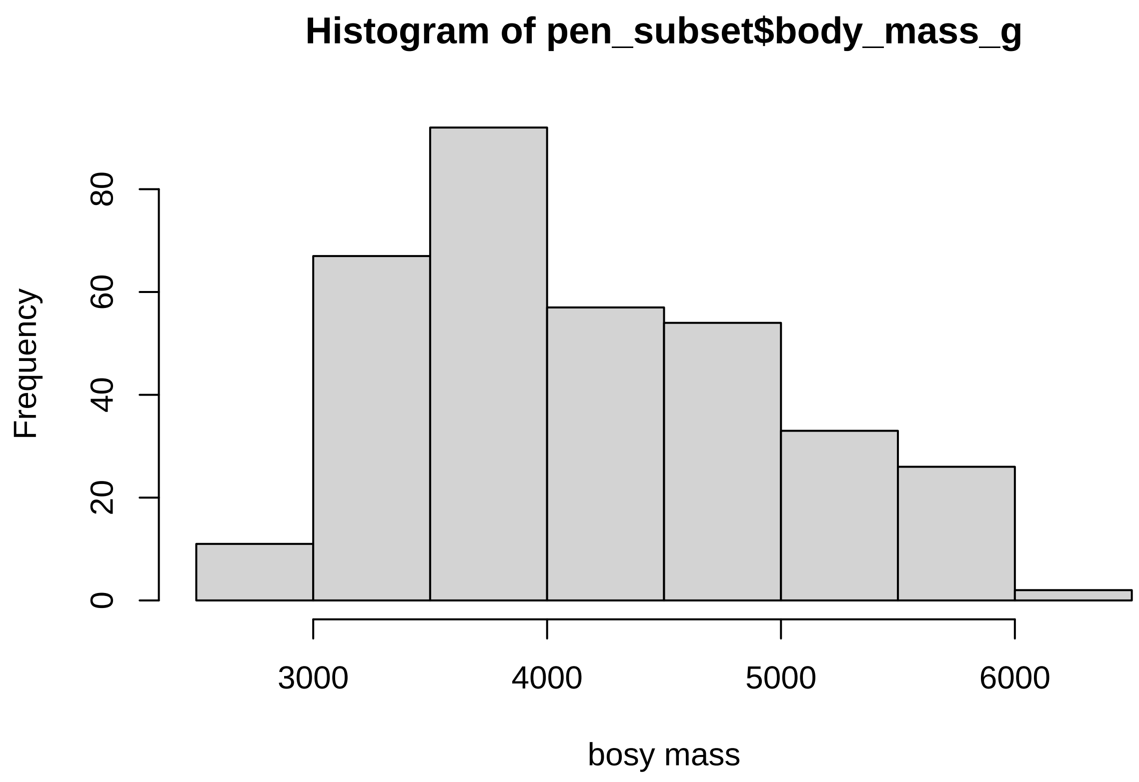
Plotting

One of the first things you should always do before running an analysis is to plot your data. This will give you an idea of not only what relationships might be there but also a feel for what the distributions of the data look like. Lets first do a histogram of each of the continuous variables.

```
hist(pen_subset$flipper_length_mm,
     xlab = "flipper length")
```



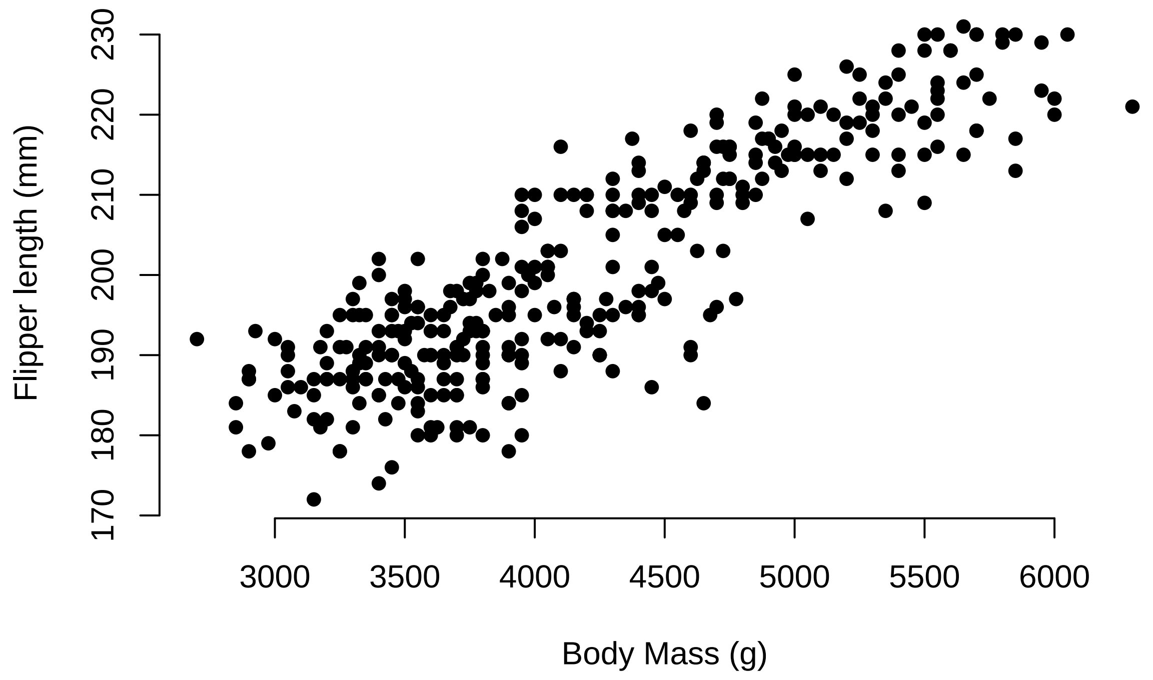
```
hist(pen_subset$body_mass_g,
     xlab = "bosy mass")
```



While body mass looks normally distributed flipper length looks bimodal. Thats ok for now but something we will keep in mind and explore later.

Next we will use a scatter plot to look at how the two continuous variables relate to one another. In our hypothesis we state that we think larger penguins have longer flippers so we think being larger is what leads to having longer flippers and not the other way around. This means body size is our explanatory variable while flipper length is the response variable. The explanotory varaible will always go on the x-axis with the response on the y-axis.

```
plot(pen_subset$flipper_length_mm ~ pen_subset$body_mass_g,
     pch = 16,
     xlab = "Body Mass (g)",
     ylab = "Flipper length (mm)",
     bty = "n")
```



From the plot we can see a clear trend, with bigger individuals having longer flippers. However, we can check using models whether this is a significant trend.

Pearson correlation test

The Pearson correlation coefficient (r) is a common way of measuring a linear correlation. It is a number between -1 and 1 that measures the strength and direction of the relationship between two variables. It works by calculating how close the residuals are to the best line of fit.

We can use the cor() function R. Notice this function reads in the variables as cor(X-axis, Y-axis).

```
cor_test <- cor(pen_subset$body_mass_g,
                pen_subset$flipper_length_mm)
cor_test
```

```
## [1] 0.8712018
```

The value of 0.87 indicates that there is a strong positive relationship between.

