General Linear Models in R

Andrew McAdam

2020-11-02

Contents

1 Overview

General linear models are a flexible statistical framework for analyzing a variety of types of data. They encompass many of the statistical approaches that you are probably already familiar with (e.g. ANOVA, regression) and also provide a foundation for dealing with more complex analyses including non-normal error distributions (generalized linear models) and the inclusion of random effects (mixed-effect models).

A general linear model (glm) is a statistical model that predicts some response variable, y, by linear combinations of a variety of predictor variables. For example,

$$y_i = \beta_0 \ + \beta_1 \ X_1 + \beta_2 \ X_2 + \beta_3 \ X_3 + \beta_4 \ X_4 + e_i$$

How is this different from a multiple regression? It isn't really. The only restriction is that in a multiple regression all of the X's are continuous variables. In an ANOVA all of the X's are categorical. In a general linear model we can use the same approach for all of these cases as well as those where we have a mix of categorical and continuous variables,

2 Continuous Predictor Variables

We will begin with a general linear model that includes only contonuous predictors (i.e. a regression). For this example, I will use a dataset on the timing of spring breeding by red squirrels. The Julian Date of the year on which they give birth is referred to as 'Julian' in the dataset and there are several other covariates also included in the data frame.

To begin with we will reload the data set for today and make the necessary changes to the variables. This code imports the data file called PartD2.csv and then converts some variables to factors while creating a few others.

```
Part.temp<-read.csv("data/PartD2.csv", header=T)
 Part.temp$BR<-factor(Part.temp$BR)
 Part.temp$LN<-factor(Part.temp$LN)
 Part.temp$Dam<-factor(Part.temp$Dam)
 Part.temp$FOOD<-factor(Part.temp$FOOD)
 Part.temp$BYF<-factor(Part.temp$BY)
 Part.temp$AGEF<-factor(Part.temp$AGE)
 Part.temp$YRF<-factor(Part.temp$YR)
```

We can retrieve the number of observations using…

```
length(Part.temp$GRID)
```

```
## [1] 1422
```

And the number of columns (or variables) as

```
length(Part.temp)
```

```
## [1] 14
```

It is always good to summarize the data you have imported just to make sure everything looks the way it should.

```
summary(Part.temp)
```

```
##      GRID                 BR             YR            AGE
##   Length:1422        1      :1211   Min.   :1989   Min.   :1.000
##   Class :character   2      :  85   1st Qu.:1994   1st Qu.:2.000
##   Mode  :character   4      :  75   Median :1997   Median :3.000
##                      3      :  27   Mean   :1997   Mean   :2.943
##                      0      :  12   3rd Qu.:2000   3rd Qu.:4.000
##                      6      :   6   Max.   :2004   Max.   :8.000
##                      (Other):   6                  NA's   :42
##        BY       LN            Dam            FOOD          LSIZE
##   Min.   :1983   0:  13   166    :   8   0      :1080   Min.   :0.000
##   1st Qu.:1991   1:1408   167    :   7   2      : 102   1st Qu.:2.000
##   Median :1994   5:   1   485    :   7   3      :  19   Median :3.000
##   Mean   :1994            9      :   6   4      :  19   Mean   :2.734
##   3rd Qu.:1997            201    :   6   10     :  16   3rd Qu.:3.000
##   Max.   :2003            218    :   6   (Other):  10   Max.   :7.000
##                           (Other):1382   NA's   : 176   NA's   :11
##      Julian          conestm1          BYF           AGEF          YRF
##  Min.   : 57.00   Min.   :0.000   1993   :203   2      :371   1999   :181
##  1st Qu.: 99.25   1st Qu.:1.402   1998   :173   3      :337   1996   :129
##  Median :116.00   Median :2.759   1995   :159   1      :233   1994   :115
##  Mean   :117.96   Mean   :2.677   1997   :159   4      :231   1995   :111
##  3rd Qu.:134.00   3rd Qu.:3.748   1991   :119   5      :121   1998   : 98
##  Max.   :203.00   Max.   :5.327   1992   : 84   (Other): 87   1992   : 97
##                                   (Other):525   NA's   : 42   (Other):691
```

So everything looks pretty good. We have both continuous and categorical predictors to work with.

2.1 Specifying a Linear Model

The first thing we are going to do is to relate food abundance to the timing of breeding by female red squirrels in the spring.

There is one continuous predictor and one response variable so we are dealing with a simple regression. This part should conceptually be review for you, although the inputs and outputs from R might be new.

We specify all general linear models in R using the lm command.

```
julian.lm<-lm(Julian~conestm1, data=Part.temp, na.action=na.omit)
```

Note that there was no output from this command. We simply stored the statistical model into the object that we called *julian.lm*. We could have called this *apple* or *bobo* but *julian.lm* makes a little more sense to me.

Here we have specified the response variable *Julian*, which is the date on which a female bred, as well as one predictor variable, *conestm1*. This is a continuous variable that simply measures the amount of cones produced by spruce trees in the previous fall. Note that we specify the model as a linear model where *Julian* is modelled by *conestm1*. We represent "is modelled by" in R syntax using the ~ (tilde) symbol. We have then indicated where R can find these variables using *data=Part.temp*. Note that we could have instead used the following:

```
julian.lm<-lm(Part.temp$Julian~Part.temp$conestm1, na.action=na.omit)
```

Here instead of indicating the two variables within the specified data frame we have indicated two vectors separately.

The last statement in the original code above simply tells R what we should do with missing data. This is the *na.action* command. There are a few options here. For now we will simply tell R to omit observations for which either of these variables is missing. The code for this is *na.action=na.omit*.

Bonus Recall Question: How would you plot this relationship, and the regression line from this julian.lm model?

We can recall the model like any other object in our workspace using

```
julian.lm
```

```
##
## Call:
## lm(formula = Part.temp$Julian ~ Part.temp$conestm1, na.action = na.omit)
##
## Coefficients:
##        (Intercept)  Part.temp$conestm1
##             146.41              -10.63
```

This provides the model formula, the slope and the intercept. But this isn't what we are used to seeing. We still don't have any stats!

```
summary (julian.lm)
```

```
##
## Call:
## lm(formula = Part.temp$Julian ~ Part.temp$conestm1, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.585 -11.987  -1.820   9.206 108.199
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        146.4099     0.9409  155.61   <2e-16 ***
## Part.temp$conestm1 -10.6268     0.3041  -34.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.79 on 1420 degrees of freedom
## Multiple R-squared:  0.4624, Adjusted R-squared:  0.462
## F-statistic:  1221 on 1 and 1420 DF,  p-value: < 2.2e-16
```

Ahhh. Now this looks a little bit more familiar!

2.2 Interpretting the Output of a Linear Model

So let's walk through the output from R to make sure that we are all on the same page. At the start of the output is simply a restatement of the model as we entered it. The next part of the output proivides some desccriptive statistics on the residuals (see below for more information on residuals). Next is the actual 'meat' of the analysis. Recall that statistics is about BOTH estimation and inference. Here we are interested in estimating parameters (what is the slope of this relationship?), as well as making inferences about whether these parameters are different from zero. Since we are dealing with a regression we are estimating a slope and an intercept. These are the coefficients in the model (also called parameters). From the output you can see that the slope is -10.6 and the intercept is 146.4. Each of these is estimated with some uncertainty. In this output this is represented by the standard error (Std. Error) for these parameters.

Next we have the inference part of the model. Are the slope and intercept different from zero? In a regression we test these hypotheses using a t-distribution with some degrees of freedom. We calculate the t test-statistic as the ratio of the parameter to its standard error (you might not have known this previously). So note that:

```
146.4099/0.94099
```

```
## [1] 155.5913
```

```
#and
-10.56268/0.3041
```

```
## [1] -34.73423
```

These test statistics are then tested against the known t distribution to come up with a p-value for each. Since P is less than 0.05 in both cases we reject our null hypothesis of no difference between our paramater and zero and instead accept the hypothesis that the two paramaters are in fact different from zero. Note that the scientific notation *<2e-16* simply indicates that P is less than 2 * 10^{-16} (a very small number). As a quick way to visualize the results R also throws a bunch of asterisks after significant P-values.

Note that we can check the summary command's math using the pt command in R. This returns a p-value for a given t-statistics and known df. We can ask R to give us the p-value from a t distribution with 1420 df and a t test statistic of 155.61 using:

```
2*pt(155.61, df=1420, lower.tail=F)
```

```
## [1] 0
```

This is a handy command for checking other people's p-values! Note that we needed first specify that we are talking about the upper tail of the distribution (lower.tail=F) because the t statistic is positive. Second note that the df are the error df from the entire model (see the model summary). The significance of parameters in linear models are traditionally tested using two-tailed tests. This is why I mutliplied the probability by two (not that it mattered in this case).

As an aside, have you ever wondered why people sometimes use the Rule of Thumb that if the parameter is more than twice as big as its standard error then it is significantly different from zero? Well take a look at this:

```
2*pt(2, df=1000, lower.tail=F)
```

```
## [1] 0.04577035
```

```
2*pt(2, df=100, lower.tail=F)
```

```
## [1] 0.04821218
```

```
2*pt(2, df=80, lower.tail=F)
```

```
## [1] 0.04889385
```

```
2*pt(2, df=70, lower.tail=F)
```

```
## [1] 0.04938161
```

```
2*pt(2, df=60, lower.tail=F)
```

```
## [1] 0.05003304
```

```
2*pt(2, df=10, lower.tail=F)
```

```
## [1] 0.07338803
```

So while it depends (a bit) on the sample size, in most cases this Rule of Thumb works well. This Rule of Thumb is then why it is common to show SEs on figures and not some other measure of uncertainty (e.g. sd or variance).

Back to our summary. The last part of summary output contains the information on the significance of the OVERALL MODEL and not the INDIVIDUAL PARAMETERS. Here we are no longer testing using a t-statistic, but now we are using an F-statistic. So when we report the significance of the slope we use t_{1420} *= -34.95, P < 0.001* and when we report on the significance of the overall model we use *F~1, 1420~=1221, P < 0.001*. Note that the numerator and denominator df are listed in the summary output. The numerator df is simply based on the number of parameters (other than the intercept) that are being estimated. The denominator df is based on the overall sample size and the number if parameters being estimated.

$df_{denominator} = n - 1 - df_{numerator}$

Note that when we tested the signficiance of the slope parameter we tested it using the t-staristic and the error (donominator) degrees of freedom. Finally, the summary output also indicates the R^2, which represents the proportion of variation in your response variable that is explained by your predictor variable. The summary also reports an adjusted R^2 value, which is designed to try and penalize the R^2 for the complexity of the model. Note that in this case the two are very similar because the model is very simple. We will talk later (see Model Selection notes) about better ways to assess the fit of models to data while accounting for the complexity of the model to try and assess "what is a good/better model?".

2.3 What is a Linear Model Anyway?

Remember that our statistical model

```
julian.lm
```

```
##
## Call:
## lm(formula = Part.temp$Julian ~ Part.temp$conestm1, na.action = na.omit)
##
## Coefficients:
##        (Intercept)  Part.temp$conestm1
##             146.41              -10.63
```

represents a linear model that has some equation:

$ y=b_{0} +b_{1}X_{1} + e $

In this case: $ParturitionDate_i = 146.4 - 10.6 * conestm1 + e_i$

So for female i, experiencing food abundance X_i we can predict what her parturition date should be:

$$ParturitionDate_i = 146.4 - 10.6 * conestm1_i + e_i$$

So let's take an example in R. How about female 300? Recall that we can index the datafile *Part.temp* to only consider row 300 using…

```
Part.temp[300, ]
```

```
##     GRID BR   YR AGE   BY LN Dam FOOD LSIZE Julian conestm1  BYF AGEF  YRF
## 300   KL  3 1993   1 1992  1 213    0     2    142 3.454216 1992    1 1993
```

We would predict that her parturition date would be: $146.4 - 10.6 * conestm1$, or

```
146.4-10.6*3.45216
```

```
## [1] 109.8071
```

because she experienced a cone abundance of 3.45216.

Note that we can get R to do some work for us:

```
coef(julian.lm)
```

```
##        (Intercept) Part.temp$conestm1
##          146.40987          -10.62678
```

This returns all of the coefficients in a model. We can instead ask R for only the first coefficient in the model using

```
coef(julian.lm)[1]
```

```
## (Intercept)
##    146.4099
```

Or the second coefficient using…

```
coef(julian.lm)[2]
```

```
## Part.temp$conestm1
##          -10.62678
```

We can use these in an equation too, such as…

```
coef(julian.lm)[1]+(coef(julian.lm)[2]*Part.temp$conestm1[300])
```

```
## (Intercept)
##    109.7027
```

We can get rid of that remaining text associated with the coefficients by using the as.numeric command.

```
as.numeric(coef(julian.lm)[1])+(as.numeric(coef(julian.lm)[2])*Part.temp$conestm1[300])
```

```
## [1] 109.7027
```

Or we can get this in an even more straight forward way

```
predict(julian.lm)[300]
```

```
##      300
## 109.7027
```

The difference between our calculation and R's is due to rounding. The point of this was not to reinvent the wheel, but to remind you that a linear model is simply a best prediction of what the response variable will be across all observations and that all observations have both an observed value and a predicted value.

Remember also that the difference between the predicted value and the observed value is our residual for that female. That is, this is the value of e for female i (i.e., e_i). haha, that is stats homour!

```
Part.temp$Julian[300]-predict(julian.lm)[300]
```

```
##      300
## 32.29733
```

Or simply

```
resid(julian.lm)[300]
```

```
##      300
## 32.29733
```

So the residual for female 300 is 32.29. That is she bred 32 days after we would have predicted based on food abundance alone.

Her parturition date (142) is equal to $142 = 146.4 - 10.6 * 3.45 + 32.3$

3 General Linear Models with Categorical Predictors

Most people find regressions to be a bit more intuitive than a lm with categorical predictors but the same basic idea applies. The response variable is predicted by one or more categorical predictors. When the categorical predictor has more than one level R needs to recode this variable containing n levels into n-1 coding variables that help to explain all levels based on a series of Yes/No questions (coded as 1 or 0).

We will use an example to look at the effects of Age on the timing of breeding.

```
age.model<-lm(Julian~AGEF, data=Part.temp, na.action=na.omit)
```

Note that we used *AGEF* because this variable is a factor. The variable *AGE* is not a factor and if we included that in our model we would have been specifying a regression of parturition date on age. We could have also specified age as a factor using:

```
age.model<-lm(Julian~factor(AGE), data=Part.temp, na.action=na.omit)
```

In both cases we specified the model just like we did for a regression. R doesn't care whether you are using continuous predictors (regression) or categorical predictors (ANOVA) or some combination of the two (ANCOVA or something more complicated). THESE ARE ALL LINEAR MODELS (!!!) and are specified in R using the 'lm' command.

We can also recall and summarize the results of our model using the same commands as above.

```
age.model
```

```
##
## Call:
## lm(formula = Julian ~ factor(AGE), data = Part.temp, na.action = na.omit)
##
## Coefficients:
##  (Intercept)  factor(AGE)2  factor(AGE)3  factor(AGE)4  factor(AGE)5
##      124.318        -4.671        -8.237        -9.067        -4.086
## factor(AGE)6  factor(AGE)7  factor(AGE)8
##      -11.302        -6.791        -1.318
```

Or we can use the more informative summary command.

```
summary(age.model)
```

```
##
## Call:
## lm(formula = Julian ~ factor(AGE), data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   124.318      1.576  78.887  < 2e-16 ***
## factor(AGE)2   -4.671      2.011  -2.323 0.020332 *
## factor(AGE)3   -8.237      2.050  -4.019 6.16e-05 ***
## factor(AGE)4   -9.067      2.233  -4.059 5.20e-05 ***
## factor(AGE)5   -4.086      2.695  -1.516 0.129765
## factor(AGE)6  -11.302      3.374  -3.350 0.000831 ***
## factor(AGE)7   -6.791      5.739  -1.183 0.236889
## factor(AGE)8   -1.318     13.977  -0.094 0.924911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

We can also summarize this model in the more familiar ANOVA output using the anova command

```
anova (age.model)
```

```
## Analysis of Variance Table
##
## Response: Julian
##               Df Sum Sq Mean Sq F value   Pr(>F)
## factor(AGE)    7  15179 2168.48  3.7475 0.000497 ***
## Residuals   1372 793899  578.64
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So there are significant age effects but this doesn't tell us anything about the effects of any one age category. More specifically, it doesn't provide us with any of the parameters are that we were interested in. So lets go back to the summary output.

```
summary (age.model)
```

```
##
## Call:
## lm(formula = Julian ~ factor(AGE), data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   124.318      1.576  78.887  < 2e-16 ***
## factor(AGE)2   -4.671      2.011  -2.323 0.020332 *
## factor(AGE)3   -8.237      2.050  -4.019 6.16e-05 ***
## factor(AGE)4   -9.067      2.233  -4.059 5.20e-05 ***
## factor(AGE)5   -4.086      2.695  -1.516 0.129765
## factor(AGE)6  -11.302      3.374  -3.350 0.000831 ***
## factor(AGE)7   -6.791      5.739  -1.183 0.236889
## factor(AGE)8   -1.318     13.977  -0.094 0.924911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

In order to interpret these parameters we need to know that the default contrasts for R are *treatment contrasts* in which the first level of the factor is put in the intercept and the parameters for each other level represents the difference between the mean for that category and the intercept category. For more information on Contrasts please see the General Linear Models Primer.

We can change this default using:

```
options(contrasts=c("contr.helmert", "contr.poly"))
age.model<-lm(Julian~AGEF, data=Part.temp, na.action=na.omit)
summary (age.model)
```

```
##
## Call:
## lm(formula = Julian ~ AGEF, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 118.63360    1.95770  60.598  < 2e-16 ***
## AGEF1        -2.33535    1.00538  -2.323 0.020332 *
## AGEF2        -1.96738    0.55054  -3.574 0.000364 ***
## AGEF3        -1.19095    0.44335  -2.686 0.007313 **
## AGEF4         0.28150    0.46038   0.611 0.541008
## AGEF5        -1.01501    0.51164  -1.984 0.047476 *
## AGEF6        -0.08059    0.79622  -0.101 0.919399
## AGEF7         0.62377    1.74083   0.358 0.720160
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

We will go back to the treatment contrasts in a minute since they are easier to interpret, but first note a couple of things: * Model Significance is Unchanged. Review the statistics for the overall models and see that the choice of contrasts does not affect the overall significance of the model.

- Parameters certainly do change! Note that the parameters (including the intercept) are very different in the two model formulations. This means that it is essential that we know what the contrasts are if we are going to make sense of the parameters in our linear model.

Returning to our original (default) contrasts

```r
options(contrasts=c("contr.treatment", "contr.poly"))
age.model<-lm(Julian~AGEF, data=Part.temp, na.action=na.omit)
summary (age.model)
```

```
##
## Call:
## lm(formula = Julian ~ AGEF, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  124.318      1.576  78.887  < 2e-16 ***
## AGEF2         -4.671      2.011  -2.323 0.020332 *
## AGEF3         -8.237      2.050  -4.019 6.16e-05 ***
## AGEF4         -9.067      2.233  -4.059 5.20e-05 ***
## AGEF5         -4.086      2.695  -1.516 0.129765
## AGEF6        -11.302      3.374  -3.350 0.000831 ***
## AGEF7         -6.791      5.739  -1.183 0.236889
## AGEF8         -1.318     13.977  -0.094 0.924911
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

Since we know that these are treatment contrasts then we know that the equation can be written out as:

$Parturitiondate = 124.3 - 4.7 * (2yrold?) - 8.2 * (3yrold?) - 9.1 * (4yrold?) - 4.1(5yrold?) - 11.3(6yrold?) - 6.8(7yrold? - 1.3(8yrold?)$

This is a long equation for just one predictor variable, but we need all of these parameters in the model to account for the 8 different levels of age. Remember that 8 levels of AGE results in 7 Yes/No questions and uses 7 df to estimate these 7 parameters. Note that I could tell which parameter goes with which yes/no question because the summary output lists not just the predictor variable ($AGEF$) but it also lists each specific level after the variable names (e.g. $AGEF2$, $AGEF7$). To come up with a predicted value you simply need to substitute a 1 when the answer to the question is Yes and a 0 when the answer is No. Note also that the coding variables should have only one Yes answer!

So what if we wanted to predict that parturition date for female 300 again but this time using our age model.