We could also estimate how the fit of this correlation, if its a good fit it would mean the data cluster very tightly together into a line, if the fit is poor the data will be spread out more like a cloud. One common measure of this is the r-squared value. It is the square of the correlation value but can be thought of as a measure of how far away the data are from a fitted line which we will see later. It has a value between 0 (completely random data) to 1 (data completely fit along some line or model). Lets calualte it here.

```
cor_test^2
```

```
## [1] 0.7589925
```

This is a high R-squared so it indicates a good fit for the correlation. We will see this value again later in the linear model.

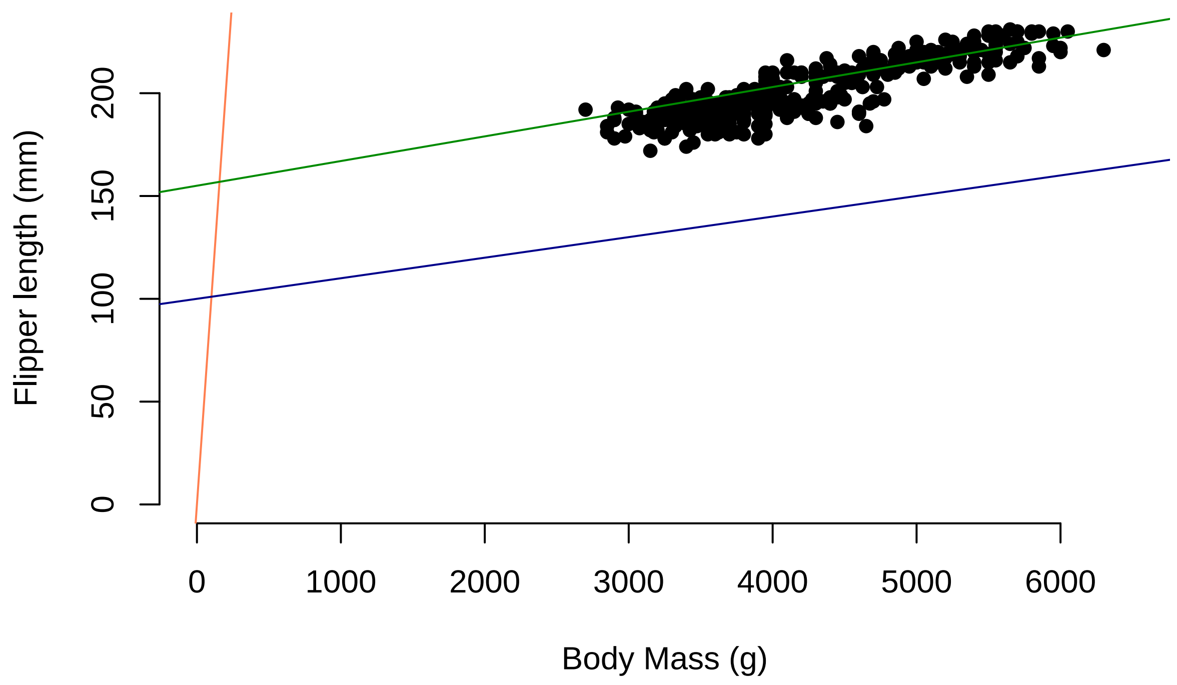
Overall these correlation values are very limited in that it does not tell us how much flipper length changes with body size. For example, I might want to know how much longer I should expected a flipper to be if I increase the body mass of a penguin by one gram. To do that we need to use linear models

linear models

Linear models work by trying to find some line that best fits the some data. To do so it tries to estimate at least 2 parameters that describe a line, the intercept and the slope. The equation of a line is y = Mx + c where c is the intercept and M is the slope, with y and x representing our explanatory (x) and response variables (y). If we know the intercept and slope we can estimate what y value we expect for any given x value.

We could manually try and fit our own lines to the data trying to get as close as we can. Lets try do it for three separate lines. We will show the full graph here so you can see the intercepts clearly and use the abline() function which plots lines according to abline(intercept, slope)

```
plot(pen_subset$flipper_length_mm ~ pen_subset$body_mass_g,
     pch = 16,
     xlab = "Body Mass (g)",
     ylab = "Flipper length (mm)",
     bty = "n",
     ylim = c(0, 230),
     xlim = c(0, 6500))

#line with intercept of 0 and a slope of 1
#abline()
abline (0,1,
        col = "coral")
#This line seems way too step so lets try a smaller slope and a higher intercept


#line with intercept of 100 and a slope of 10
abline (100,0.01,
        col = "blue4")
#The slope seems about right but we still need a higher intercept

#line with intercept of 100 and a slope of 10
abline (155,0.012,
        col = "green4")
```



```
#The slope seems about right
```

I could use the green line as my model to describe the data which could be written as y = 0.012x + 155. This means if I found a penguin that weighed 4000g my model would predict its flippers to be y = 0.012(4000) + 155 which is y = 203cm. This is called the expected value.

However, we can probably find a better line of fit and to do so we can use the idea of maximum likelihood. For linear model we simply pick the line which is closest to the data or more specifically reduces the distance to the residuals the most. This is called maximum likelihood were R will estimate the values of the intercept and slope that reduces the overall sum of the square of the residuals (i.e. the distance of the data points from the line).

We can do this using the lm() function which uses the Y ~ X notation

```
mod1 <- lm(pen_subset$flipper_length_mm ~ pen_subset$body_mass_g)

summary(mod1)
```

```
##
## Call:
## lm(formula = pen_subset$flipper_length_mm ~ pen_subset$body_mass_g)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7626  -4.9138   0.9891   5.1166  16.6392
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.367e+02  1.997e+00   68.47   <2e-16 ***
## pen_subset$body_mass_g 1.528e-02  4.668e-04   32.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

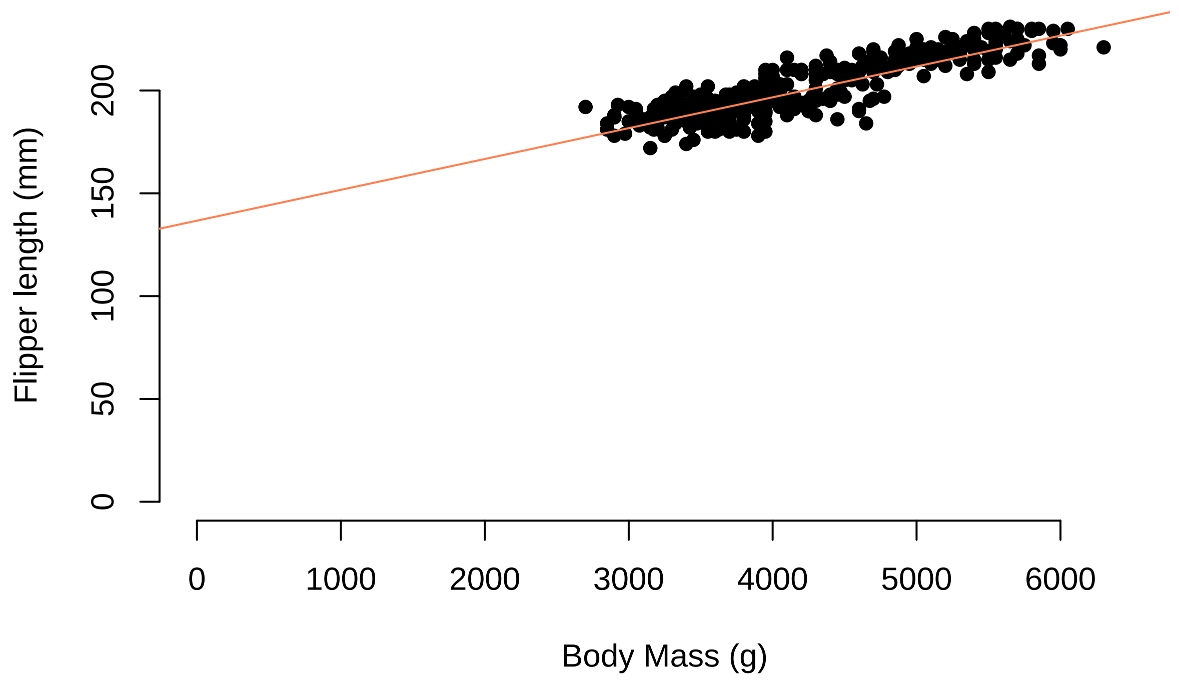
```
##
## Residual standard error: 6.913 on 340 degrees of freedom
## Multiple R-squared:  0.759,  Adjusted R-squared:  0.7583
## F-statistic:  1071 on 1 and 340 DF,  p-value: < 2.2e-16
```

The output gives a small summary of the residuals distribution (min, max, median , and the 1st and 3rd quarterly). Remember the residuals are just the distance of the data points from the fitted line. We can also see the r-squared value at the bottom of the summary as 0.759 which was what we estimated above. The adjusted R-squared is a version of this value taking into account how many parameters there are in the model. For ou purposes we will ignore the final line with the F-statistic and p-value for the entire model as we are interested in the intercept and slope of the model.

The Coefficients part of the output gives the estimate for both the intercept and the slope of our best fitted line. In this case the intercept is 136.7 (when you see the e+02 it means move the decimal point to the right 2 places). The slope of the line is given as 0.01528 (e-02 means move the decimal place to the left 2 places). This means the best fit line can be expressed as y = 0.015x + 136.7.

The Std. Error gives a measure of how sure the estimate for the each of the intercept and slope is. A high Std. Error indicates that there is a lot of uncertainty around the estimate value. Before looking at the t and p-values lets plot this line.

```
plot(pen_subset$flipper_length_mm ~ pen_subset$body_mass_g,
     pch = 16,
     xlab = "Body Mass (g)",
     ylab = "Flipper length (mm)",
     bty = "n",
     ylim = c(0, 230),
     xlim = c(0, 6500))

#line with intercept of 136.7 and a slope of 0.015
#abline()
abline (136.7, 0.015,
        col = "coral")
```



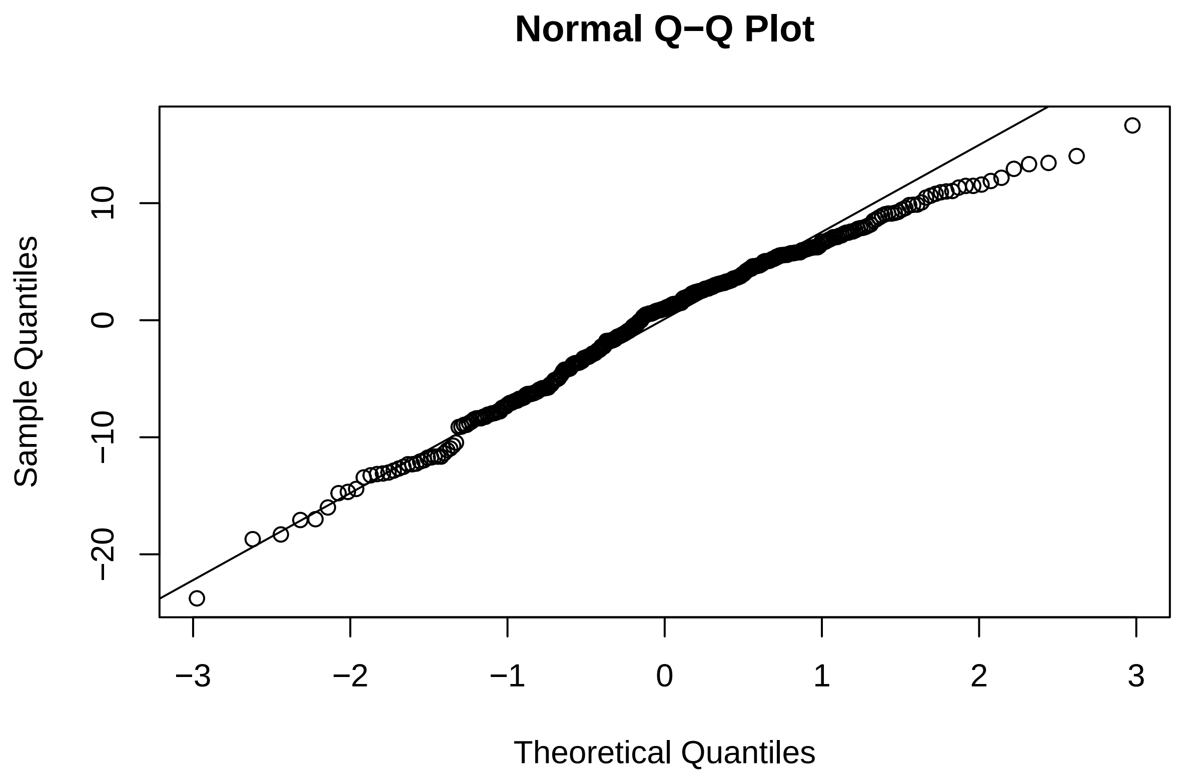
We can see from the plot that the line now fits the data very well. We can also see that the line has a value of 136.7 when x = 0 which is the intercept (BTW when plotting you do not need to show the intercept at zero I am only doing this to demonstrate the point). The slope of 0.015 also tells us something very important about the relationship between the body size and flipper length. It means for every unit increase in body size we change by 0.015 units on flipper length. That is for every gram increase we expect flipper length to increase by 0.015mm.

However, we also need to test if this relationship is just due to chance, that is we need to test if we can reject the null hypothesis that there is no relationship between body mass and flipper length. The linear model does this for every parameter so in this case there is a Null hypothesis tested for the intercept value and the slope. For the intercept the Null model is that the intercept is zero, for the slope the Null is that the slope is zero. In the summary the t-values are used to calculate a p-value for each of these estimates and test if the estimate values of these parameters is significantly different to zero. In this case the p-values for both intercept and slope are well below 0.05 so we can reject the null hypothesis for both these parameters.

Checking the model assumptions

The standard linear model assumes the residuals are normally distributed so we need to check this. We can check just like with the ANOVA using a qq-plot