```
Part.temp[300, ]
```

```
##     GRID BR   YR AGE   BY LN Dam FOOD LSIZE Julian conestm1  BYF AGEF  YRF
## 300   KL  3 1993   1 1992  1 213    0     2    142 3.454216 1992    1 1993
```

She is a yearling so the equation is simple

Parturition date =

```
124.3-4.7*(0)-8.2*(0)-9.1*(0)-4.1*(0)-11.3*(0)-6.8*(0)-1.3*(0)
```

```
## [1] 124.3
```

In this case the residual is:

```
resid(age.model)[300]
```

```
##     300
## 17.6824
```

Note that this residual value for female 300 is less than the residual value for our previous regression model…

```
resid(julian.lm)[300]
```

```
##      300
## 32.29733
```

So the age model does a better job at predicting the parturition date of female 300. However, in order to determine whether it does a better job for all female we will need to compare the overall fit of the two models. We will leave that for another day.

We can calculate the predicted value for a more complicated female:

```
Part.temp[100, ]
```

```
##     GRID BR   YR AGE   BY LN Dam FOOD LSIZE Julian conestm1  BYF AGEF  YRF
## 100   SU  1 1990   3 1987  1 128    0     3    144        0 1987    3 1990
```

This female is a 3 year-old.

Parturition date =

```
124.3-4.7*(0)-8.2*(1)-9.1*(0)-4.1*(0)-11.3*(0)-6.8*(0)-1.3*(0)
```

```
## [1] 116.1
```

or

```
predict(age.model)[100]
```

```
##      100
## 116.0801
```

Remember that with the treatment contrasts we are looking at the parameters that compare the mean of each level to some reference level that has been “put” in the intercept. In this case our decision of ages was arbitrary, but we could have specified a specific order to the levels of the factor.

The levels command returns the levels of a specified variable

```
levels(Part.temp$AGEF)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8"
```

We can specify any order for these levels if we want to define a new reference category.

```
Part.temp$AGE.reverse<-factor(Part.temp$AGE, levels=8:1)
# or
Part.temp$AGE.arbitrary<-factor(Part.temp$AGE, levels=c(3,4,1,2,5,6,7,8))
levels(Part.temp$AGE.reverse)
```

```
## [1] "8" "7" "6" "5" "4" "3" "2" "1"
```

```
levels(Part.temp$AGE.arbitrary)
```

```
## [1] "3" "4" "1" "2" "5" "6" "7" "8"
```

```
summary(lm(Julian~AGE.reverse, data=Part.temp, na.action=na.omit))
```

```
##
## Call:
## lm(formula = Julian ~ AGE.reverse, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   123.000     13.888   8.856   <2e-16 ***
## AGE.reverse7   -5.474     14.944  -0.366    0.714
## AGE.reverse6   -9.985     14.205  -0.703    0.482
## AGE.reverse5   -2.769     14.059  -0.197    0.844
## AGE.reverse4   -7.749     13.978  -0.554    0.579
## AGE.reverse3   -6.920     13.950  -0.496    0.620
## AGE.reverse2   -3.353     13.944  -0.240    0.810
## AGE.reverse1    1.318     13.977   0.094    0.925
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

```
summary(lm(Julian~AGE.arbitrary, data=Part.temp, na.action=na.omit))
```

```
##
## Call:
## lm(formula = Julian ~ AGE.arbitrary, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     116.080      1.310  88.586  < 2e-16 ***
## AGE.arbitrary4   -0.829      2.055  -0.403    0.687
## AGE.arbitrary1    8.238      2.050   4.019 6.16e-05 ***
## AGE.arbitrary2    3.567      1.810   1.970    0.049 *
## AGE.arbitrary5    4.151      2.549   1.628    0.104
## AGE.arbitrary6   -3.065      3.259  -0.940    0.347
## AGE.arbitrary7    1.446      5.672   0.255    0.799
## AGE.arbitrary8    6.920     13.950   0.496    0.620
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

Note that in each case the overall significance of the model doesn't change but the parameters certainly do!! It also doesn't matter to the overall significance which contrasts you use, but again it does matter to the parameters. So it is IMPORTANT THAT YOU THINK ABOUT WHAT YOU ARE MODELING.

If you don't want to change all of the levels of the factor you can assign one level to be the new reference level using…

```r
Part.temp$AGE.arbitrary<-relevel(Part.temp$AGE.arbitrary, ref="6")
summary(lm(Julian~AGE.arbitrary, data=Part.temp, na.action=na.omit))
```

```
##
## Call:
## lm(formula = Julian ~ AGE.arbitrary, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     113.015      2.984  37.878  < 2e-16 ***
## AGE.arbitrary3    3.065      3.259   0.940 0.347141
## AGE.arbitrary4    2.236      3.377   0.662 0.508115
## AGE.arbitrary1   11.302      3.374   3.350 0.000831 ***
## AGE.arbitrary2    6.632      3.234   2.050 0.040529 *
## AGE.arbitrary5    7.216      3.699   1.951 0.051299 .
## AGE.arbitrary7    4.511      6.274   0.719 0.472237
## AGE.arbitrary8    9.985     14.205   0.703 0.482242
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.01876,    Adjusted R-squared:  0.01375
## F-statistic: 3.748 on 7 and 1372 DF,  p-value: 0.000497
```

```
levels (Part.temp$AGE.arbitrary)
```

```
## [1] "6" "3" "4" "1" "2" "5" "7" "8"
```

Before we leave this model. I want to show you the cell mean contrasts. We can specify this as a model that does not have an intercept. In a linear model the intercept is denoted by a one. If we instead fit a model of $y = x - 1$ then we are indicating that we want to predict y by x but we want to exclude the intercept from the model.

We can do this for our model of parturition date also.

```
summary(lm(Julian~AGEF-1, data=Part.temp, na.action=na.omit))
```

```
##
## Call:
## lm(formula = Julian ~ AGEF - 1, data = Part.temp, na.action = na.omit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.318 -18.318  -1.449  16.985  83.353
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## AGEF1  124.318      1.576  78.887   <2e-16 ***
## AGEF2  119.647      1.249  95.804   <2e-16 ***
## AGEF3  116.080      1.310  88.586   <2e-16 ***
## AGEF4  115.251      1.583  72.819   <2e-16 ***
## AGEF5  120.231      2.187  54.980   <2e-16 ***
## AGEF6  113.015      2.984  37.878   <2e-16 ***
## AGEF7  117.526      5.519  21.296   <2e-16 ***
## AGEF8  123.000     13.888   8.856   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.06 on 1372 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.9607, Adjusted R-squared:  0.9605
## F-statistic:  4193 on 8 and 1372 DF,  p-value: < 2.2e-16
```

Note here that we are no longer testing the statsitical null hypothesis of no DIFFERENCE between AGEs. Instead we are testing the null hypothesis that the timing of breeding is each year is equal to zero. This isn't very meaningful, because in this case zero represents December 31 and we know that squirrels don't breed in December!

3.1 Post hoc Tests

Some of you are probably thinking about the parameters from the treatment contrasts like some sort of post hoc test. In some ways this is correct and in some ways it is not. It is a pairwise test, but it does not test all pairwise relationships and these p values are not adjusted like they would be in a true *post hoc* test.

We can do pairwise t-tests between all levels of our factor (correcting for these multiple tests using one possible correction)) using pairwise.t.test.

```
pairwise.t.test(Part.temp$Julian, Part.temp$AGEF, p.adjust.methods="holm")
```

```
##
```

```
##  Pairwise comparisons using t tests with pooled SD
##
## data:  Part.temp$Julian and Part.temp$AGEF
##
##   1      2      3      4      5      6      7
## 2 0.5083 -      -      -      -      -      -
## 3 0.0017 1.0000 -      -      -      -      -
## 4 0.0015 0.7056 1.0000 -      -      -      -
## 5 1.0000 1.0000 1.0000 1.0000 -      -      -
## 6 0.0216 0.9322 1.0000 1.0000 1.0000 -      -
## 7 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -
## 8 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
##
## P value adjustment method: holm
```

So note that 3, 4 and 6 yr olds are significantly different from yearlings but that there aren't any other significant differences.

OR

We can specify our model slightly differently using the aov command instead of the lm command. In this case we can use what might be a more familiar technique for conducting post hos tests using Tukey's Honest Significant Difference test.

```
age.aov<-aov(Julian~AGEF, data=Part.temp, na.action=na.omit)
TukeyHSD(age.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Julian ~ AGEF, data = Part.temp, na.action = na.omit)
##
## $AGEF
##            diff        lwr       upr     p adj
## 2-1  -4.6706963 -10.774511  1.433119 0.2817367
## 3-1  -8.2374779 -14.458938 -2.016018 0.0015822
## 4-1  -9.0665143 -15.846408 -2.286621 0.0013425
## 5-1  -4.0861916 -12.268556  4.096173 0.7989918
## 6-1 -11.3022120 -21.545074 -1.059350 0.0188142
## 7-1  -6.7912808 -24.213093 10.630531 0.9367436
## 8-1  -1.3175966 -43.746834 41.111641 1.0000000
## 3-2  -3.5667816  -9.061716  1.928153 0.5024335
## 4-2  -4.3958180 -10.515842  1.724206 0.3642825
## 5-2   0.5845047  -7.060024  8.229034 0.9999982
## 6-2  -6.6315157 -16.450067  3.187036 0.4482084
## 7-2  -2.1205845 -19.296360 15.055191 0.9999514
## 8-2   3.3530997 -38.975708 45.681908 0.9999977
## 4-3  -0.8290364  -7.066400  5.408327 0.9999199
## 5-3   4.1512863  -3.587501 11.890074 0.7330769
## 6-3  -3.0647341 -12.956850  6.827382 0.9820314
## 7-3   1.4461971 -15.771738 18.664132 0.9999965
## 8-3   6.9198813 -35.426051 49.265814 0.9996799
## 5-4   4.9803227  -3.214140 13.174785 0.5892748
## 6-4  -2.2356976 -12.488227  8.016831 0.9978980
## 7-4   2.2752335 -15.152264 19.702731 0.9999291
## 8-4   7.7489177 -34.682655 50.180490 0.9993331
## 6-5  -7.2160203 -18.445378  4.013338 0.5160258
## 7-5  -2.7050892 -20.724573 15.314395 0.9998183
## 8-5   2.7685950 -39.909532 45.446722 0.9999994
## 7-6   4.5109312 -14.532885 23.554747 0.9964530
## 8-6   9.9846154 -33.136002 53.105233 0.9969249
## 8-7   5.4736842 -39.891401 50.838769 0.9999585
```

Note that the results are similar but not exactly the same. We will not spend much time on post hoc tests (to the disappointment of some of you).

4 Model Diagnostics in R

General linear models have assumptions associated with what is called the 'structural part of the model' as well as with the distribution of errors (residuals).

Assumptions from the structural component of the model:

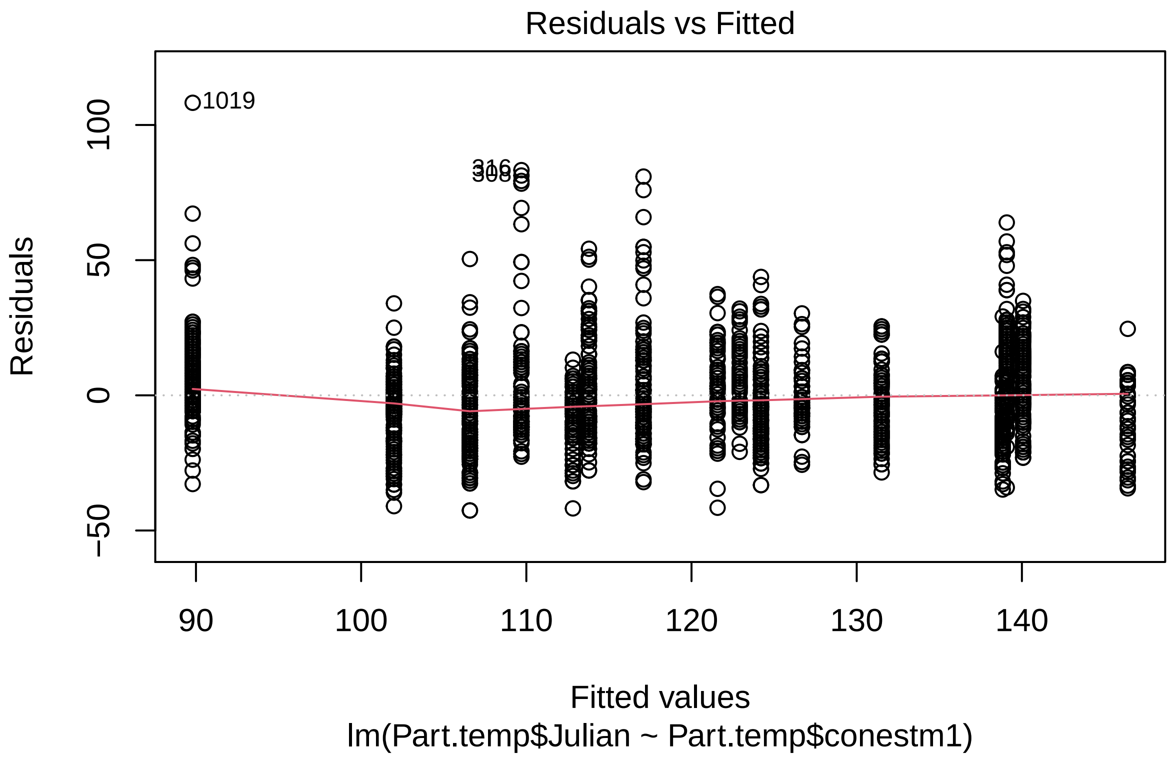
- Y is a continuous variable
- Y depends on between 1 and many predictors variables, X's
- Y is linearly related to the X's. This one requires some further clarification. By this we mean that we can specify Y as being predicted by a linear combination of X's. It is perfectly fine to transform either Y or X's to linearize the relationships (e.g. $Y = log(X)$). It is also possible to account for some non-linear relationships between X and Y using quadratic terms. We will talk more about this later. What cannot be handled in general linear models are relationships that cannot be linearized (e.g. Von Bertalanffy growth curves, Michaelis-Menton kinetics). In these cases we would need to use another, explicitly nonlinear, approach.

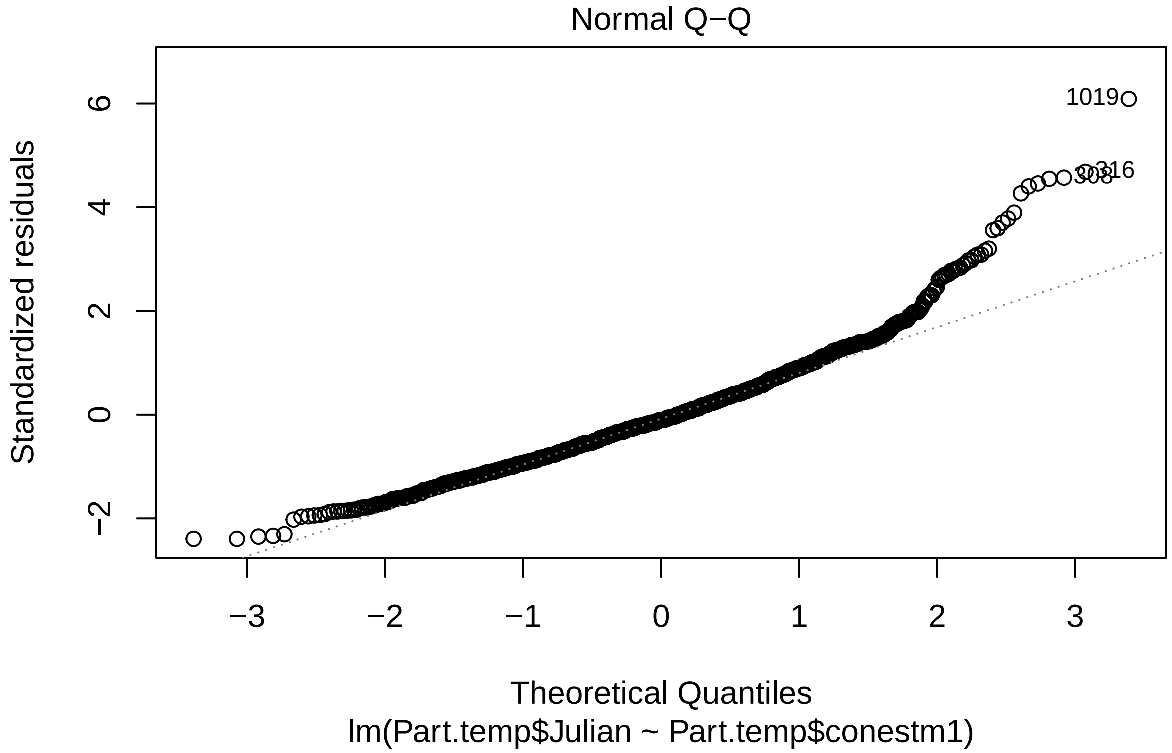
The next 4 assumptions are the ones that we are most often concerned with and these all deal with the residual part of the model. Remember that in a general linear model we assume that the residuals come from a normal distribution with a mean of zero and some constant variance. This basic assumption lies at the heart of the specific assumptions that we want to assess.

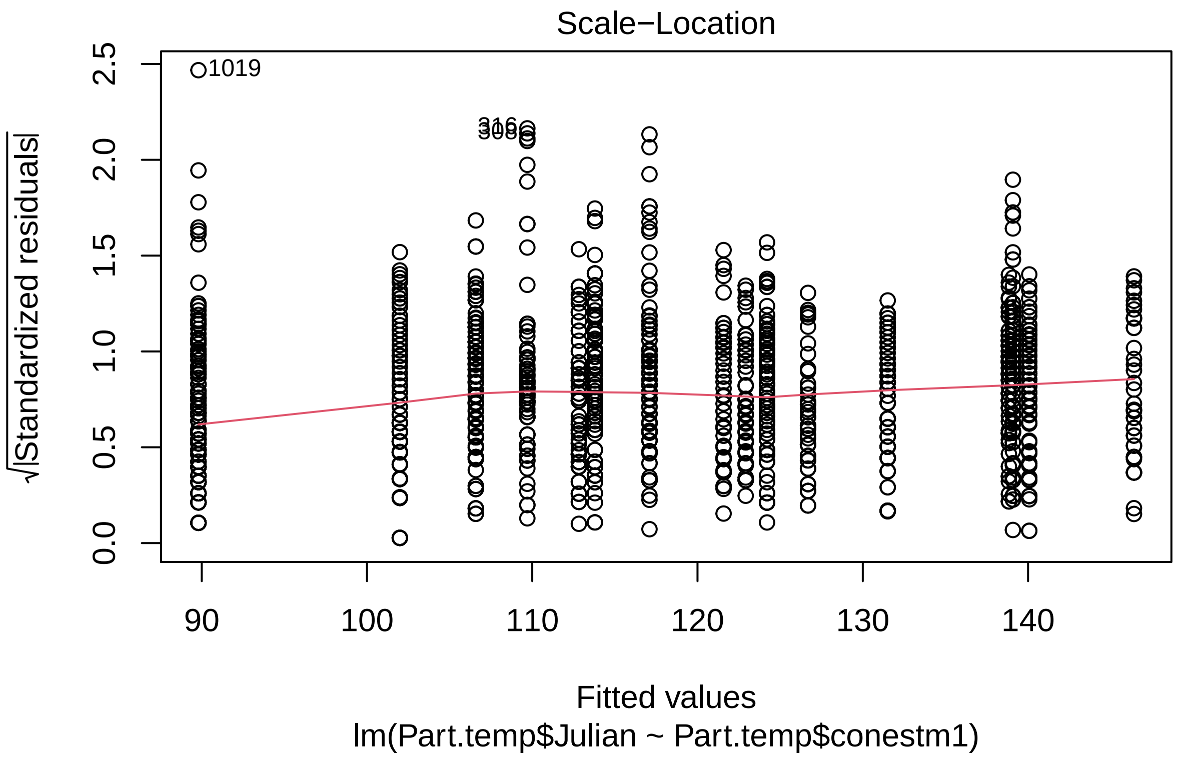
- Errors (residuals) are normally distributed.
- Variance in errors remain constant. This is referred to *homoscedasticity*. The opposite of this, which is a problem, is called *heteroscedasticity*. This means that the residual variance is not constant, but instead varies with predicted values or across levels of a factor.
- Errors are independent of one another. Refer to the replication handout for more information on independence and lack of independence of observations (i.e. pseudoreplication). We will deal with this again later in the course when we discuss mixed effect models.
- Predictors are independent of one another. If predictors are correlated this is referred to as *collinearity*. We will deal with this more later. For now, it is enough to know that this causes problems for the model in trying to sort out the unique effect of predictor A and predictor B on some response variable, Y, when A and B are correlated.

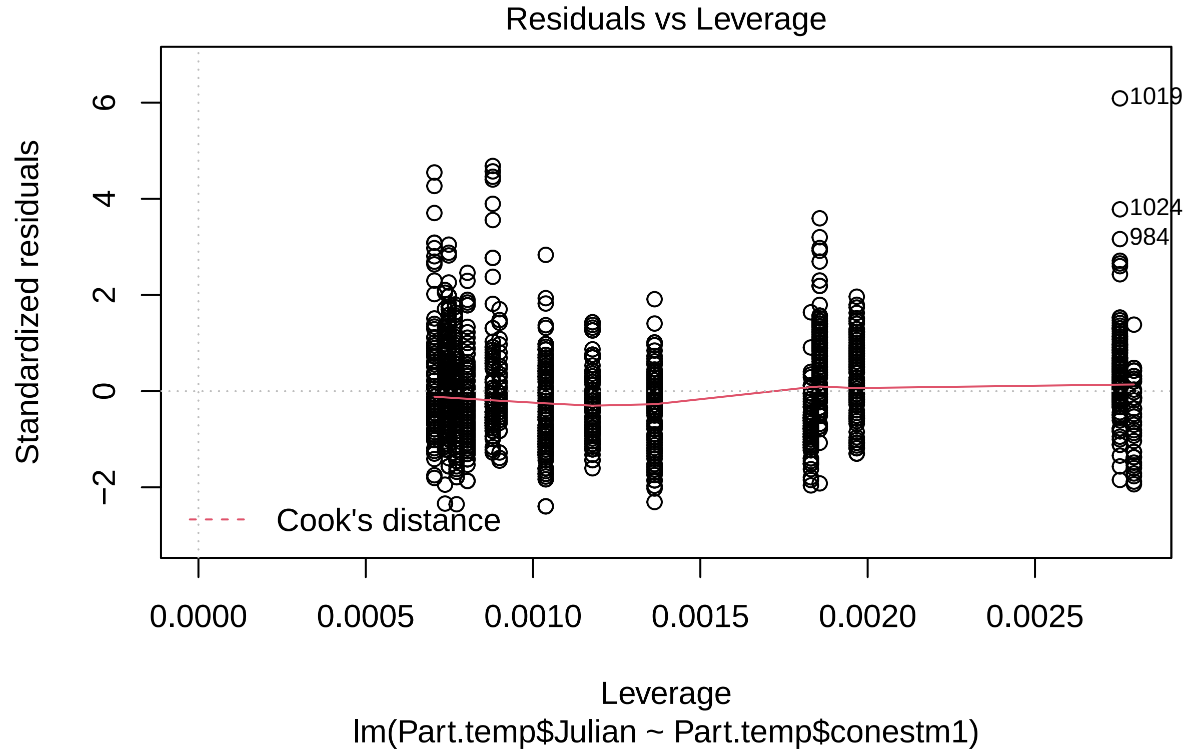
Rather than assessing the validity of these assumptions by using some rigid test, we will instead assess the assumptions of the model visually by looking at diagnostic plots. We can easily get a graphical display of model diagnostics using the plot command. In your R session, try and use the following commands. Note that in some cases you will need to remove the #.