```
qqnorm(mod1$residuals)
qqline(mod1$residuals)
```



This looks generally good as the points are generally on the line.

If you want you can further explore your model just by using plot(mod1) which will give you 4 new plots. The first plot (residuals vs. fitted values) is a simple scatterplot between residuals and predicted values. It should look more or less random. The second is a qq plot like above.

The third plot (Scale-Location), is very similar to the first, it should look random with no patterns.

The last plot (Cook's distance) tells us which points have the greatest influence on the regression (leverage points). Points with high leverage are outliers that if moved slightly could change the intercept and slope estimate the most. Its a good way of pinpoint which data might be having an oversize importance in your model.

However, what if your residuals are clearly not normal, then what? One thing is to transform your data.

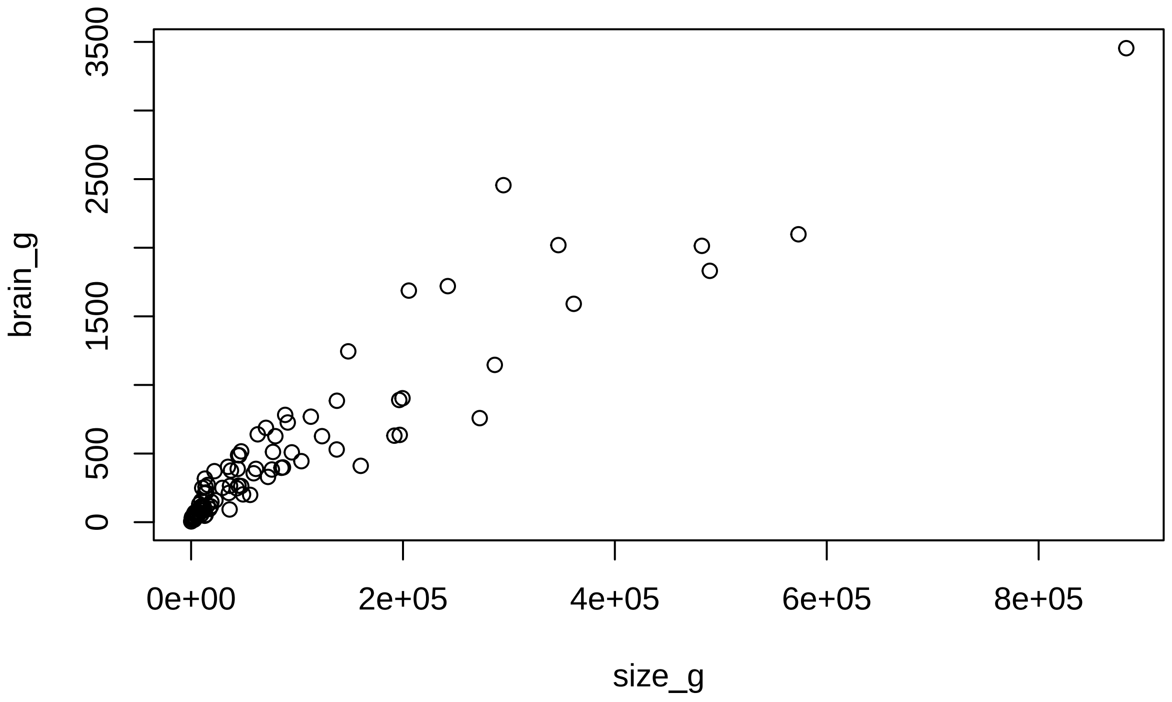
Transforming you data.

Lets create a scenario where the data is log-normal. To do so I will simulate some data on body size and brain size for 100 species. The body size data is randomly generate while the brain size data based on a model I made up, notice how brain size is calculated according to the power of size. This relationship will be reflected in our model.

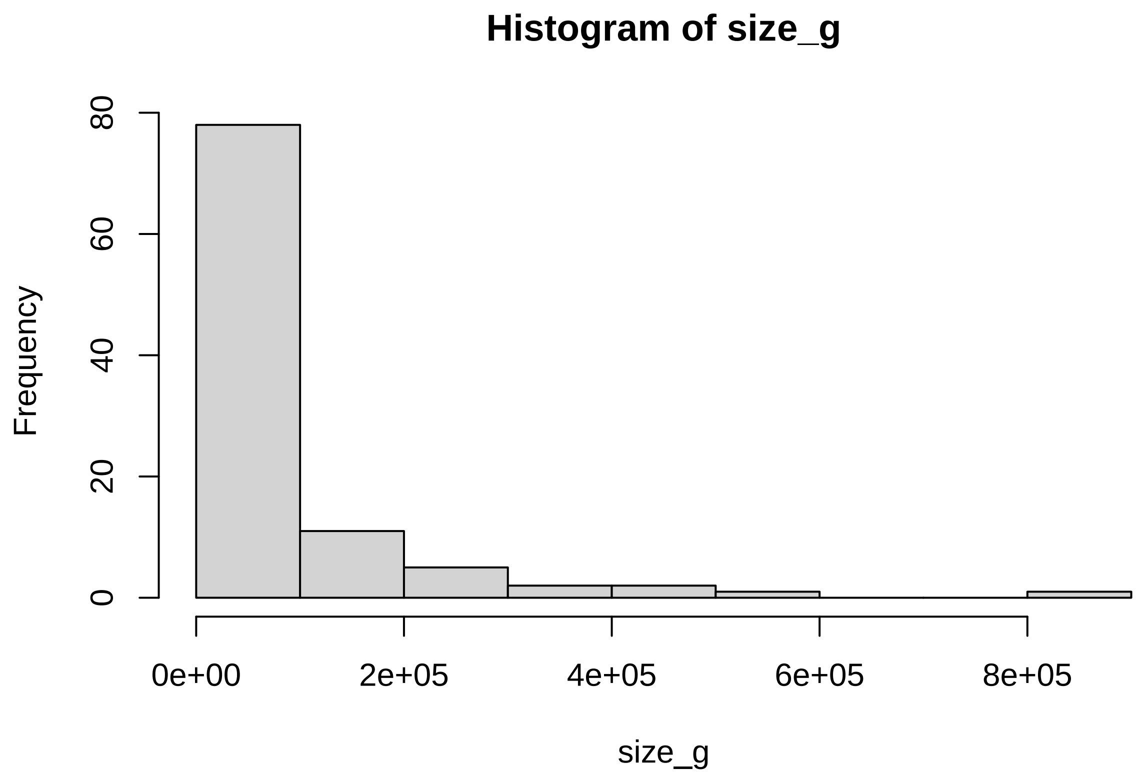
```r
#generate some random body sizes
size_g <- rlnorm(100, 10, 2)