```r
plot(julian.lm)
```


Residuals vs Fitted
Residuals
100
50
0
−50
1019
316
308
90
100
110
120
130
140
Fitted values
lm(Part.temp$Julian ~ Part.temp$conestm1)

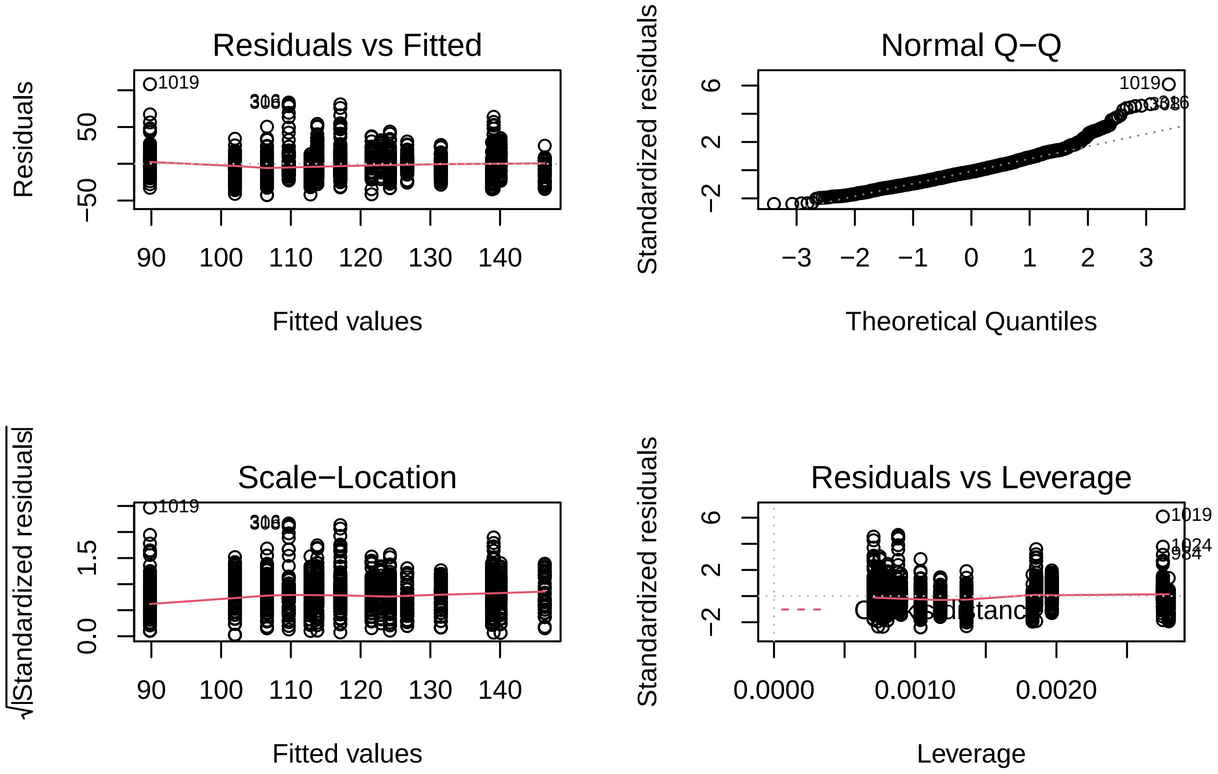

Normal Q−Q
Standardized residuals
6
4
2
0
−2
1019
308
316
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Part.temp$Julian ~ Part.temp$conestm1)





It's sometimes easiest if all 4 plots are placed on the same page. We can do this by specifying that we want teh plots to be arranged in 2 rows and 2 columns by specifying mfrow=c(2,2) in the par command:

```r
par(mfrow=c(2,2))
plot(julian.lm)
```

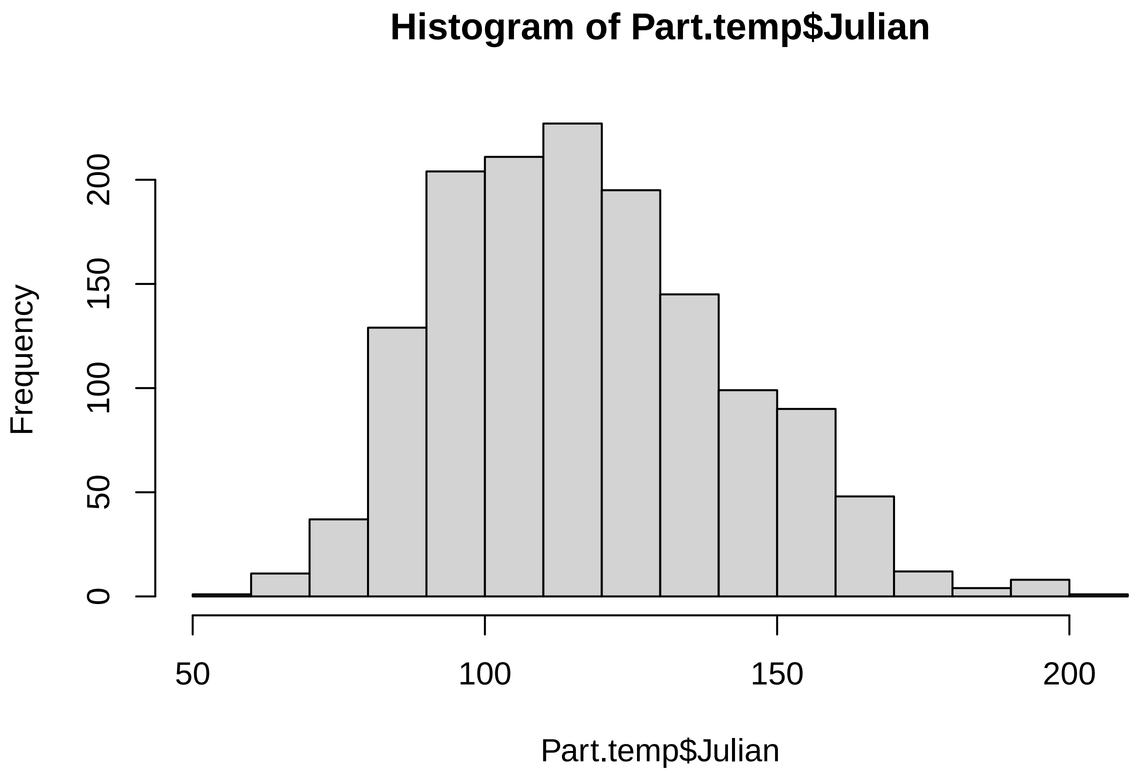


So what are we looking at here? These are 4 diagnostic plots that are designed to assess graphically whether or not the assumptions of general linear models have been met. These are not done through some rigid test, but are instead assessed qualitatively. As a result this takes some thought and some practice to interpret them. We will start with the assumption of normality.

4.1 Normality

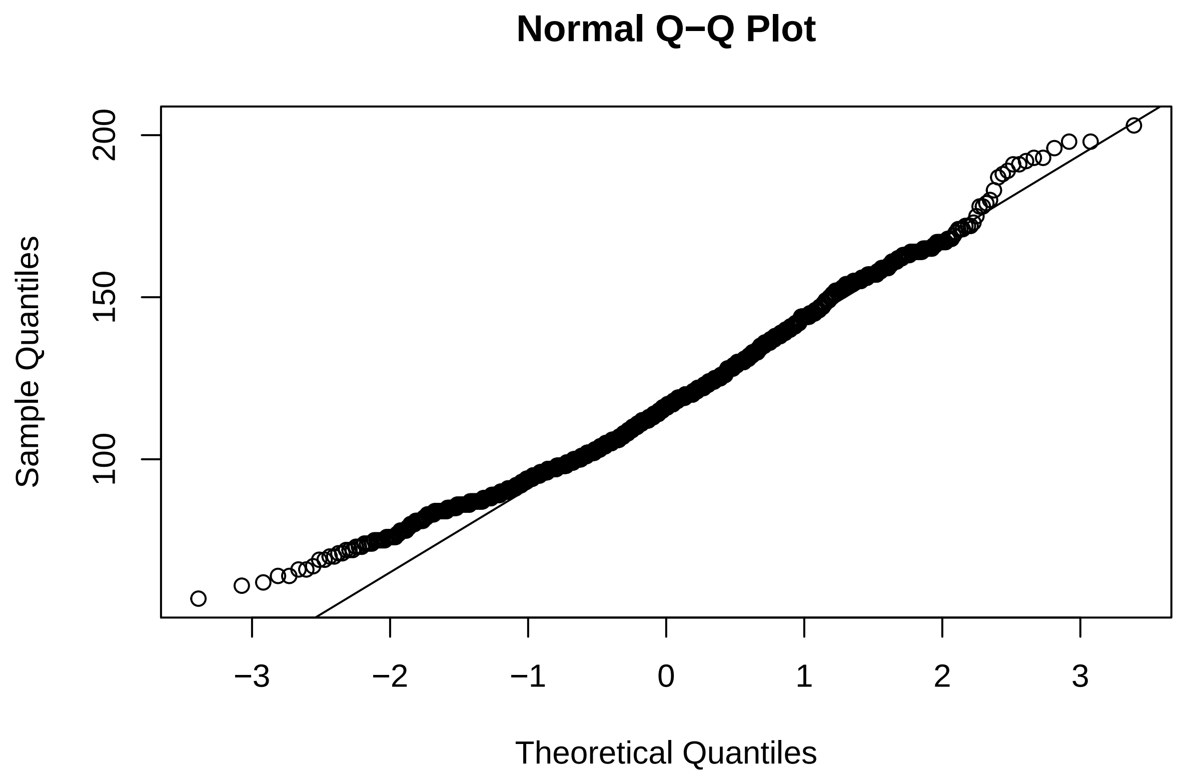
We can first look at a histogram of the response variable…

```
par(mfrow=c(1,1))
hist(Part.temp$Julian)
```



This is a familiar way for looking for 'normality'. R often uses a different type of plot that is informative but a bit trickier to assess, called called quartile-quartile plots (or qq plots). These plot the observed data against the expected data given an assumed distribution (e.g. normal). For a 'normal' qq plot we would use the qqnorm function.

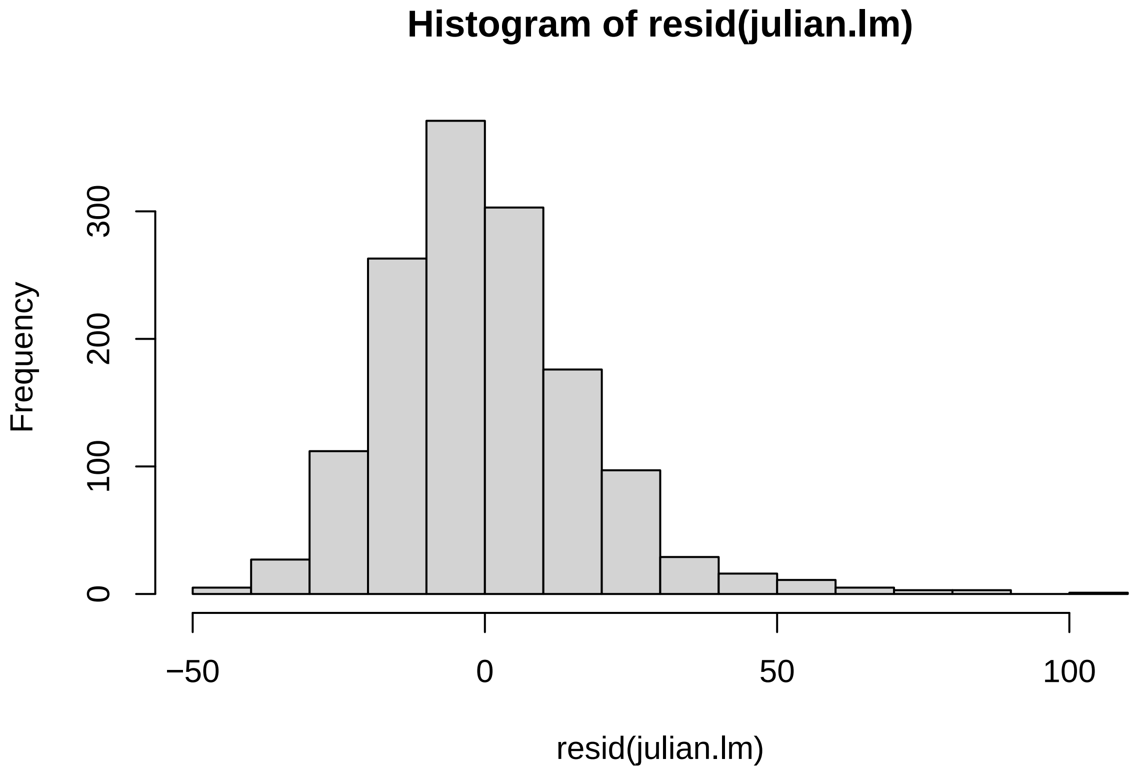
```
qqnorm(Part.temp$Julian)
qqline(Part.temp$Julian)
```



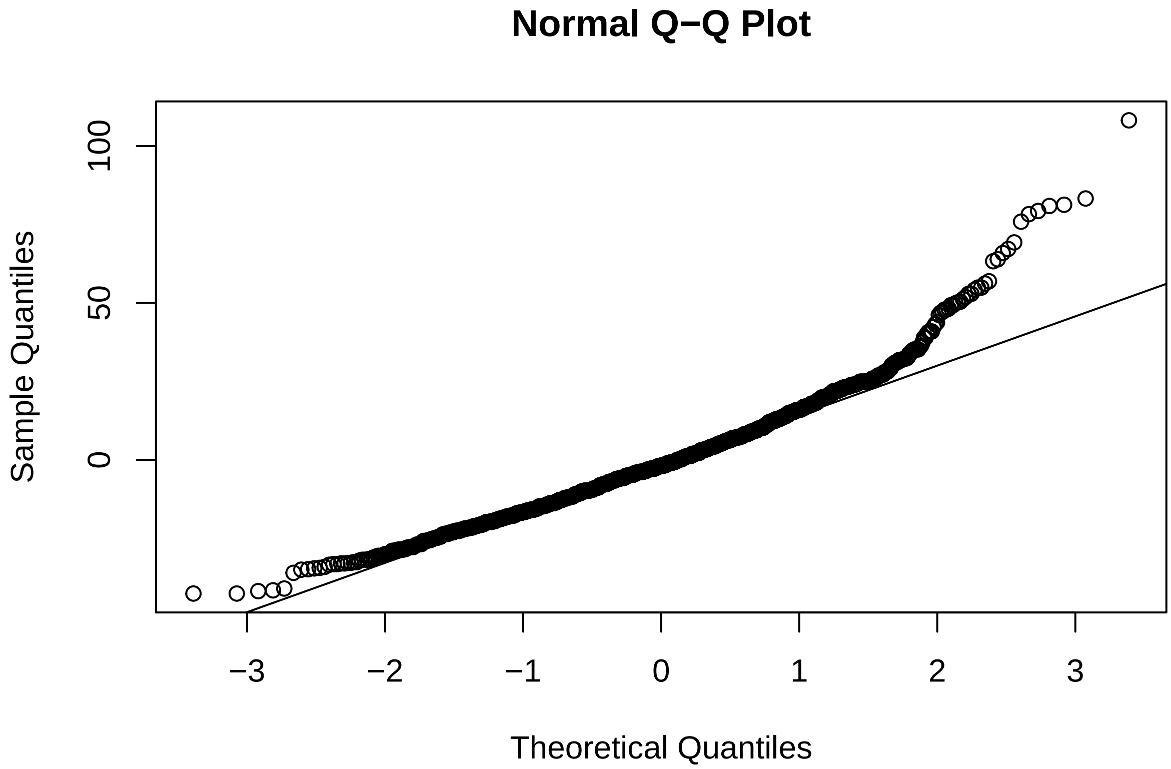
Note that the qqnorm plots the data and the qqline adds the line.

Remember though that we are actually interested in the residuals of the model and not the raw data. The assumptions of linear models pertain to the residuals and NOT the raw data themselves.