#generate some brain sizes based on a simple scaling model
brain_g <- 0.1*rlnorm(100,0.01,0.4)*size_g^0.75 + rnorm(100,1,1)
```

Lets plot a scatted plot and the histogram of these.

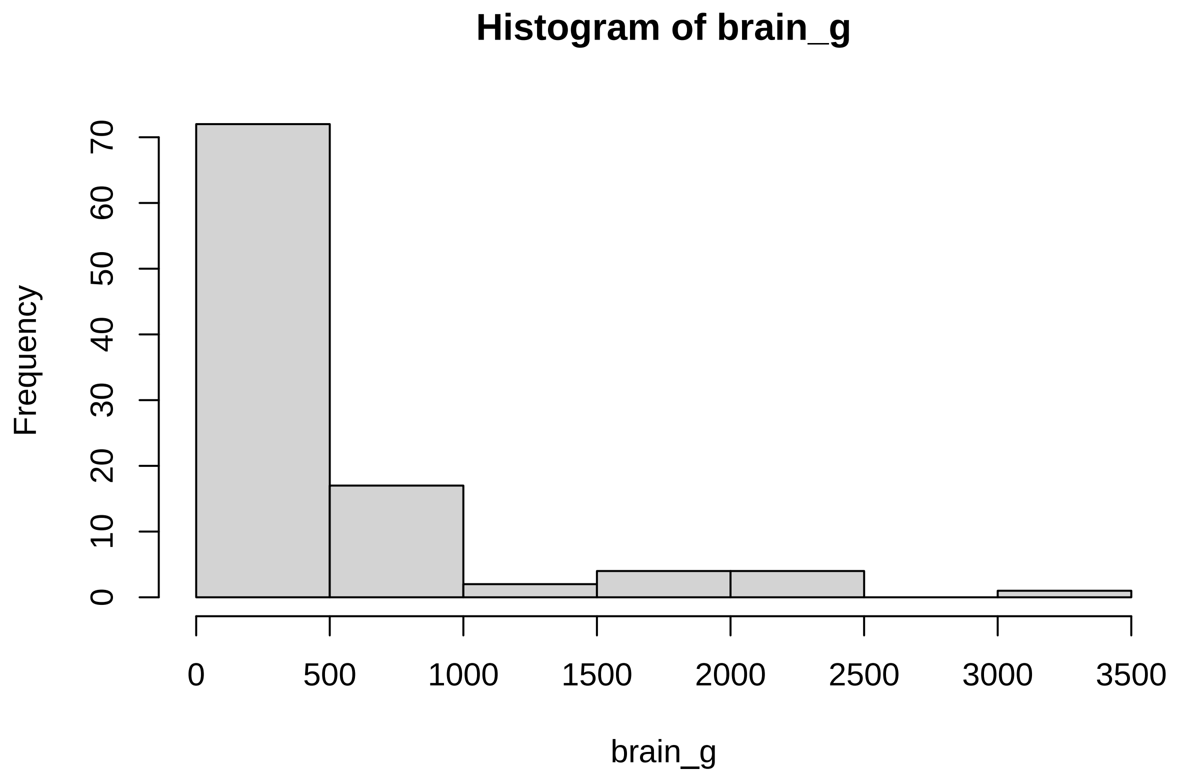
```r
plot(brain_g ~ size_g)
```



```r
hist(size_g)
```

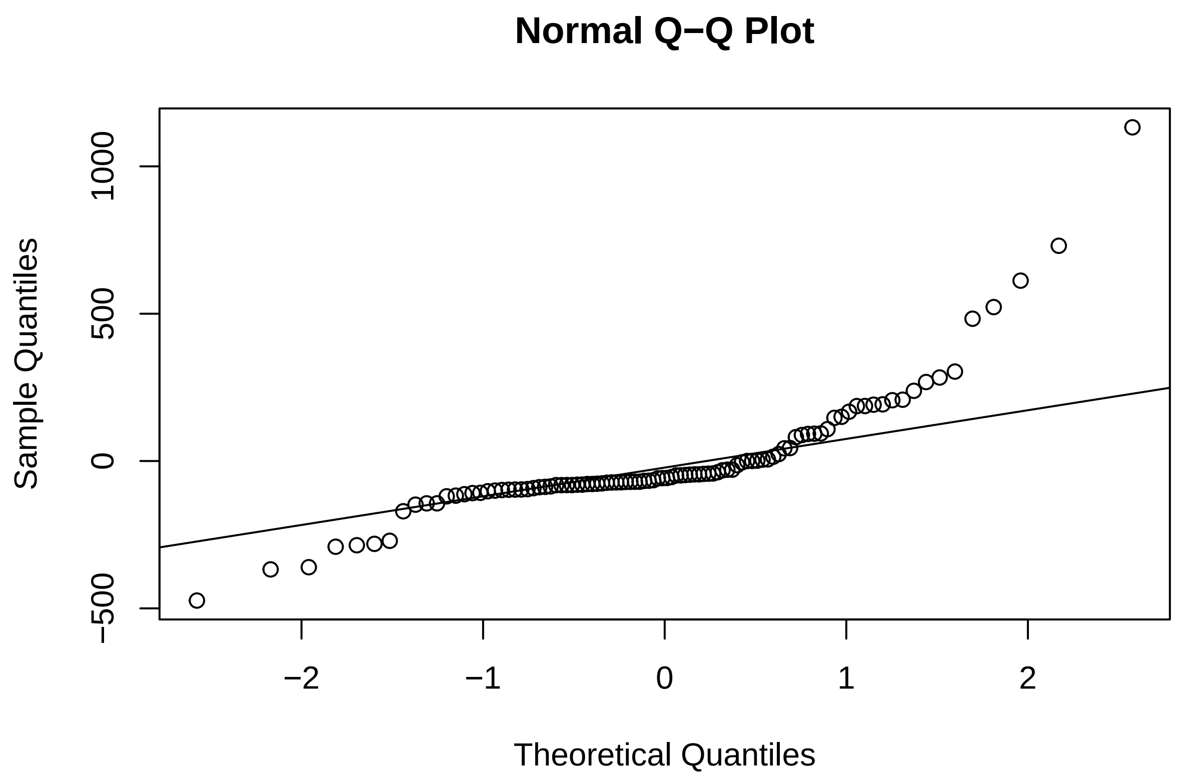


```
hist(brain_g)
```



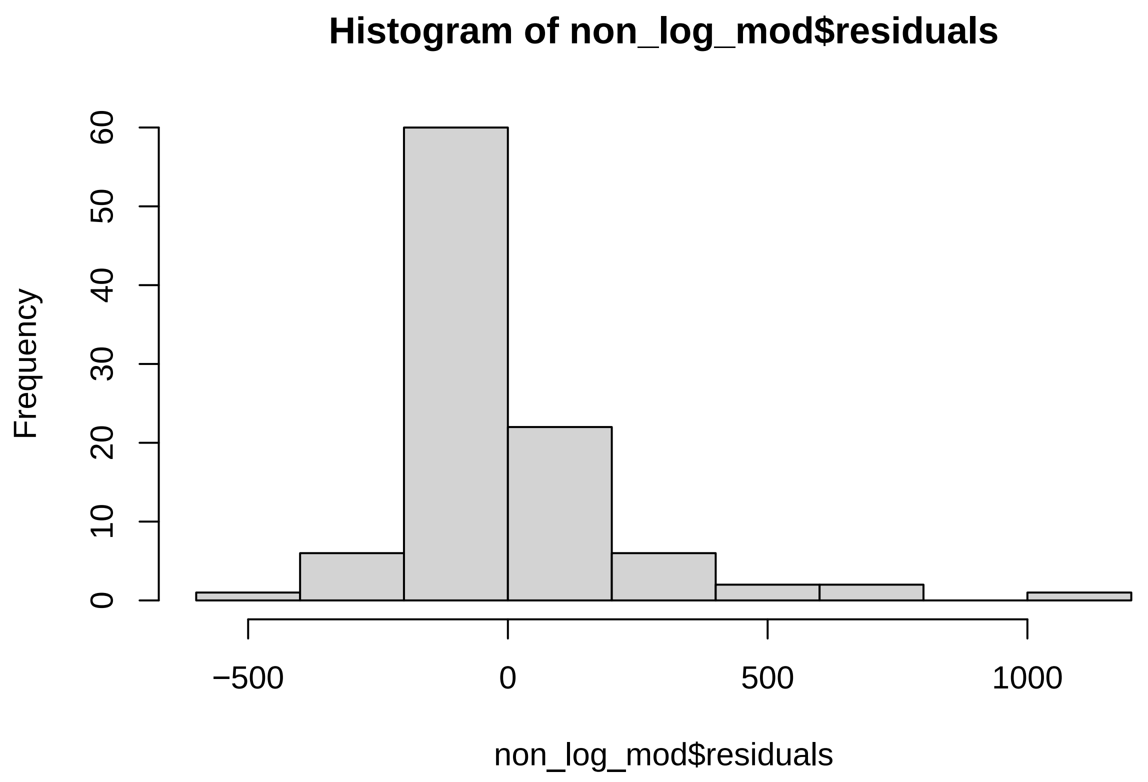
These are extremely log normal with most values being very small while a few are extremely large like a blue whale. Lets fit a linear model of this and check residuals.