```
hist(resid(julian.lm))
```

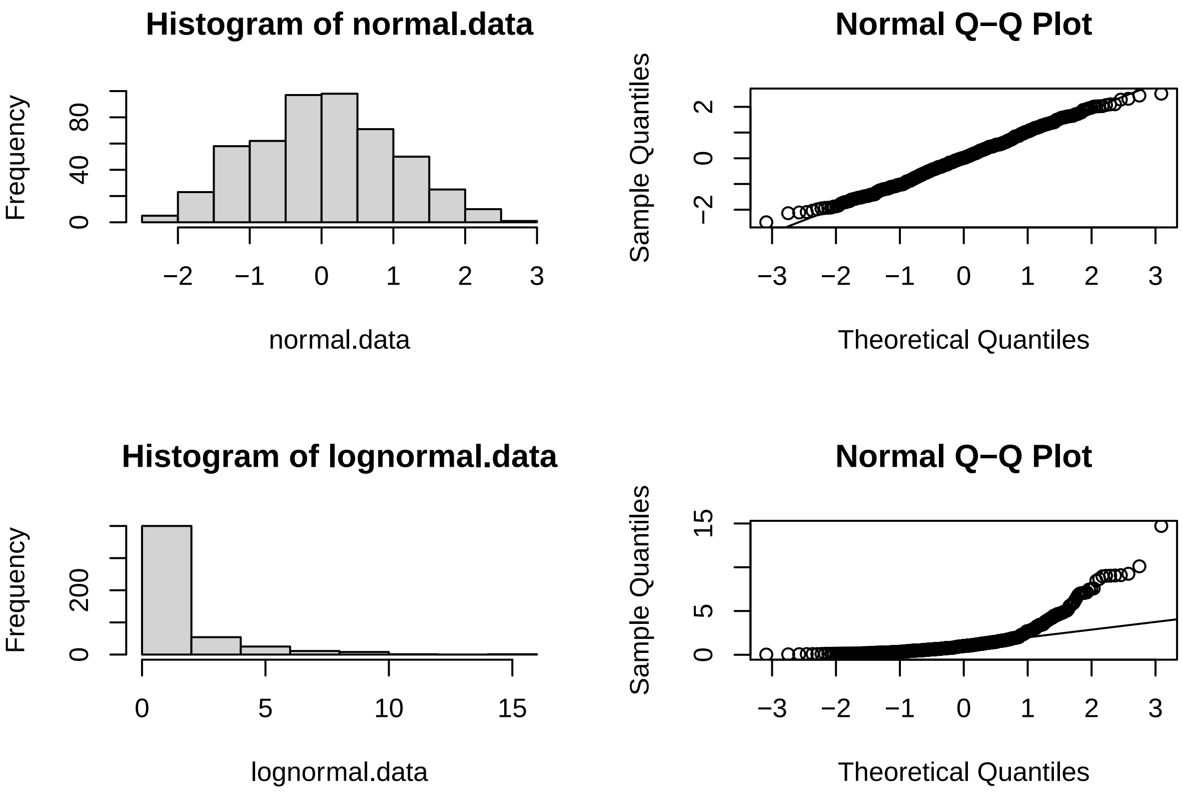


```
qqnorm(resid(julian.lm))
qqline(resid(julian.lm))
```

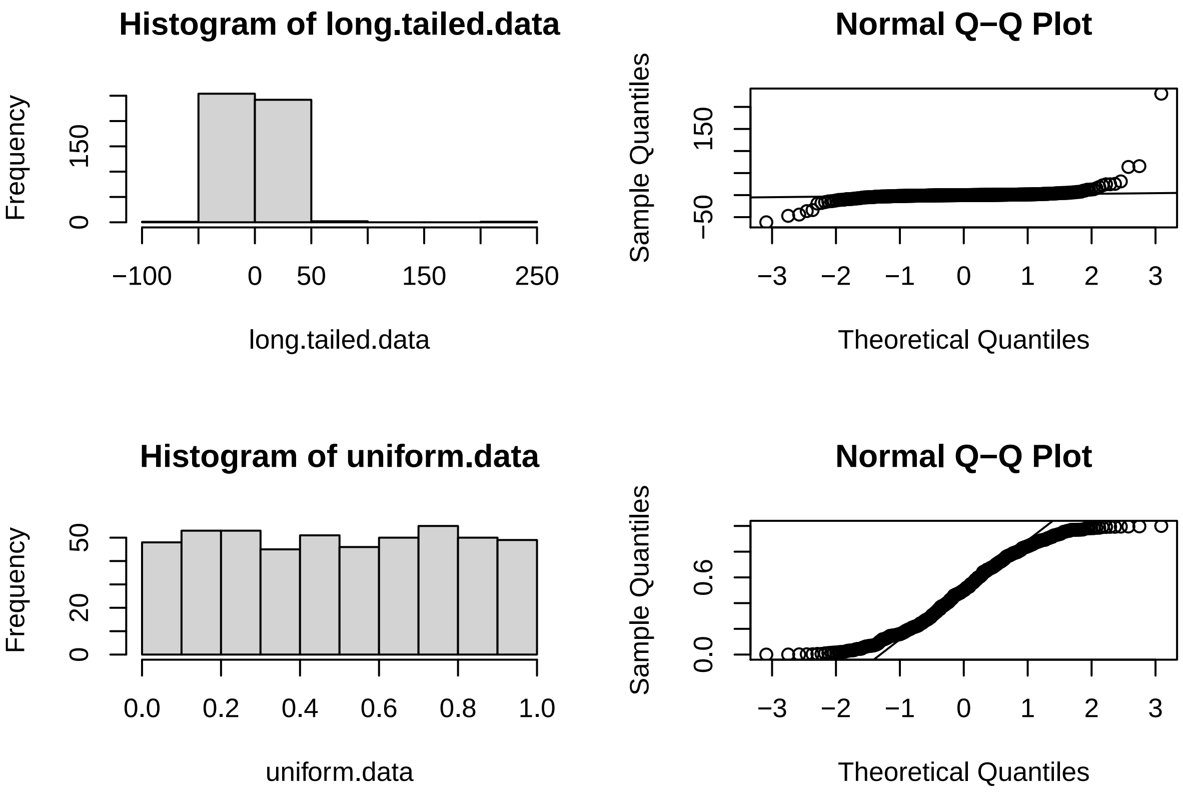


While most people have looked at histograms of residuals before, these qq plots are likely new to anyone who hasn't used R. As a result, it is hard to know what you are looking at and what you should be looking for. We can compare these results to what we would expect if the data came from a normal distribution or some other common distributions…

```
normal.data<-rnorm(500)
lognormal.data<-exp(rnorm(500))
long.tailed.data<-rcauchy(500)
uniform.data<-runif(500)

par(mfrow=c(2,2))
hist(normal.data)
qqnorm(normal.data)
qqline(normal.data)

hist(lognormal.data)
qqnorm(lognormal.data)
qqline(lognormal.data)
```



```r
hist(long.tailed.data)
qqnorm(long.tailed.data)
qqline(long.tailed.data)

hist(uniform.data)
qqnorm(uniform.data)
qqline(uniform.data)
```



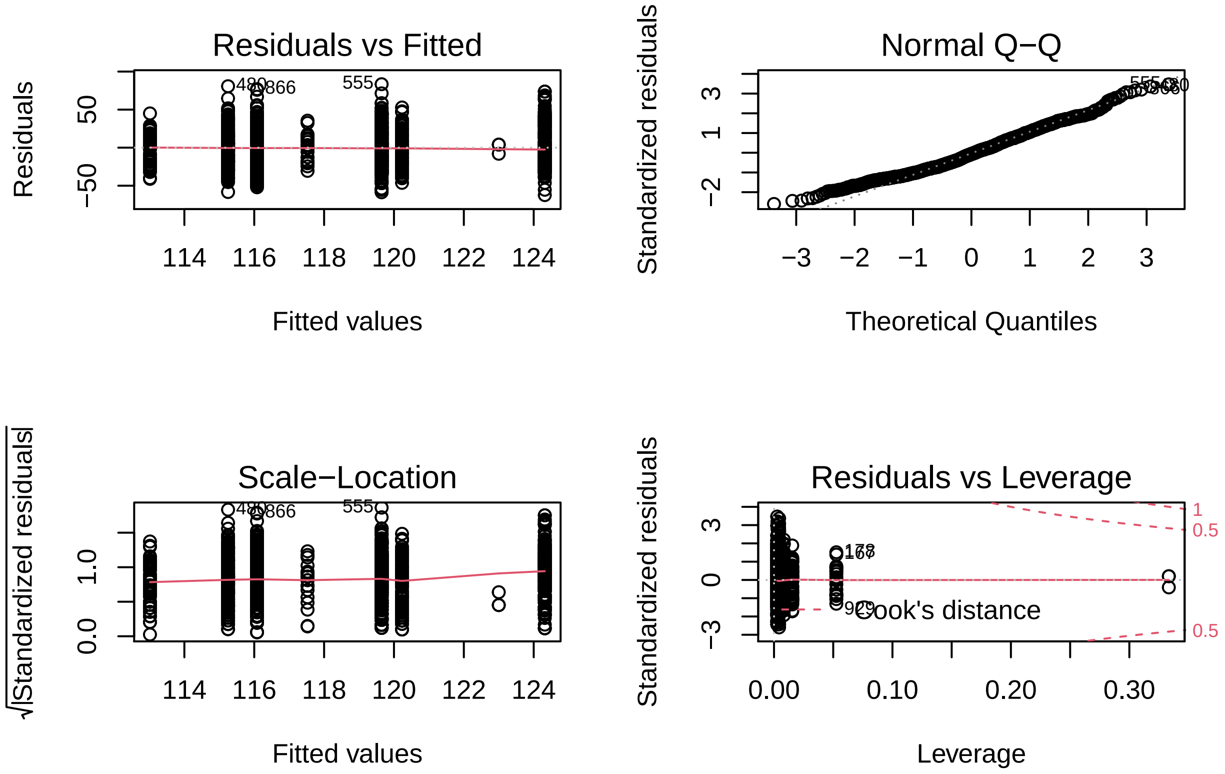
```r
par(mfrow=c(1,1))
```

I have plotted the historgrams for the simulated data next to their q-q plot so that you can get used to seeing how data appear in these q-q plots. See that for the normal data most of the points all fall along the straight line. It is common for there to be a little bit of curvature in this plot at the ends of the line. Now compare these figures to the long-normal data where there is a long tail to the right. See how the observed data fall well above the expected line for larger values. This means that large values are much larger than expected by a normal distribution. The long-tailed (platykurtic) data are a bit harder to see in the histogram, but note that small values fall below the lines whereas large values fall well above the line. This means that small values are smaller than expected and large values are larger than expected by a normal distribution. Finally in the uniform distribution (short-tailed or leptokurtic) the small values are larger than expected (above the expected line) and large values are smaller than expected (below the line). Remember that if we overlay two normal distributions with different means, the result is a leptokurtic distribution. So residuals with short-tails might indicate that maybe there is some underlying factor (difference) in your residuals that you have not yet accounted for.

Now compare these figures to the one above for the residuals of the julian.lm model. See how the residuals of the model largely fall along the straight line (they are largely normal), but that at high positive values of the residuals the observed values (sample quartiles) are greater than what we would expect (theoretical quartiles). This means that the positive tail of the distribution of residuals is a bit longer than we would expect based on a normal distribution. If you refer back to the histogram of the residuals for this model you can see that the right tail of the distribution of residuals is a bit longer than what we would expect to see.

We can also look at the residuals of the Age model

```r
par(mfrow=c(2,2))
plot(age.model)
```



QUESTION: Why are the predicted values all stacked up???

You can of course do a 'test' for normality of your residuals.

```
shapiro.test(residuals(julian.lm))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(julian.lm)
## W = 0.9535, p-value < 2.2e-16
```

Many of you might be tempted to think that such an 'objective' test is superior than simply eye-balling a diagnostic plot. The problem here, however, is that a large model with lots of data will have lots of 'power' to detect departures from normality, whereas a smaller dataset might have weak power to detect departures from normality in the residuals. This single test also doesn't provide you with any tips on the 'way' in which your residuals might deviate from normality. Viewing the q-q plot and seeing the distribution of the residuals will provide some suggestions on how you might deal with the non-normality, either through transformation or perhaps through an improvement in your model to account for structure in your residuals.

So what if the residuals of your model are not normal? In many cases linear models are robust to violation of the assumption of normality. As long as you have a roughly symmetrical distribution you are probably OK. If you have a heavily skewed distribution then you are potentially going to be too liberal in your test (too likely to reject the null), which is a potential problem. In this case you can consider transforming the data (e.g. log transform) to normalize the residuals, perhaps use a generalized linear model instead (see future classes) or perhaps use non-parametric tests.

4.2 Homoscedasticity

Homoscedasticity refers to the assumption of general linear models that variance in the residuals remains constant. That is, that residuals are not more or less variable for some predicted values than others. If variance in the residuals is related to some other quantity then we might be less certain about what a particular parameter is than we suspect based on the assumption of some common variance. So for example, if we have many groups in an ANOVA type analysis then we test for differences among groups based on the assumption that there is a common variance in observations within each group. If instead, residuals are more variable in some groups than others (heteroscedasticity) then we might be less confident about differences between some more variable groups than we would be if variances were constant across groups.

We cannot assess whether there is heteroscedasticity in our model by looking at the residuals alone. Instead we need to plot the residuals against some aspect of the model. If there are categories in our model (like an ANOVA) then we can look at box plots of residuals for each level of the category.

For example, look at the variance in the residuals of our age model of parturition dates across age categories:

NOTE: That here if we used:

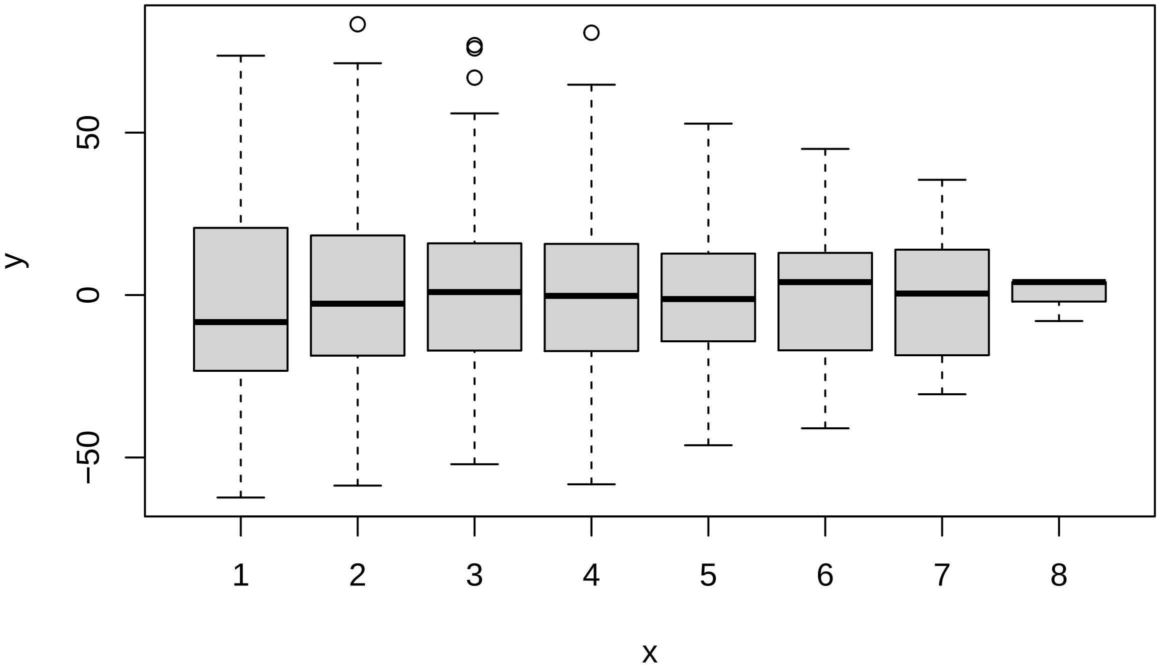
par(mfrow=c(1,1)) plot(Part.temp$AGEF, resid(age.model))

We would get…

Error in model.frame.default(formula = y ~ x) : variable lengths differ (found for 'x')

We get an error because there were excluded missing data in our model for either age or Julian so the residuals vector is shorter than the AGEF vector

```
par(mfrow=c(1,1))
plot(Part.temp$AGEF[!is.na(Part.temp$AGEF)&!is.na(Part.temp$Julian)], resid(age.model))
```



There seems to be large differences among age categories in variance. Younger ages seem to have much more variance in their residuals than older ages. The linear model is trying to make inferences among ages based on a common within-age variance, but it is clear that using this average variance might be appropriate for comparisons between ages 4 and 5, but that there is much more uncertainty about what the breeding date of 1 year-olds is (more variance in residuals) than would be represented by a common within-age variance.

We can test this more formally using a Levene test. I have found a simple function for performing a Levene test for heterogeneity of variance.

```
 Levene<-function(y, group){
#Small function for performing Levene test for homogeneity of variance among groups.
#Written by Brian Ripley and copied from the R-help mailing list.
#Mean can be replaced with median for a more robust test
 group<-as.factor(group) #Precautionary
 meds<-tapply(y, group, mean, na.rm=T)
 resp<-abs(y-meds[group])
 anova(lm(resp~group))
 }
 Levene (Part.temp$Julian, Part.temp$AGEF)
```

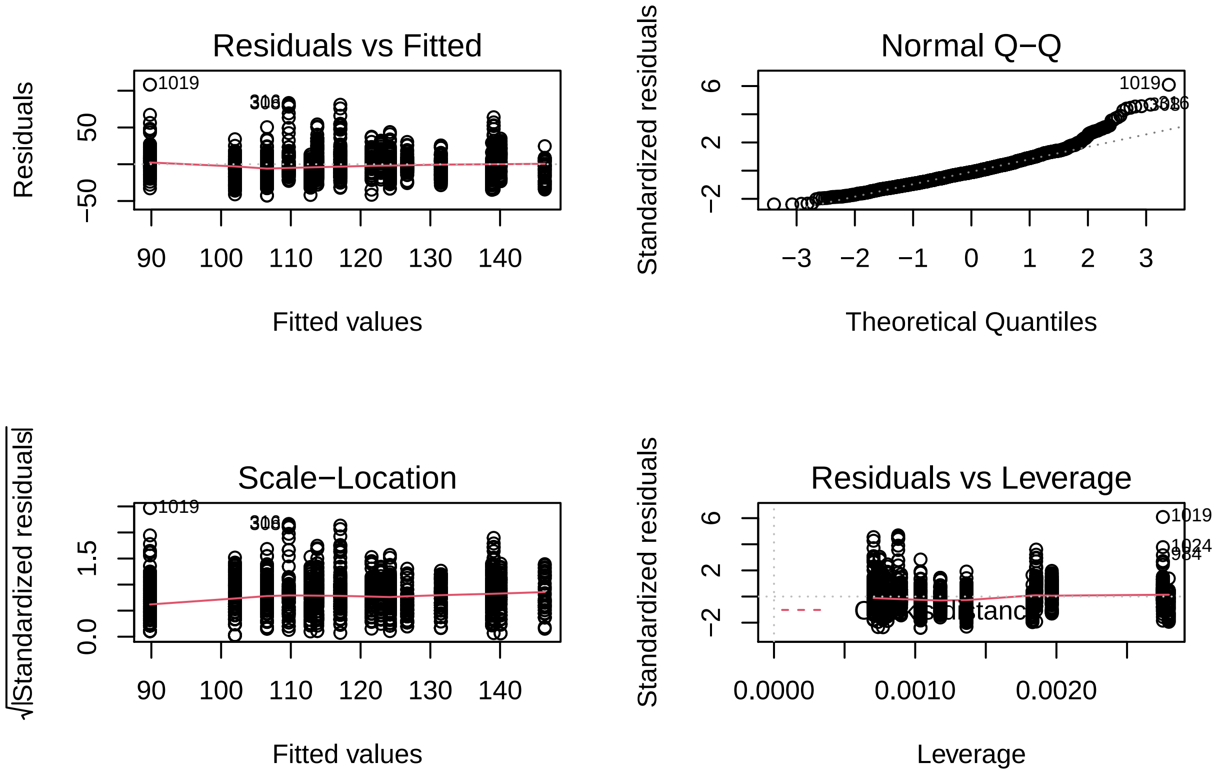
```
## Analysis of Variance Table
##
## Response: resp
##             Df Sum Sq Mean Sq F value   Pr(>F)
## group        7   7468 1066.81  5.8576 1.02e-06 ***
## Residuals 1372 249873  182.12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Oh boy!! So we clearly have a problem with our errors.