```r
non_log_mod <- lm(brain_g ~ size_g)

qqnorm(non_log_mod$residuals)
qqline(non_log_mod$residuals)
```



We can see the lower and upper quantiles of the residual distribution does not match what we would expect form a normal distribution of residuals, this is even clearer if we plot a histogram of the residuals with most the values near the middle but a few values skewed left and right.

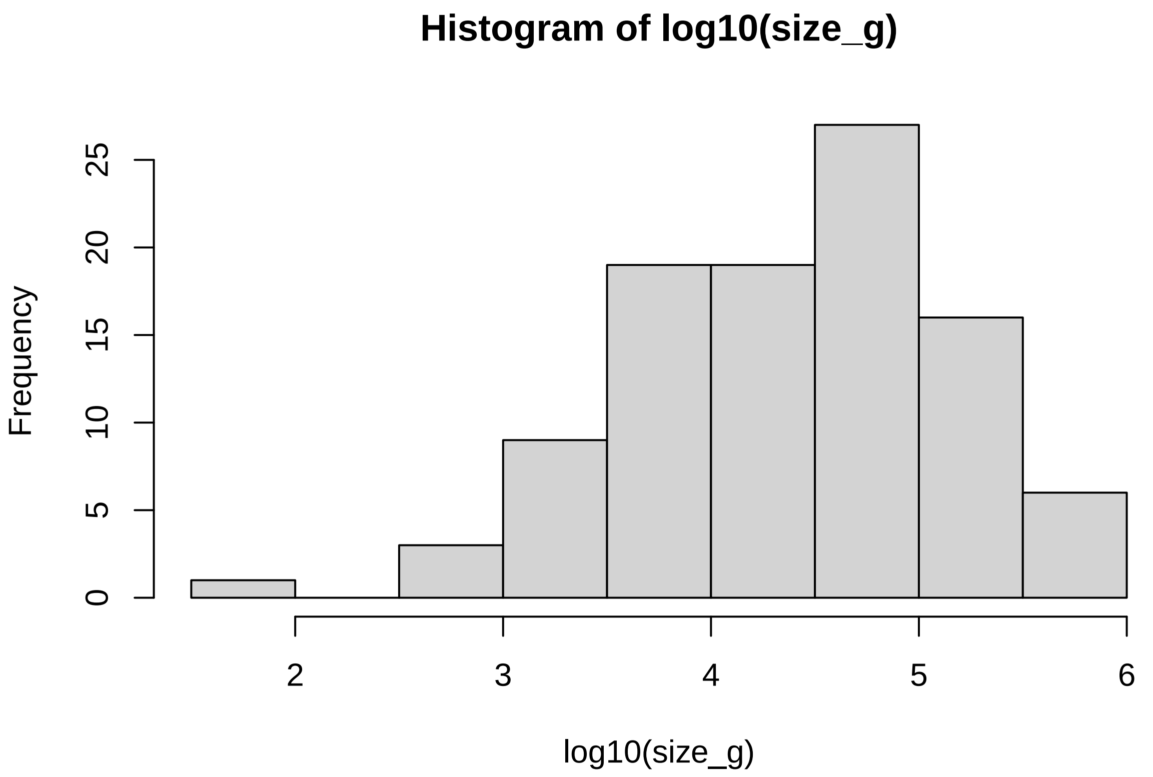
```r
hist(non_log_mod$residuals)
```



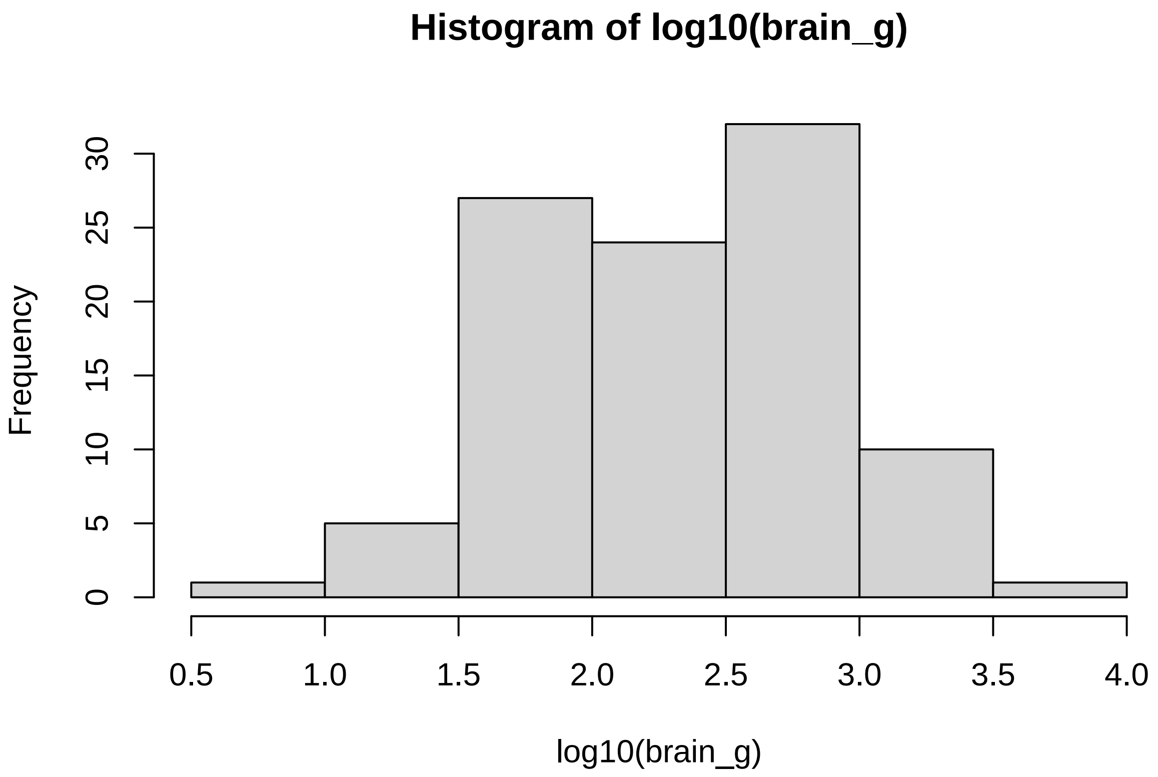
So what to do. One easy transformation is the log transformation. By getting the log of all the values we compress them together onto a scale where outliner are less of an issue.

From a maths point of view what we are doing is changing the equation of the line from y = mx +c to log(y) = mlog(x) + log(c). Lets try this.

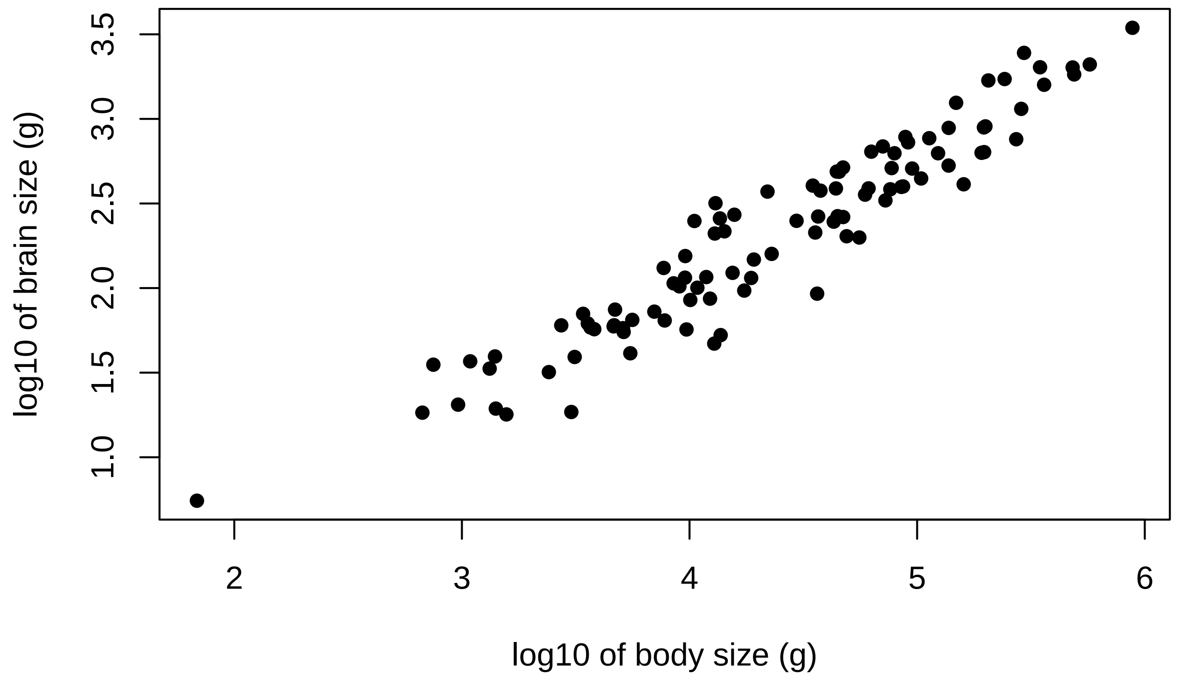
```
hist(log10(size_g))
```



```
hist(log10(brain_g))
```



```
plot(log10(brain_g) ~ log10(size_g),
     pch = 16,
     xlab = "log10 of body size (g)",
     ylab = "log10 of brain size (g)")
```



These look much better. The data no longer increases is units of grames but in orders of magnitube. Since we used base 10 with log10() this means 1 here is 10^1 = 10, 2 stands for 10^2 = 100, 3 stands for 10^3 = 1000 and so on. This is called a log scale.Notice also that the data is much more like a line. This happened as the relationship between brain size and body size is actually a power law of the form y = x^m + c (have a look at my simulation for brain size). When we logged this we get log(y) = log(x^m) + log(c) which can be expressed as log(y) = mlog(x) + log(c) which is the equation of the line. In effect we changes a curved line (the power law) into a straight line which makes doing statistics much easier. It also means that the slope in our logged model represents the power in the relationship between the X and Y axis.

Lets fit our model.

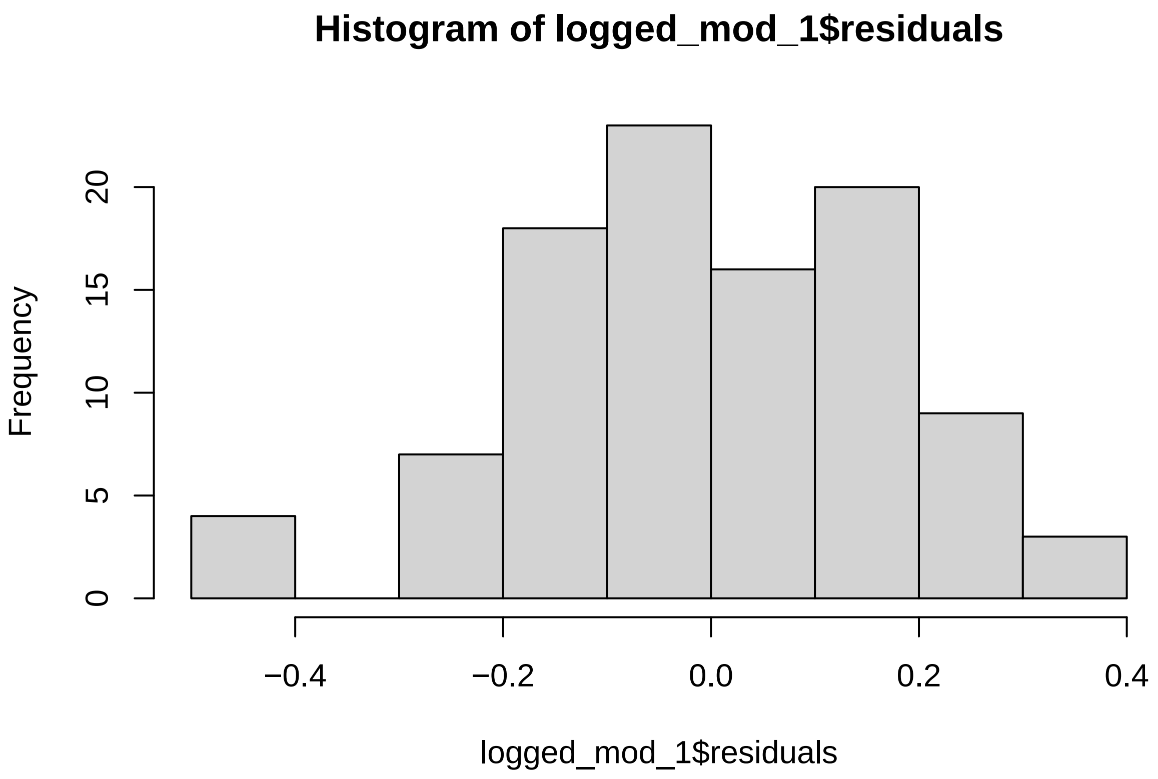
```
logged_mod_1 <- lm(log10(brain_g) ~ log10(size_g))
summary(logged_mod_1)
```