In general linear models we may or may not have categorical predictors to assess heteroscedasticity in this way. In addition, this is only one way in which variance in residuals might differ. A much more general way in which to assess heteroscedasticity is to plot the residuals against the predicted values of the model.

So for our model of breeding date by red squirrels we can look at the first diagnostic plot.

```
par(mfrow=c(2,2))
plot(julian.lm)
```



Look at the panel in the top left. The residuals should appear as a cloud of points. It is natural for them to take on an ellipse-like shape where there is a bit more variation in the residuals for intermediate predicted values. Otherwise there should not be large changes in the 'spread' of the residuals across predicted values. If you see a diagnostic plot where the residuals take on a funnel shape (much more spread in residuals for small or more commonly large predicted values) then this is an example of heteroscedasticity that needs to be dealt with.

In an ANOVA-type design then linear models are generally robust to heteroscedasticity if the design is relatively balanced (approximately the same number of observations for each group). In a regression-type analysis, heteroscedasticity will mean that one side of the relationship figures more heavily into the estimates than the others. Often a good transformation will fix problems of heteroscedasticity.

4.3 Outliers

A normal distribution assumes that tails of the distribution are 'small'. Extreme observations (for example those greater than 3 sd from the mean) should occur so rarely that in practice we should not observe them. Such extreme observations have the potential to heavily skew our results. These are often called outliers. Residuals can be detected by plotting residuals. It is important to note that these could occur through data collection or entry errors. They could also be 'real' but result from extreme circumstances. If we have adequately explained these circumstances in our model then an extreme observation will not translate into an extreme residual. If we have not adequately accounted for these extreme circumstances in our model then the extreme observation will result in a very very large (positive or negative) residual.

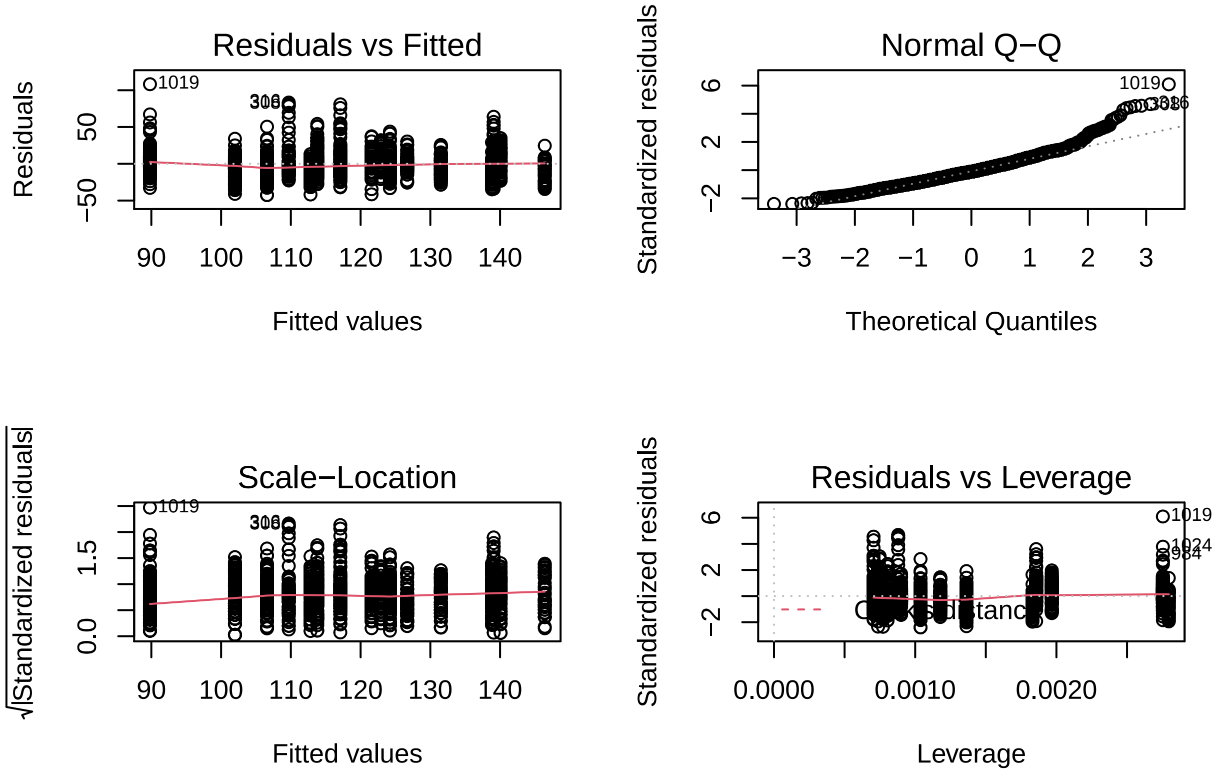
What to do about outliers? Once outliers have been identified, revisit the paper trail of data collection. Check original unmanipulated data files as well as field/lab notebooks. If this was simply a human error in data entry then simply correct the error. If you cannot confirm an error then you need to proceed cautiously. You have the potential to be misled either by including or excluding this unusual observation. NOTE THAT IT IS UNETHICAL TO REMOVE AN OUTLIER FROM AN ANALYSIS WITHOUT BEING TRANSPARENT ABOUT THIS TO YOUR READERS!

Outliers certainly increase our uncertainty around our parameter estimates, but they do not necessarily bias these parameter estimates. Bias results from a combination of a large residual and high leverage. Recall that residuals are simply the deviation between the observed and the predicted value. The 'leverage' of a point is simply how far that observation's predicted value is from the mean predicted value. So in a plot of observed residuals versus predicted values, leverage is measured by how far that observation is to the left or the right of the mean predicted values along the x axis.

You might not yet know, but a regression between X and Y always passes through the mean value of X and the mean value of Y. I then like to think of the slope of this relationship like a teeter-totter. Metaphorically, the magnitude of the residual is like the weight of the kid sitting on the teeter-totter. The leverage of the observation is like how far away from the fulcrum the child is sitting. An observation could have a massive residual, but if it has no leverage (i.e. has a predicted value equal to the mean) then this large residual will have had no effect on the parameter estimate. In contrast, an observation might have an extreme predicted value because of an extreme value for a single or set of predictor variable. This very high leverage means that an extreme observation for this point could have a very strong effect on parameter estimates (i.e. the slope in our simple regression teeter-totter example).

We can assess the effect that each observation has on the model parameters using what's called a Cook's Distance. These can be visualized as a surface plot of Residual against Leverage. Observations with high leverage and extreme residuals will have undue influence on relationships. This is the bottom right plot in our standard set of diagnostic plots. I believe that a Cook's distance represents the expected change in parameter value (in units of SEs) that would be expected from the removal of this observation. So Cook's distances less than 1 are not a problem. Cook's distances greater than 2 are a problem! This diagnostic plot is a contour plot. If you don't see any contours then this is a good thing! All your observations fall below the minimum Cook's distance for the first contour (I think this is 0.5??).

```r
par(mfrow=c(2,2))
plot(julian.lm)
```



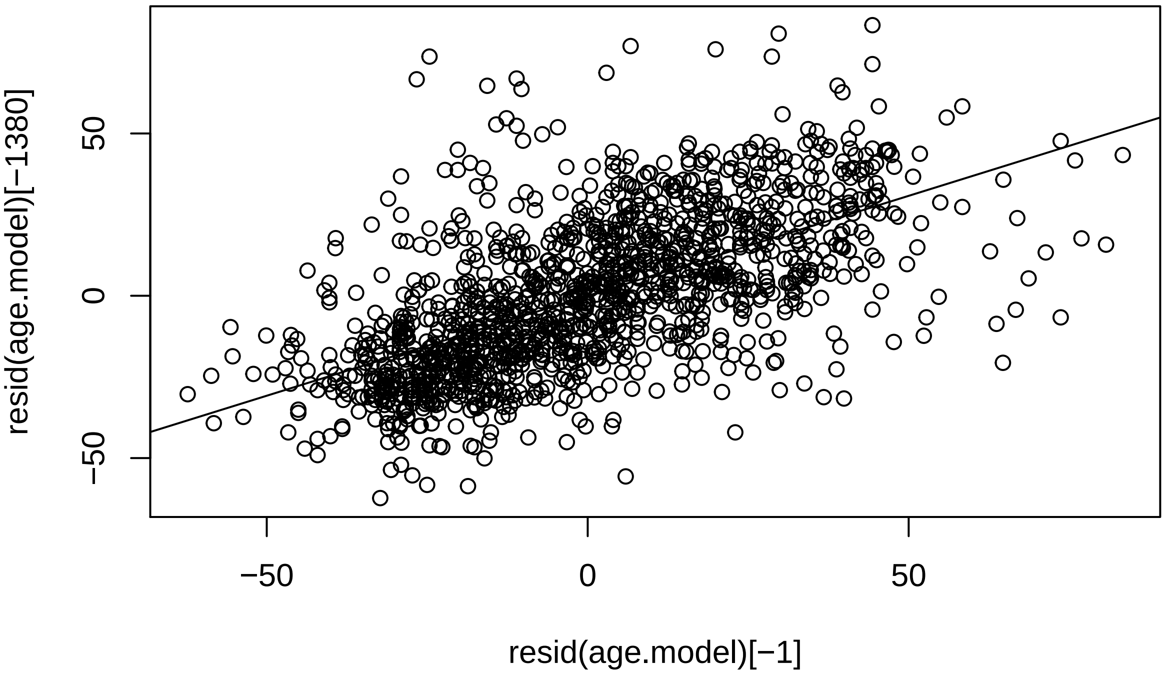
4.4 Independence of residuals

Observations must be independent of one another. One way to test for independence of sequential observations is to plot an autocorrelation of observations. We can also test for autocorrelation in the residuals by…

```
length(resid(age.model))
```

```
## [1] 1380
```

```
plot(resid(age.model)[-1], resid(age.model)[-1380])
abline(lm(resid(age.model)[-1380]~resid(age.model)[-1]))
```



It helps if you put a line through the residuals.

They certainly don't look independent!!! This means that the residual for each observation is highly correlated with the residual for the previous observation. Part of the reason for this is that we have collected several data points within a year and years certainly aren't independent of one another (some years have earlier breeding than others - note that this was also evident from the stacking-up of residuals in the diagnostic plots). Note also that the dataset is sorted by Year so that residuals next to one another tend to come from the same year.

We will deal with this when we talk more about random effects and mixed effects models.

5 Example: 'ANCOVA' model of hatchling mass of lizards

I am going to switch things up a bit and talk about a new dataset called *eggtemp.csv*. In this datafile we have observations of the size of lizard eggs (PostEMass in grams), whether or not the egg experienced an experimental removal of yolk (EggCode: C is control and M is miniaturized), whether the hatchling that emerged from the egg was Male (M) or female (F) and the size (SVL in mm) and mass (H.Mass in grams) of the hatchling.

```
eggtemp<-read.table("data/eggtemp.csv", header=T, sep=",")
eggtemp$Sex<-as.factor(eggtemp$Sex)
eggtemp$EggCode<-as.factor(eggtemp$EggCode)
summary (eggtemp)
```

```
##  EggCode   PostEMass       Sex         H.Mass            SVL
##  C:635   Min.   :0.1600   F:516   Min.   :0.2400   Min.   :19.00
##  M:351   1st Qu.:0.3600   M:470   1st Qu.:0.5100   1st Qu.:24.00
##          Median :0.4100           Median :0.5700   Median :25.00
##          Mean   :0.4129           Mean   :0.5632   Mean   :24.56
##          3rd Qu.:0.4600           3rd Qu.:0.6300   3rd Qu.:25.00
##          Max.   :0.9100           Max.   :0.8000   Max.   :28.00
```

We can build a linear model that explains variation in hatchling mass based on its body size, whether it was male or female and whether it was miniaturized

```
egg.lm<-lm(H.Mass~EggCode+Sex+SVL, data=eggtemp, na.action=na.omit)
summary (egg.lm)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode + Sex + SVL, data = eggtemp, na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28708 -0.03925 -0.00128  0.03343  0.34148
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.5950993  0.0394144 -15.099   <2e-16 ***
## EggCodeM    -0.0420225  0.0041881 -10.034   <2e-16 ***
## SexM         0.0005127  0.0036829   0.139    0.889
## SVL          0.0477565  0.0015821  30.186   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05752 on 982 degrees of freedom
## Multiple R-squared:  0.6072, Adjusted R-squared:  0.606
## F-statistic: 506.1 on 3 and 982 DF,  p-value: < 2.2e-16
```

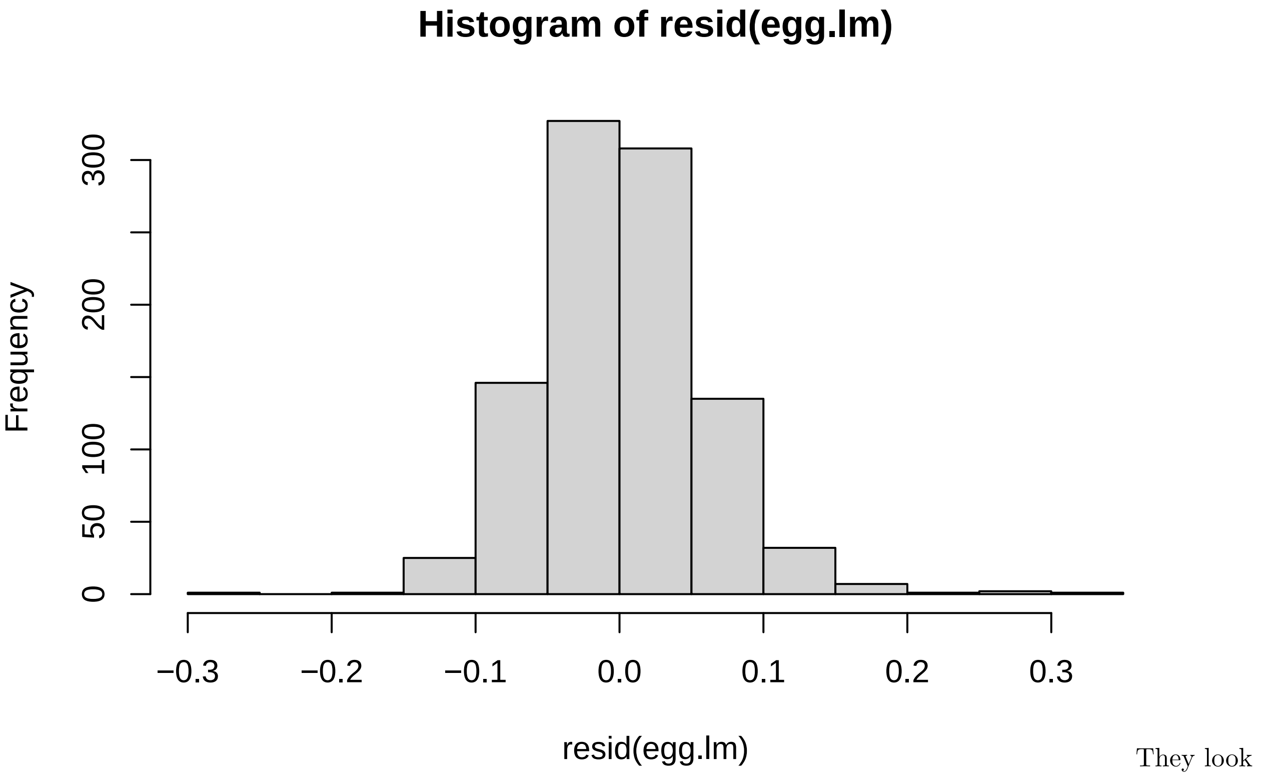
Pay attention to the parameters above and think about what they mean. Would you be able to figure out what the expected mass of a male hatchling with a hatchling body size of 25mm and no experimental manipulation would be?

Look at the significance of the overall model and of each parameter individually.

OK so lets look at some of the diagnostic plots for this model

Lets first look at the distribution of the residuals using a histogram

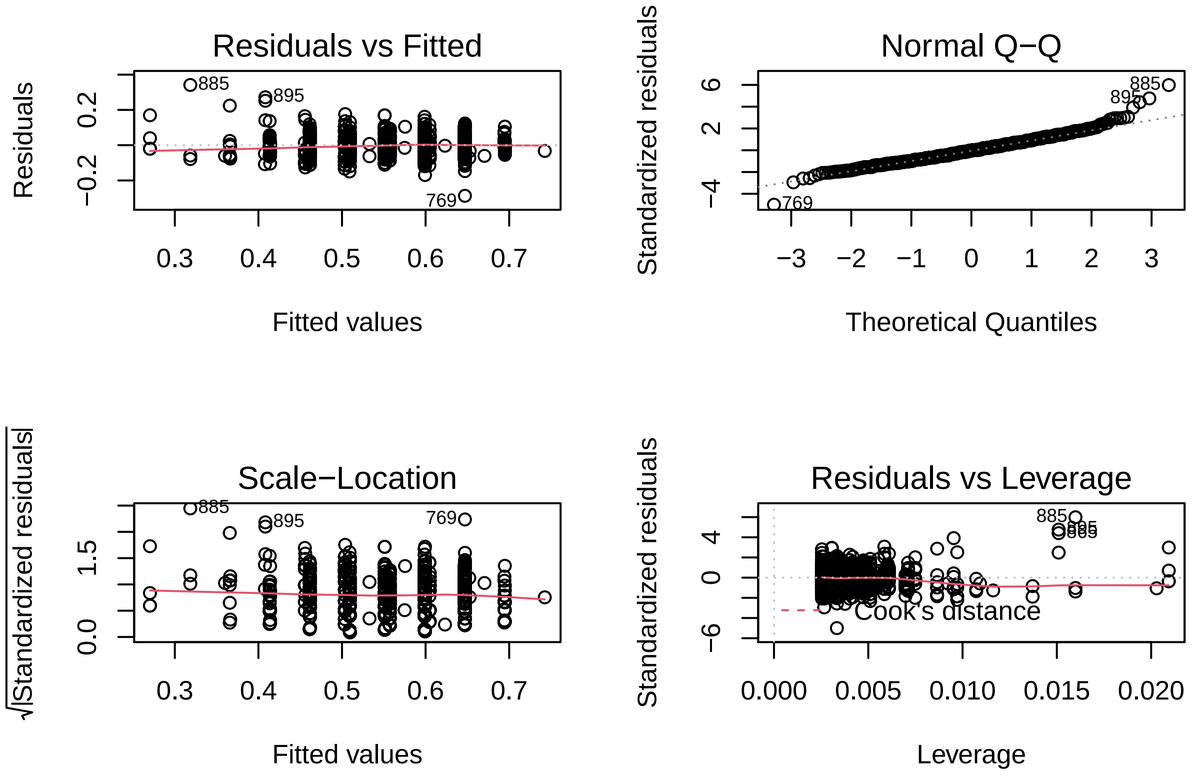
```
hist(resid(egg.lm))
```



They look quite good

We can assess several residual plots at the same time using the plot command

```
par(mfrow=c(2,2))
plot(egg.lm)
```



We can see from this that in general the diagnostic plots look quite good. Note that in the first plot the variance in residuals does not seem to change much from low to high predicted values. So we don't seem to have problems with heteroscedasticity overall. There are however a couple of observations that are repeatedly identified as standing out from the others. We might want to rerun the analysis with those observations excluded.