```
##
## Call:
## lm(formula = log10(brain_g) ~ log10(size_g))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47760 -0.10688 -0.01338  0.13905  0.37454
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.79712    0.10170  -7.838 5.61e-12 ***
## log10(size_g)  0.71080    0.02289  31.051  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1778 on 98 degrees of freedom
## Multiple R-squared:  0.9077, Adjusted R-squared:  0.9068
## F-statistic: 964.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

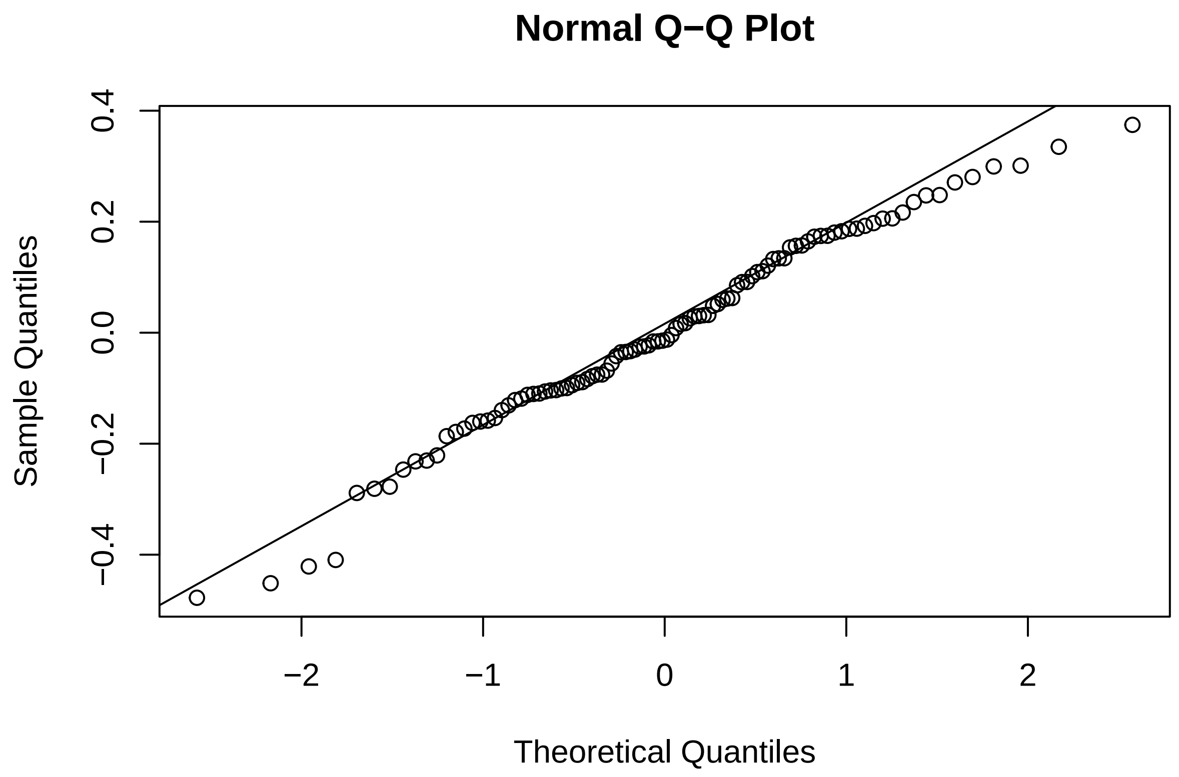
Notice the slope is close to 0.75, which is the power I used in my simulated data. The intercept is -1.14 (notice how I round up to 2 decimal places, this is good practice) and both the intercept and slope are signifincatly differnt to zero.

We can do all the other checks with the q plot here too.

```
hist(logged_mod_1$residuals)
```



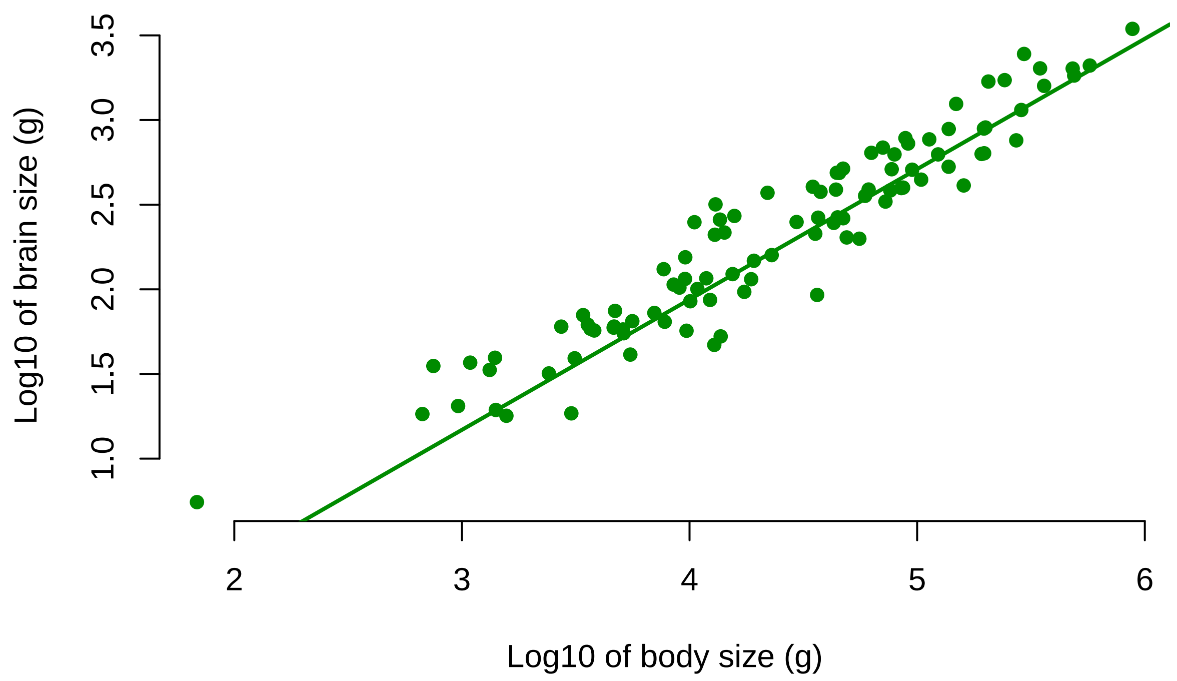
```
qqnorm(logged_mod_1$residuals)
qqline(logged_mod_1$residuals)
```



This is better, its not perfect but you will rarely get that. There are still some values that are lower than expected (Note this may be different when you run the code as the simulation has a large random element to it).

If we are happy with it we can plot our final model.

```
plot(log10(brain_g) ~ log10(size_g),
     pch = 16,
     xlab = "Log10 of body size (g)",
     ylab = "Log10 of brain size (g)",
     col ="green4",
     bty = "n")


abline(-1.14, 0.77,
       col ="green4",
       lwd = 2)
```


Log10 of brain size (g)
1.0
1.5
2.0
2.5
3.0
3.5
Log10 of body size (g)
2
3
4
5
6