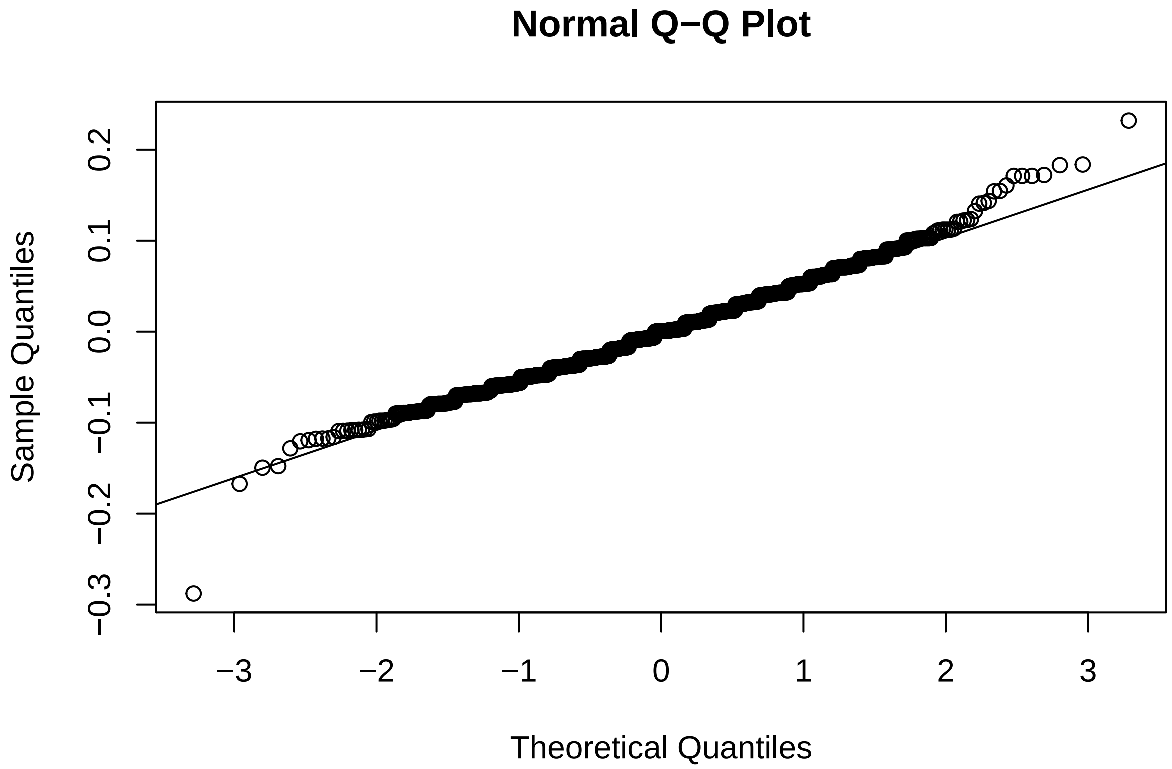
```
egg.lm2<-lm(H.Mass~EggCode+Sex+SVL, data=eggtemp, na.action=na.omit, subset=c(-865, -885, -895))
summary (egg.lm2)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode + Sex + SVL, data = eggtemp, subset = c(-865,
##     -885, -895), na.action = na.omit)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.287825 -0.038075  0.000696  0.033192  0.231972
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.663412   0.038549 -17.210   <2e-16 ***
## EggCodeM    -0.039241   0.004038  -9.718   <2e-16 ***
## SexM        -0.001987   0.003545  -0.561    0.575
## SVL          0.050509   0.001547  32.639   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05521 on 979 degrees of freedom
## Multiple R-squared:  0.6379, Adjusted R-squared:  0.6368
## F-statistic: 574.8 on 3 and 979 DF,  p-value: < 2.2e-16
```

Notice that the exclusion of these points did not change the results very much. We already knew this from the Cook's distance. A Cook's distance less than one usually indicates that no individual points have a large effect on the parameters.

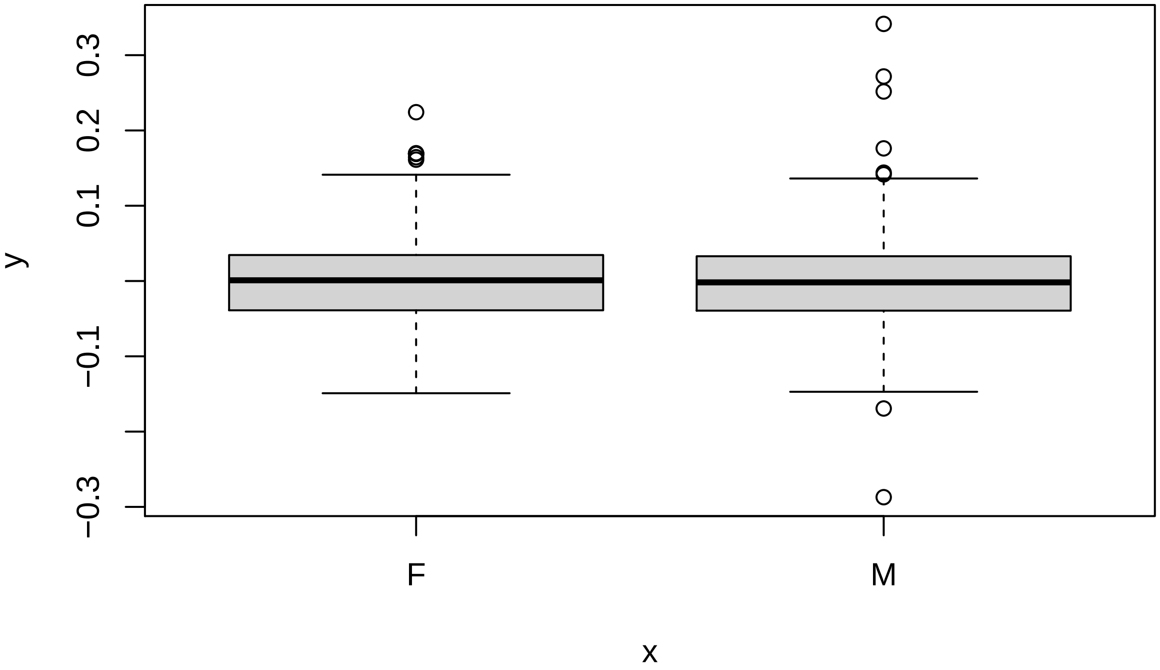
We can look at the residuals of this new model using the qqplot. Note that qqline adds the straight line to the plot

```
par(mfrow=c(1,1))
qqnorm(resid(egg.lm2))
qqline(resid(egg.lm2))
```



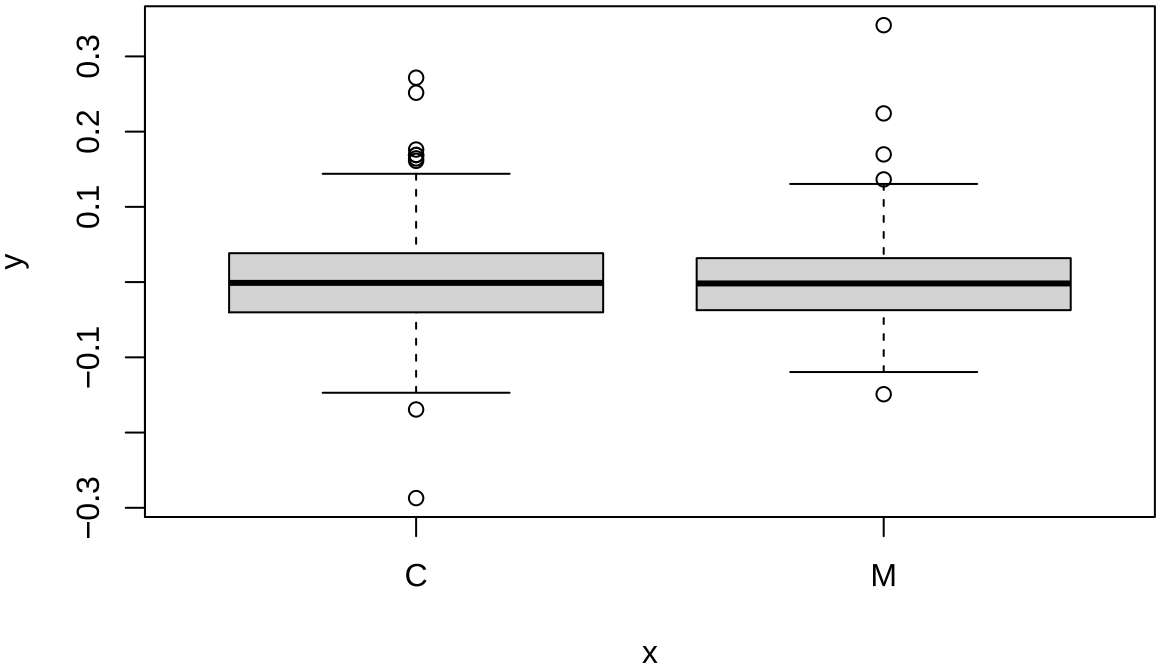
In order to test for heteroscedasticity between males and females we could plot

```r
plot(eggtemp$Sex, resid(egg.lm))
```



They look pretty even. What about between control and miniaturized eggs.

```r
plot(eggtemp$EggCode, resid(egg.lm))
```



```
Levene (resid(egg.lm), eggtemp$Sex)
```

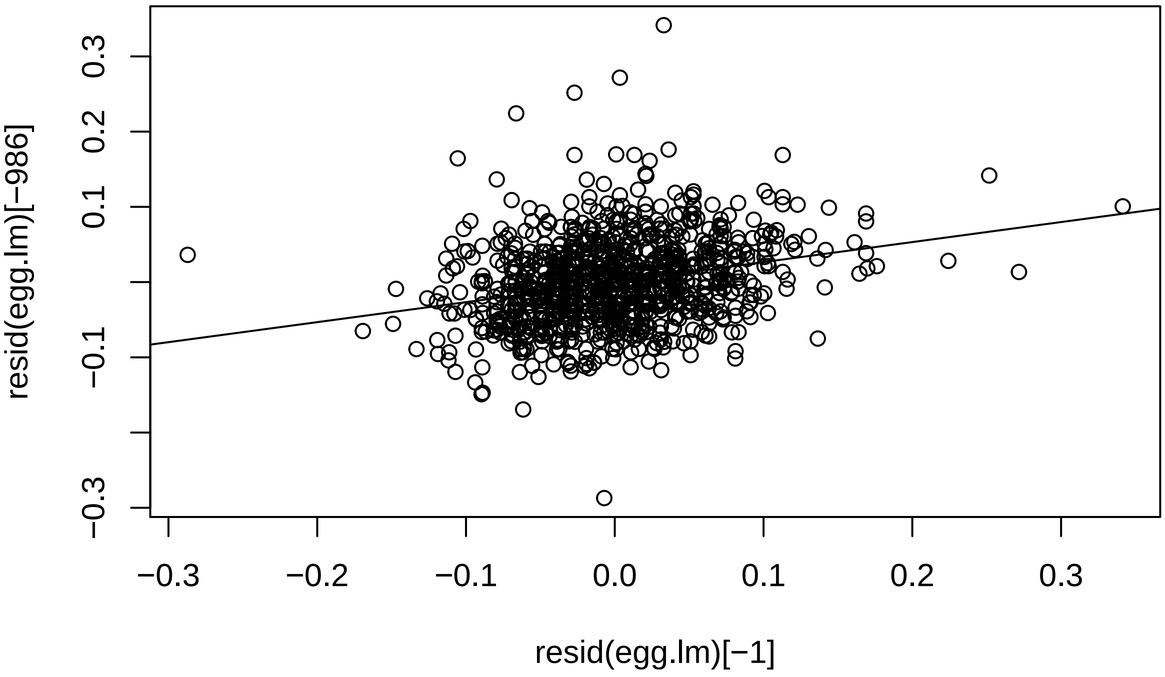
```
## Analysis of Variance Table
##
## Response: resp
##            Df  Sum Sq   Mean Sq F value Pr(>F)
## group       1 0.00044 0.0004370  0.3279  0.567
## Residuals 984 1.31132 0.0013326
```

We can also look for autocorrelation in the residuals by…

```
length(resid(egg.lm))
```

```
## [1] 986
```

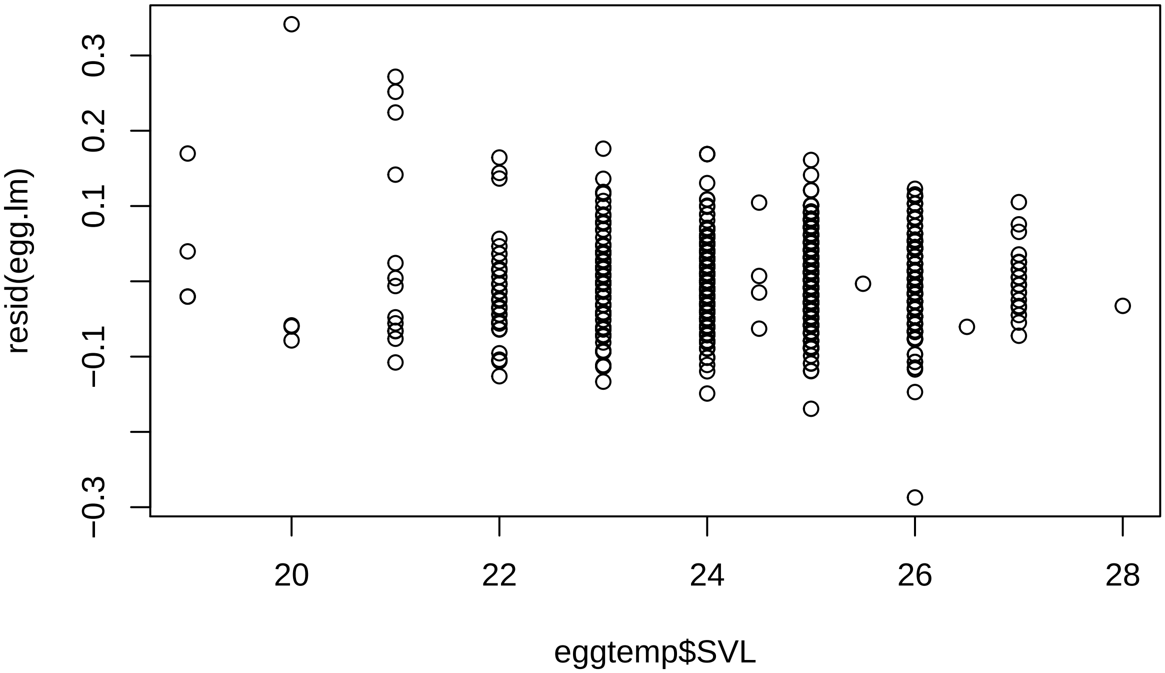
```
plot(resid(egg.lm)[-1], resid(egg.lm)[-986])
abline(lm(resid(egg.lm)[-986]~resid(egg.lm)[-1]))
```



So there does seem to be some autocorrelation in the residuals. We can test this statistically by performing an analysis of the residuals.

We should also check to see that the residuals are independent of the other covariates.

```
plot(eggtemp$SVL, resid(egg.lm))
```



6 Interactions in R

Up to this point we have considered only additive effects. That is, the effects of each predictor variable acts additively on the response variable. Linear models, however, can also accommodate variables that affect the response in a non-additive way. These are called interactions. A statistical interaction occurs when *the effect of a covariate is NOT CONSTANT, but the magnitude of the effect depends on the level of some other covariate*. So for example, a statistical interaction between A and B on some response variable y would mean that the effect of A on y is not constant, but it depends on the value of B. You will find this general definition of an interaction to be extremely important - so repeat it over and over in your head. In the future, you might find yourself in a jam to explain an interaction between two of your predictors. When stuck just fall back on this general definition.

Statistical interactions can occur between two categorical covariates (I use the word 'covariate' to refer generally to predictors regardless of whether they are continuous or categorical), two continuous covariates or any combination of the two. There can of course also be higher level interactions (3-way or even 4-way interactions). Two-way interactions (interactions between two main effects) can be hard enough to interpret. 3-way interactions (or 4-way!) can be extremely hard to interpret. Note, however, that we can always fall back on our basic interpretation of an interaction. For example, if we have a 3-way interaction between A, B and C on some response variable y, then this means (by definition) that the two-way interaction between A and B on y is not constant, but depends on the value of C. See? You have it!

In R we denote interactions between two covariates using a colon placed between them. So a statistical interaction between A and B on y would be coded as $y = A + B + A : B$. It doesn't matter whether A and B are continuous or categorical, the R notation is the same. Note that you should always include the component main effects in model for which a higher order interaction is included. There are times when you might want to intentionally exclude a main effect but keep its interaction but these can make the model uninterpretable, so as a general rule you should always include the main effects for terms that are included in an interaction. Similarly if you include a 3-way interaction between A:B:C you should also include all combinations of pairwise 2-way interactions (A:B, A:C, B:C). In R, you can use a simpler notation that will include all interactions and component main effects by placing an asterisk between the predictors. So $y = A * B$ is a short-hand for using $y = A + B + A : B$.

6.1 Interaction Example: lizard body mass

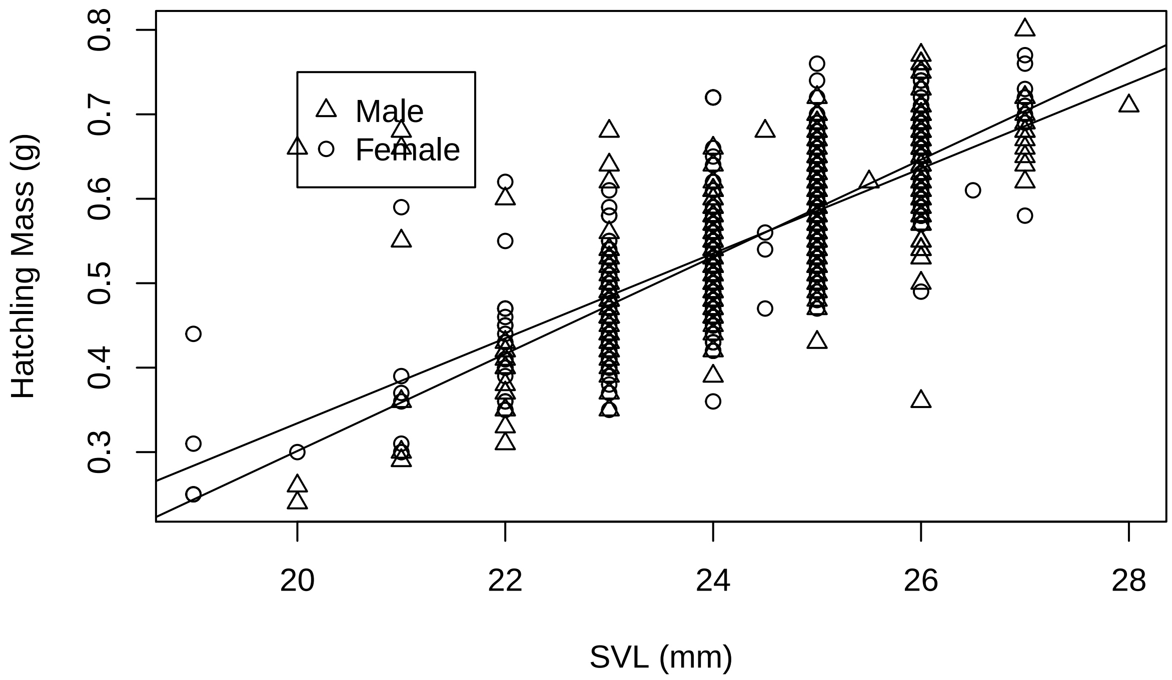
Recall that we previously found that there was not an effect of SEX on the hatchling mass of lizards. That is male were not consistently heavier than females. But it is possible that the effect of SEX on the mass of the hatchling lizards depends on their body size. That is, perhaps there is an inTeraction between SEX and body size on lizard hatchling mass.

```
egg.lm3<-lm(H.Mass~EggCode+Sex+SVL+Sex:SVL, data=eggtemp, na.action=na.omit)
summary(egg.lm3)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode + Sex + SVL + Sex:SVL, data = eggtemp,
##     na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28226 -0.03959 -0.00085  0.03572  0.32447
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.670480   0.051105 -13.120   <2e-16 ***
## EggCodeM    -0.041844   0.004180 -10.012   <2e-16 ***
## SexM         0.164913   0.071269   2.314   0.0209 *
## SVL          0.050836   0.002066  24.601   <2e-16 ***
## SexM:SVL    -0.006689   0.002896  -2.310   0.0211 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05739 on 981 degrees of freedom
## Multiple R-squared:  0.6094, Adjusted R-squared:  0.6078
## F-statistic: 382.6 on 4 and 981 DF,  p-value: < 2.2e-16
```

Note that there does in fact seem to be an interaction between SEX and SVL on hatchling mass. We should plot these resuls so that we can see what is going on graphically

```
 plot(eggtemp$SVL, eggtemp$H.Mass, xlab="SVL (mm)", ylab="Hatchling Mass (g)",
     pch=as.numeric(eggtemp$Sex))

abline(lm(subset(eggtemp$H.Mass, eggtemp$Sex=="F")~subset(eggtemp$SVL,
                                                          eggtemp$Sex=="F")))
abline(lm(subset(eggtemp$H.Mass, eggtemp$Sex=="M")~subset(eggtemp$SVL,
                                                          eggtemp$Sex=="M")))
legend(20, 0.75, c("Male", "Female"), pch=as.numeric(eggtemp$Sex))
```



This R code for this plot is pretty complicated but see if you can break it down.

Note that in this case we have included the interaction, so it does not make sense to interpret the significance (effect) of either of the *Main effects* (SVL or Sex), because the significant interaction means that the effect of SVL depends on whether it is a male or a female OR that the effect of being a male depends on your size (SVL). Note that you can always interpret an interaction from either direction!

If the interaction was not significant then we could remove the interaction term from the model and determine the effects of each of Sex and SVL independently.

Please remember that this is a linear model. So we can write out the equation for the model and plug in the parameters to predict what the hatchling mass on a particular lizard is expected to be.

So our model is

$$HatchMass = -0.67 - 0.042(isminiaturized?) + 0.16(isMale?) + 0.05 * SVL - 0.007(ismale?) * (SVL)$$

So for individual 440

```
eggtemp[440, ]
```

```
##     EggCode PostEMass Sex H.Mass SVL
## 440       C      0.41   M    0.6  25
```

Predicted hatch mass is: $-0.67 - 0.042 * (0) + 0.16 * (1) + 0.05 * (25) - 0.007(1) * (25)$

```
-0.67048-(0.042*0)+(0.164913*1)+(0.050836*25)-(0.006689*1*25)
```

```
## [1] 0.598108
```

OR

```
predict(egg.lm3)[440]
```

```
##       440
## 0.5981122
```

NOTE an important interpretation of an interaction here. Note that in the above linear model we can interpret the interaction parameter SexM:SVL to be NOT the effect of being male on hatchling mass, but INSTEAD the effect of being male on the relationship between SVL and hatchling mass. In this case, SexM:SVL is the effect of being Male on the slope of the relationship between SVL and H.Mass. So we could then calculate the parameter for the effect of SVL on H.Mass for males as 0.050836-0.006689. Since females are the reference level here for Sex the slope of the relationship between SVL and H.Mass for females is simply 0.050836.

Recall that we can use a short-hand notation in R when we are interested in including all the main effects as well as all higher order interaction

```
egg.lm4<-lm(H.Mass~EggCode+Sex*SVL, data=eggtemp, na.action=na.omit)
summary (egg.lm4)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode + Sex * SVL, data = eggtemp, na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28226 -0.03959 -0.00085  0.03572  0.32447
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.670480   0.051105 -13.120   <2e-16 ***
## EggCodeM    -0.041844   0.004180 -10.012   <2e-16 ***
## SexM         0.164913   0.071269   2.314   0.0209 *
## SVL          0.050836   0.002066  24.601   <2e-16 ***
## SexM:SVL    -0.006689   0.002896  -2.310   0.0211 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05739 on 981 degrees of freedom
## Multiple R-squared:  0.6094, Adjusted R-squared:  0.6078
## F-statistic: 382.6 on 4 and 981 DF,  p-value: < 2.2e-16
```

Or we could get even fancier

```
egg.lm5<-lm(H.Mass~EggCode*Sex*SVL, data=eggtemp, na.action=na.omit)
summary (egg.lm5)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode * Sex * SVL, data = eggtemp, na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27910 -0.03910 -0.00011  0.03333  0.34699
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.813855   0.072577 -11.214  < 2e-16 ***
## EggCodeM           0.237195   0.105082   2.257 0.024213 *
## SexM               0.419679   0.104211   4.027 6.08e-05 ***
## SVL                0.056559   0.002915  19.401  < 2e-16 ***
## EggCodeM:SexM     -0.514132   0.155467  -3.307 0.000977 ***
## EggCodeM:SVL      -0.011363   0.004328  -2.626 0.008783 **
## SexM:SVL          -0.016817   0.004173  -4.030 6.00e-05 ***
## EggCodeM:SexM:SVL  0.020828   0.006376   3.267 0.001126 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05714 on 978 degrees of freedom
## Multiple R-squared:  0.614,  Adjusted R-squared:  0.6112
## F-statistic: 222.2 on 7 and 978 DF,  p-value: < 2.2e-16
```

Wow now this is pretty crazy! Believe it or not this is not that biologically unique for side-blotched lizards who often show highly significant complicated interactions - this is one of the challenges and fun of studying them! It would take some thought to make some biological sense of these patterns though. It also helps that we have data on nearly 1000 individuals, so we have lots of power to detect complicated interactions. In cases like this you always interpret the highest order interaction. The rest are more difficult to make sense of on their own.

REMEMBER: as a general rule - always include all of the main effects as main effects that appear as part of an interaction. For example, DO NOT include a model such as…

```
egg.lm6<-lm(H.Mass~EggCode+SVL+Sex:SVL, data=eggtemp, na.action=na.omit)
```

This model includes the interaction *Sex:SVL* but not the main effect *Sex*. AS a result it is very difficult to interpret what the parameters and the significance of the terms actually mean.

7 Sums of Squares

I have previously emphasized that you need to know how R codes its contrasts in order to be able to interpret the parameters. It is also important to be aware that there are several differEnt ways to calculate Sums of Squares (SS). That is, in an ANOVA framework you can assess the significance of a term in the model in a number of different ways.

- Type I SS - Significance of each term in a model is assessed sequentially. The effect of each term is assessed against y adjusted for the effects of all prior terms in the model. As a result, the order in which you enter your predictors in the model matters!! That is $y = A + B$ is not the same as $y = B + A$.

In the case of an interaction $y = A + B + C + A : B$ the significance of C is assessed after adjusting for the effects of A and B, but before the effects of A:B.

- Type II SS - The significance of each term is assessed against all other terms except its higher order relatives. In this case the order of terms in the model DOES NOT matter. In the case of $y = A + B + C + A : B + A : C$ the significance of B is assessed after adjusting for A and C and A:C but not A:B.
- Type III SS - The signifiance of a term in the model is assessed after adjustimng for all other terms in the model. Order of terms in the model DOES NOT matter.

I have mentioned before that the *summary* command provides measures of the significance of the parameter given all the other terms in the model. That is, the order in which the terms appear in the model doesn't matter. However, when you assess the significance of a TERM using the *anova* command then there are 3 different ways that the significance of that term can be assessed. IN SOME OF THESE THE ORDER IN WHICH THE TERMS ARE ENETERED INTO THE MODEL MATTERS!! This is because there are several ways in which SS can be calculated. The termInology of Type-III SS, or Type-I SS comes from SAS originally, but this terminology has now become widespread so other statistical packages now make use of this common terminology.

```
summary (egg.lm3)
```

```
##
## Call:
## lm(formula = H.Mass ~ EggCode + Sex + SVL + Sex:SVL, data = eggtemp,
##     na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28226 -0.03959 -0.00085  0.03572  0.32447
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.670480   0.051105 -13.120   <2e-16 ***
## EggCodeM    -0.041844   0.004180 -10.012   <2e-16 ***
## SexM         0.164913   0.071269   2.314   0.0209 *
## SVL          0.050836   0.002066  24.601   <2e-16 ***
## SexM:SVL    -0.006689   0.002896  -2.310   0.0211 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05739 on 981 degrees of freedom
## Multiple R-squared:  0.6094, Adjusted R-squared:  0.6078
## F-statistic: 382.6 on 4 and 981 DF,  p-value: < 2.2e-16
```

Remember that the significance of the parameters is based on all other terms in the model so this is like a Type-III SS.

However the default for the anova command is Type-I SS

```
anova (egg.lm3)
```

```
## Analysis of Variance Table
##
## Response: H.Mass
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## EggCode      1 1.9815 1.98153 601.6100 < 2.2e-16 ***
## Sex          1 0.0270 0.02695   8.1826  0.004319 **
## SVL         1 3.0144 3.01437 915.1899 < 2.2e-16 ***
## Sex:SVL     1 0.0176 0.01757   5.3353  0.021104 *
## Residuals 981 3.2311 0.00329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So in this case the significance of *EggCode* is assessed first. Then the significance of *Sex* is assessed relative to what *EggCode* was unable to explain. This is like we did a regression (or t-test) between *EggCode* and *H.Mass* and then tried to do another t-test on the residuals of this first test using *Sex* as the predictor now. So in this case the order in which terms are entered in the model matters. So, for example the significance of *Sex* will probably depend on whether the model was H.Mass~EggCode+Sex or H.Mass~Sex+EggCode.

If we want to get anova results based on Type-II or III SS we need to use the Anova command rather than the anova command - remember that R is case-sensitive! This needs to be loaded from the car package. You will probably need to download this package first.

```
#install.packages ("car", dependencies=T, repos="http://cran.us.r-project.org")
library (car)
```

```
## Loading required package: carData
```

```
Anova (egg.lm3, type="II")
```

```
## Anova Table (Type II tests)
##
## Response: H.Mass
##           Sum Sq  Df  F value Pr(>F)
## EggCode   0.3301   1 100.2331 <2e-16 ***
## Sex       0.0001   1   0.0195 0.8891
## SVL       3.0144   1 915.1899 <2e-16 ***
## Sex:SVL   0.0176   1   5.3353 0.0211 *
## Residuals 3.2311 981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that here I first installed the car package in case it wasn't already installed on my machine. I have added a pound sign before it here so that it doesn't keep installing each time I create a PDF of this document (I have done this now hundreds of times!). The repos subcommand simply specifies the mirror site that I want to use to get this package. Note that usually I indicate this in the preferences in R or in RStudio, but I need to specify it directly here because I am using knitr to create this document interactively. I then loaded the package before using the Anova command. Note that this is a different command from the anova command that we used previously!! Anova in the car package allows us to specify SS-type.

So remember that for Type-II the effects of all main effects are considered simultaneously before the interaction effects. So notice that the interaction SS (or P) hasn't changed. Note that the significance of EggCode has changed quite a bit. Note also that this has no effect on the parameters. Just the significance based on an F-test.

Or we can get the Type-III SS using…

```
Anova (egg.lm3, type="III")
```

```
## Anova Table (Type III tests)
##
## Response: H.Mass
##             Sum Sq  Df  F value  Pr(>F)
## (Intercept) 0.5669   1 172.1275 < 2e-16 ***
## EggCode     0.3301   1 100.2331 < 2e-16 ***
## Sex         0.0176   1   5.3544 0.02088 *
## SVL         1.9935   1 605.2299 < 2e-16 ***
## Sex:SVL     0.0176   1   5.3353 0.02110 *
## Residuals   3.2311 981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now the significance of the terms is based on the presence of all other terms in the model. Note that again the interaction SS hasn't changed since it was already being assessed based on all other terms in the model. I know that this is a bit disturbing if you haven't thought about these issues before. You shouldn't be too disturbed though if you think about what it is you are testing with your model. Each type of SS is associated with a slightly different hypothesis. Sorry, but you are going to need to think!!

My final tip is to remember that SS Type only matters when you have imbalance in your data. In an ANOVA type scenario this means unequal sample sizes in each group. In a regression type scenario this means some form of collinearity (correlations among predictor variables).

Remember that the estimation and significance of the parameters in the summary command is ALWAYS based on the presence of all other terms in the model. So when you are interpreting the significance of your parameters it NEVER matters which order they are entered into the model.

8 Assessing Non-Linearities

So far we have discussed linear combinations of terms predicting some response variable. I also indicated previously that general linear models require that predictor variables be linearly related to the response variable. But relationships in nature are often non-linear. How can we deal with these using linear models? Luckily many well-known non-linear relationships can be linearized using a transformation. Alternatively we can explore unpredictable nonlinear relationships using non-parametric approaches such as localized regressions called *splines* or we can try and force non-linear relationships into a quadratic form.

Remember that we have been working with a model of the size of hatchling lizards. In this example we will be looking at the size of hatchling lizards in relation to the size of the egg that they came from.

```r
egg.lm4<-lm(H.Mass~PostEMass, data=eggtemp)
summary (egg.lm4)
```

```
##
## Call:
## lm(formula = H.Mass ~ PostEMass, data = eggtemp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48222 -0.02721  0.00489  0.03642  0.29272
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18181    0.01097   16.58   <2e-16 ***
## PostEMass    0.92353    0.02614   35.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06087 on 984 degrees of freedom
## Multiple R-squared:  0.5593, Adjusted R-squared:  0.5588
## F-statistic:  1249 on 1 and 984 DF,  p-value: < 2.2e-16
```

So clearly there is an effect of egg mass on hatchling mass, but is it actually linear??

To test this we need to load a new package into R. This package is called 'gam'. The gam function lets us fit a localized regression to the effect of *PostEMass* on *H.Mass* using something called a spline. To denote this I will simply put the predictor within parentheses and precede it with the letter s.

```
#install.packages ("mgcv", dependencies=T, repos="http://cran.us.r-project.org")
library (mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-32. For overview type 'help("mgcv-package")'.
```

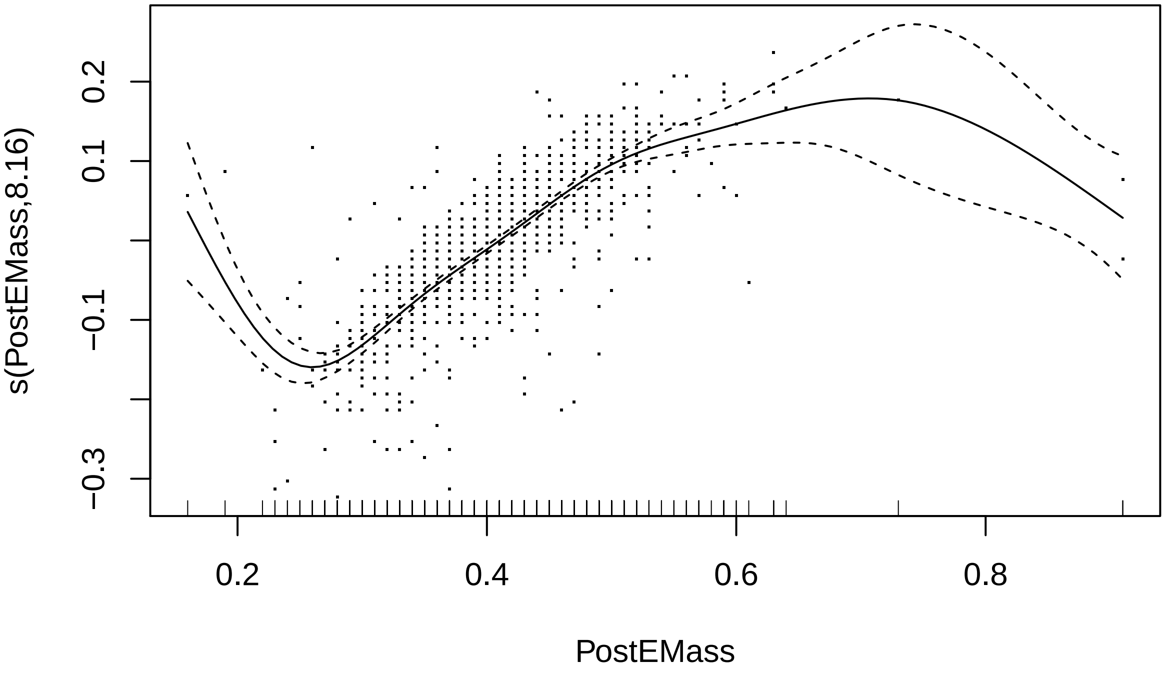
```
egg.gam<-gam(H.Mass~s(PostEMass), data=eggtemp)
anova(egg.gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## H.Mass ~ s(PostEMass)
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(PostEMass) 8.158  8.785 197.5  <2e-16
```

Note that the significance in this output refers to the significance of the "nonlinearity" and NOT the significance of the term as a whole. So it looks like there are significant nonlinearities in the relationship between hatchling body mass and egg mass.

We can explore this graphically using

```
#plot.gam(egg.gam, ask=T)
plot.gam(egg.gam, resid=T, se=T)
```



I have included two lines of code here but the first line is hidden by #. In the first line the ask=T subcommand generates the interactive menu that lets you specify some of the figures attributes and specify which relationship in a complicated model you want to plot. In the second line of code I asked for the plot directly and directly asked for residuals and SEs to be plotted. The default nonlinear plot shows the results of a localized regression called a cubic spline. This is VERY different from the 'spline' fit that programs like Excel use. In this case the 'wiggliness' of the line is not determined by you but is determined objectively using an algorithm called a generalized cross-validation score. This weights the amount of variance explained against the number of parameters needed to fit the model (polynomial > quadratic > linear) based on principles of parsimony.

The plot shows that there is clearly a nonlinear relationship. The first thing to note is that a spline is a localized regression, so the line will take on whatever slope is best desribed by the points within a small window of x values. For this reason you need to take the line with a 'grain of salt' in places where data are sparse (i.e. at the extremes of the x values). So in this case the big upswing at very low values of *PostEMass* and the strong downturn at very high values of *PostEMass* are probably not real (i.e. are probably driven by a small number of points). Nevertheless, it does seem as though the positive relationship between *PostEMass* and *H.Mass* declines as *PostEMass* increases. Lizards that come from very big eggs actually don't end up being as large as we might have expected from a linear relationship. What is nice though is that it looks like a quadratic fit might be a good first guess at what the non-linearities are.

We can specify a quadratic relationship in R using the $I(X^2)$ notation.

```
egg.gam2<-lm(H.Mass~PostEMass+I(PostEMass^2), data=eggtemp, na.action=na.omit)
summary (egg.gam2)
```

```
##
## Call:
## lm(formula = H.Mass ~ PostEMass + I(PostEMass^2), data = eggtemp,
##     na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27573 -0.02790  0.00321  0.03321  0.39076
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.08053    0.02858  -2.818  0.00493 **
## PostEMass       2.16273    0.12808  16.885  < 2e-16 ***
## I(PostEMass^2) -1.41672    0.14363  -9.864  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05809 on 983 degrees of freedom
## Multiple R-squared:  0.599,  Adjusted R-squared:  0.5981
## F-statistic: 734.1 on 2 and 983 DF,  p-value: < 2.2e-16
```

Here we have included both a linear and a quadratic term for *PostEMass*. We can interpret the significance of the nonlinearity by the significance of the quadratic $I(X^2)$ term. So there does seem to be a significant quadratic relationship. Note that the negative parameter for the squared term indicates a downward facing quadratic. We can also check to see if this has accounted for the nonlinearities.

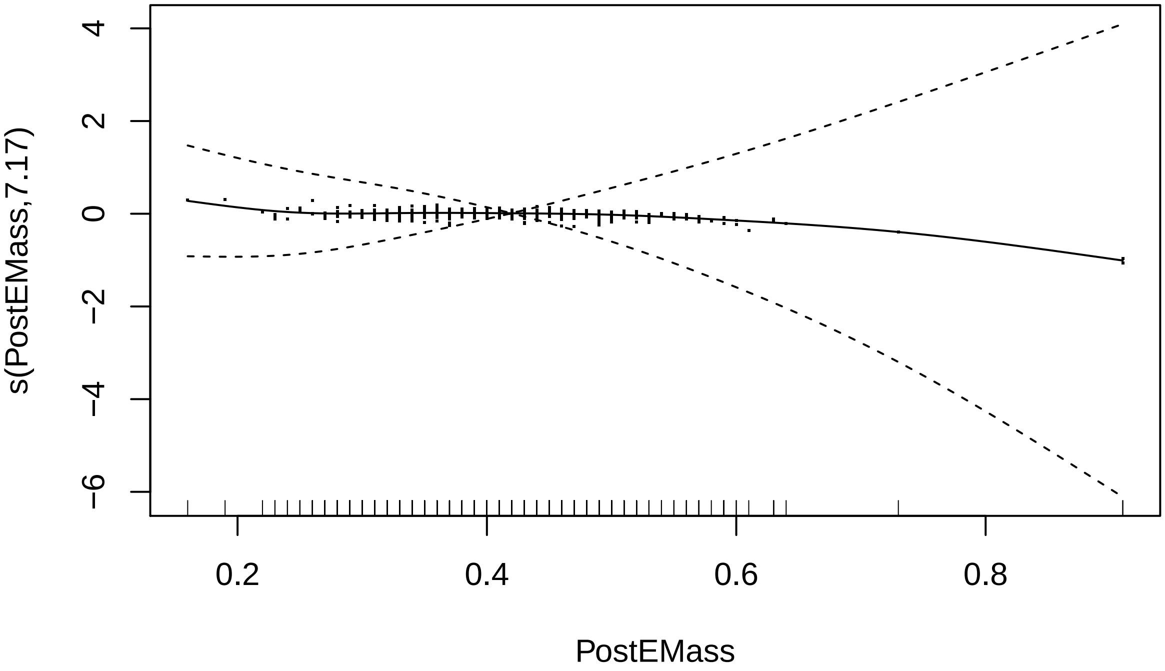
```
egg.gam3<-gam(H.Mass~s(PostEMass)+I(PostEMass^2), data=eggtemp, na.action=na.omit)
anova(egg.gam3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## H.Mass ~ s(PostEMass) + I(PostEMass^2)
##
```

```
## Parametric Terms:
##                 df     F p-value
## I(PostEMass^2)  1 0.166   0.684
##
## Approximate significance of smooth terms:
##                 edf Ref.df     F p-value
## s(PostEMass) 7.170  7.797 53.64  <2e-16
```

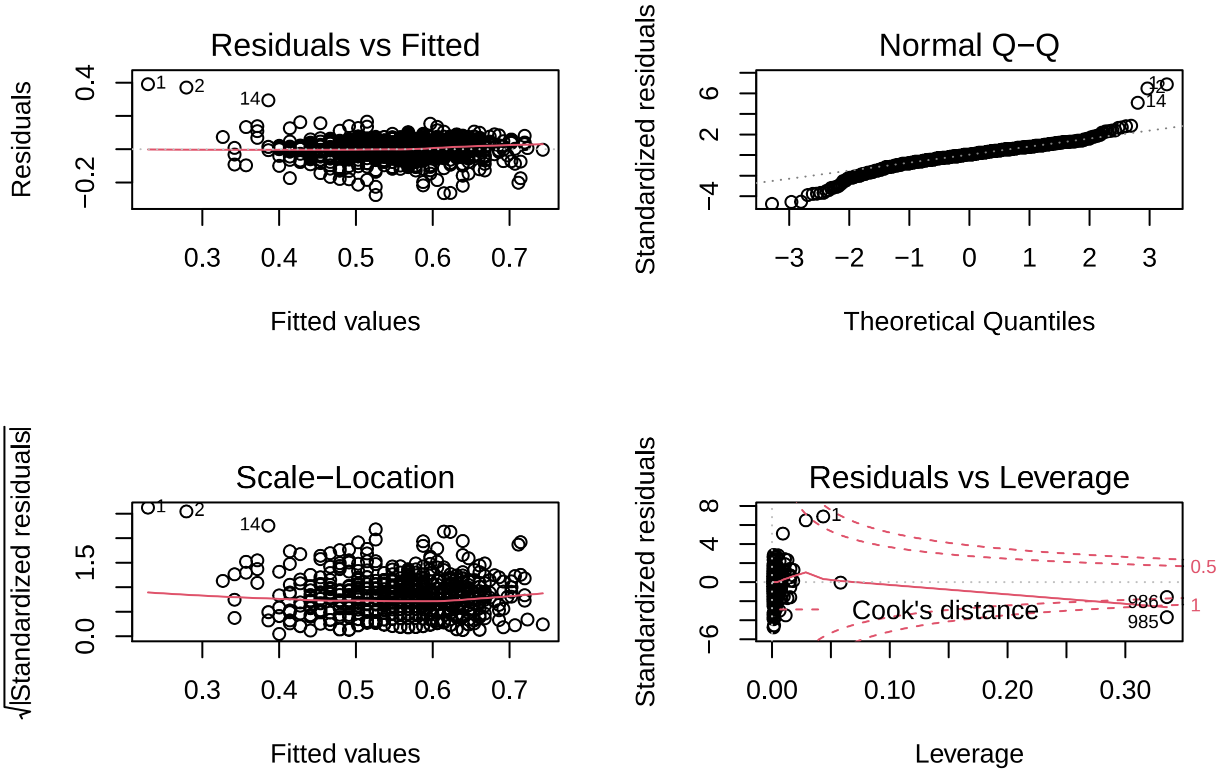
There still seem to be some nonliearities that weren't accounted for by our model. But remember that we have an awful lot of power here so we should be sure to check the plot.

```r
plot.gam(egg.gam3, resid=T, se=T)
```



This relationship sure seems flat so we will leave this for now. Note that it is always a good idea to plot these nonlinear relationships with the standard errors and residuals turned on. This forces the scale of the y axis to accommodate the range of residuals and SEs. Otherwise the y axis will be rescaled and the nonlinearities will appear much much larger!

```r
par(mfrow=c(2,2))
plot(egg.gam2)
```



Note that there are some large outliers here as well as some points that are having a very strong effect on our relationship. We should try and sort these out, but we will leave them for now.

9 Regression when there is Error in X

When we are performing general linear models with continuous covariates we are essentially performing a regression. More specifcally we are using a technique called *Ordinary Least Squares Regression*, or OLS Regression. One assumption of OLS regression is that all error in the relationship is due to error in your response variable and not error in a predictor. When predictors are categorical we hope that they are measured without error(!), but in some cases predictors might be measured with some error or even as much error as the response variable! For example if we are predicting the abundance of Y by the abundance of X then both X and Y might be measured with the same amount of error. NOTE that this is only an issue for our estimation of the ACTUAL slope of a relationship is (i.e. not just whether it is significantly different from zero) when we have a predictor variable that is measured with error.

An alternative to OLS regression is called Model II regression in which the slope of the relationship is estimated using other assumptions about how much error is present in our X and Y variables. This part of the script will cover Model II regression in R. I will use a part of the data file from ConeHandling.csv.

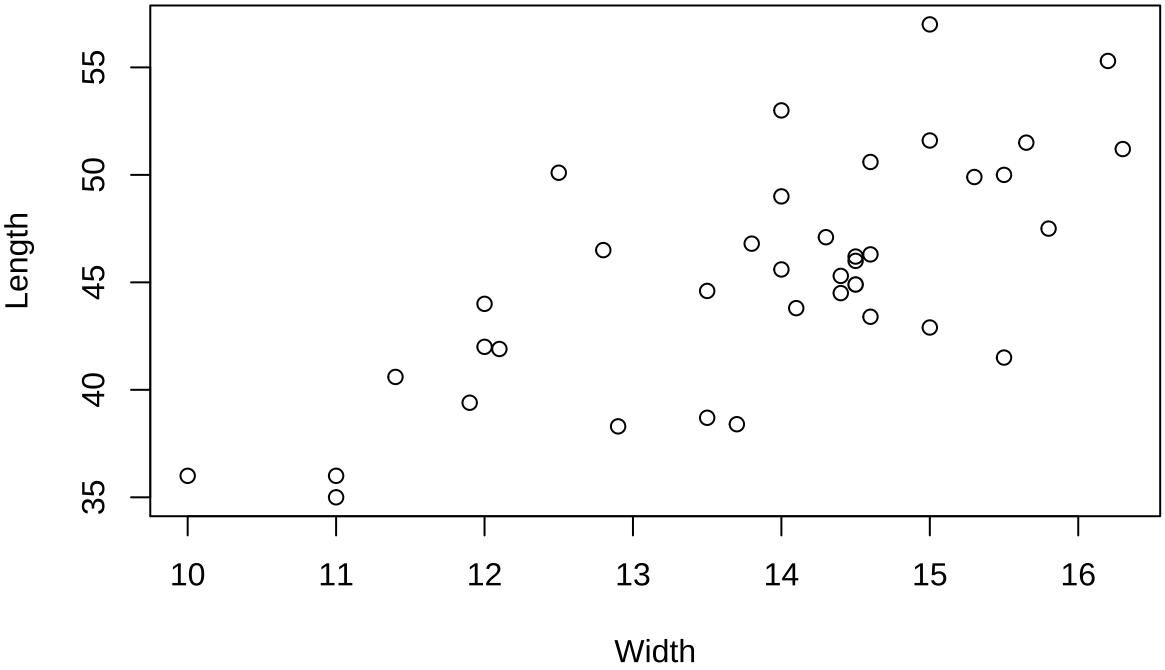
```
cone.handling<-read.table("data/ConeHandling.csv", sep=",", header=T)
summary(cone.handling)
```

```
##    Squirrel             Trial           Length           Width      
##  Length:39          Min.   :1.000   Min.   :35.00   Min.   :10.00  
##  Class :character   1st Qu.:2.000   1st Qu.:41.95   1st Qu.:12.85  
##  Mode  :character   Median :3.000   Median :45.30   Median :14.30  
##                     Mean   :2.769   Mean   :45.32   Mean   :13.86  
##                     3rd Qu.:4.000   3rd Qu.:49.45   3rd Qu.:14.80  
##                     Max.   :6.000   Max.   :57.00   Max.   :16.30  
##     Handed            Chirality             Time          Sex
##  Length:39          Length:39          Min.   : 61.0   Length:39
##  Class :character   Class :character   1st Qu.: 82.5   Class :character
##  Mode  :character   Mode  :character   Median :105.0   Mode  :character
##                                        Mean   :115.3
##                                        3rd Qu.:146.0
##                                        Max.   :209.0
```

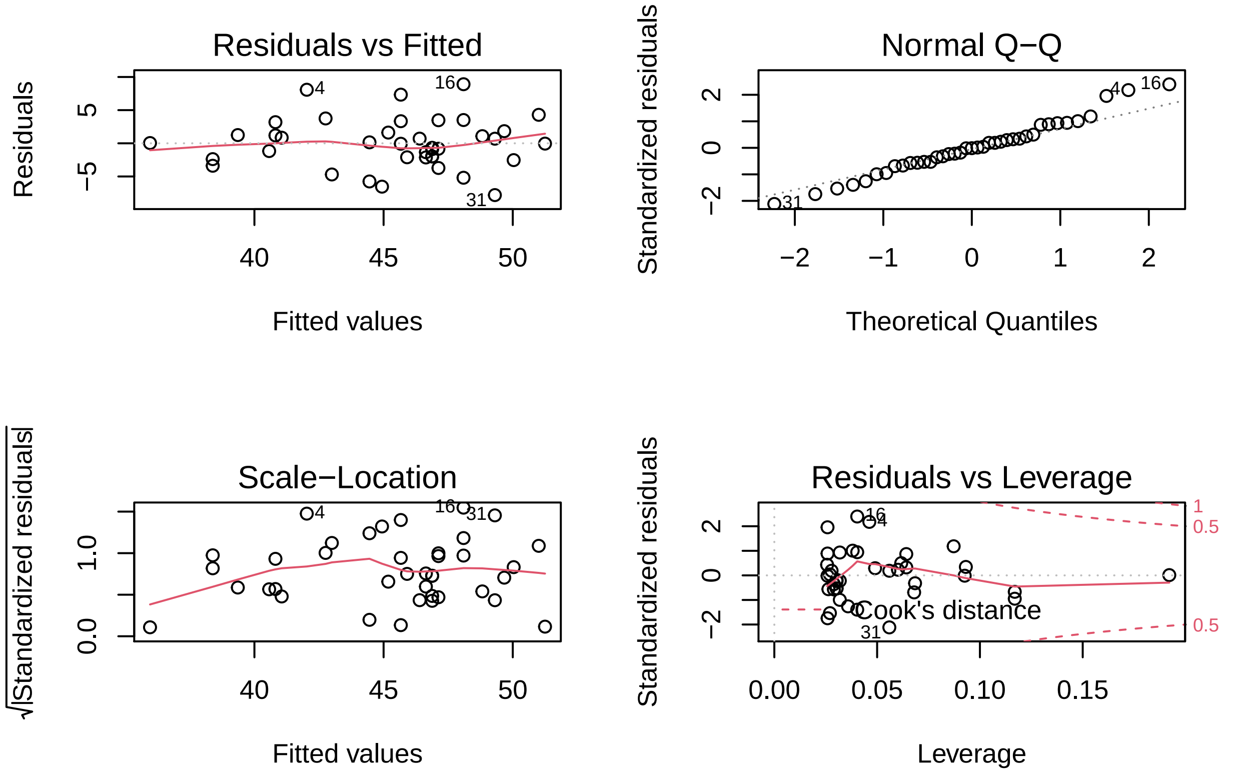
These are data that I collected with some field assistants in the Yukon on spruce cone morphology and how quickly my test subjects (field assistants!) could remove seems from the cones. We will come back to this dataset later, but for now we are only interested in the relationship between the Length of the spruce cones (mm) and their width (mm)

We will start to model the relationship between these two measures of cone size using Length as the response

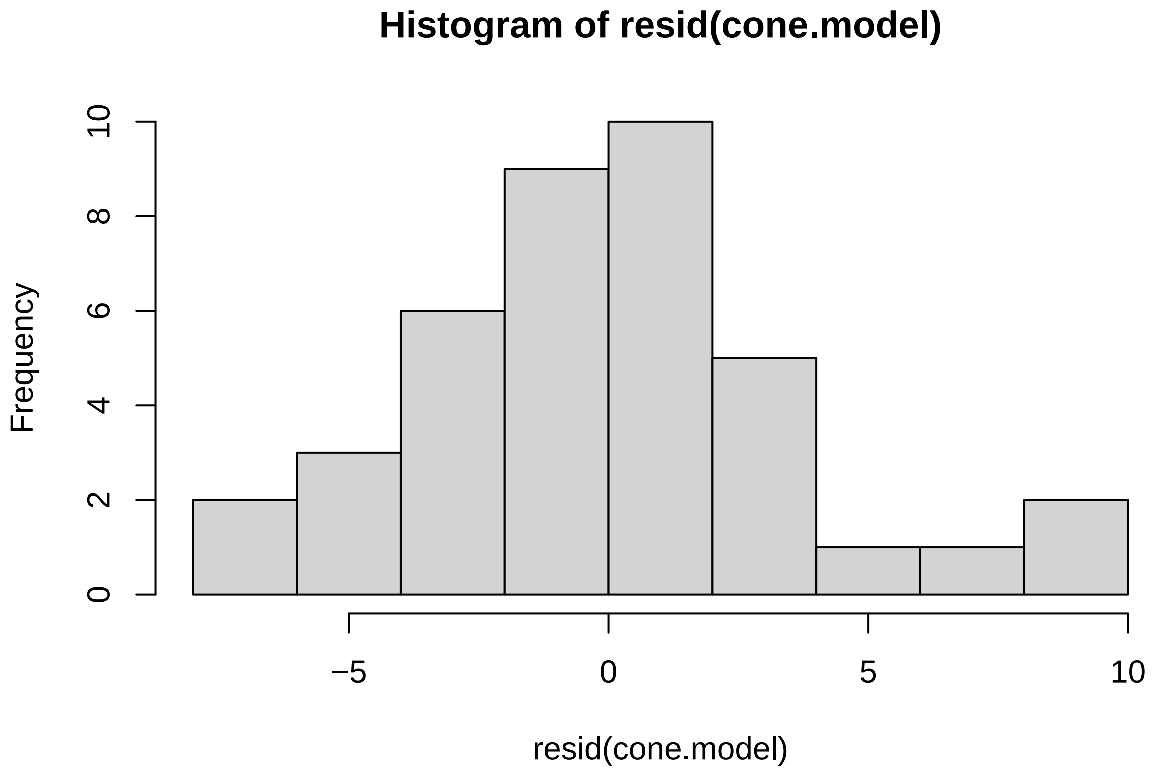
```
  attach (cone.handling)
par(mfrow=c(1,1))
  plot(Width, Length)
```



```
cone.model<-lm(Length~Width)
par(mfrow=c(2,2))
plot(cone.model)
```



```r
par(mfrow=c(1,1))
hist(resid(cone.model))
```



Diagnostics in general look pretty good

```
summary (cone.model)
```

```
##
## Call:
## lm(formula = Length ~ Width)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8069 -2.1236 -0.0482  1.7237  8.9064
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.6938     5.6009   2.088   0.0438 *
## Width         2.4267     0.4019   6.039 5.57e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.797 on 37 degrees of freedom
## Multiple R-squared:  0.4964, Adjusted R-squared:  0.4827
## F-statistic: 36.46 on 1 and 37 DF,  p-value: 5.568e-07
```

So the slope of the relationship between width and length is significantly different from zero

But what is the actual relationship? The slope above (2.43) is based on Ordinary Least Squares (OLS) regression. One of the assumptions of OLS regression is that predictor variables are measured without error and that all residual error is due to the response variable (i.e. residuals) and not errors in the predictor. In this case, we know that there is as much error in width measurements as there is in length measurements on cones. This error will tend to downwardly bias the estimated slope and the magnitude of this bias will be determined by the strength of the relationship. When the OLS R^2 is high then there is little downward bias. When the R^2 is low there could be substantial bias. Note that in this case the R^2 is reasonable but it is not so high as to prevent bias in the slope due to error in X.

Recall that the equation for the model above is:

$$Length = 11.69 + 2.43Width$$

We can show that this slope is biased by plotting the alternative model of $x = y$ and rearranging the linear equation

```
cone.model2<-lm(Width~Length)
summary (cone.model2)
```

```
##
## Call:
## lm(formula = Width ~ Length)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3338 -0.8475  0.3026  0.6992  2.4253
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.58622    1.54507   2.968  0.00523 **
## Length       0.20454    0.03387   6.039 5.57e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.102 on 37 degrees of freedom
## Multiple R-squared:  0.4964, Adjusted R-squared:  0.4827
## F-statistic: 36.46 on 1 and 37 DF,  p-value: 5.568e-07
```

Again the relationship is significant. The slope is different from zero, but now the equation is:

$$Width = 4.59 + 0.205Length$$

This can be rearranged as:

$$(Width - 4.59)/0.205 = Length$$

OR

$$Length = -22.39 + 4.88Width$$

This is quite different from the equation above:

$$Length = 11.69 + 2.43Width$$

We can calculate the Reduced Major Axis (RMA) slope as the geometric mean of y modeled by x and x modeled by y. This will give us the best slope estimate assuming equal error in our measurement of cone length and cone width.

```
sqrt(4.88*2.43)
```

```
## [1] 3.443603
```

I found a function written for the estimation of RMA slopes and intercepts with CI's, but I have not checked it so beware.

```
RMA<-function (x,y)
{
    OK <- complete.cases(x, y)
    x <- x[OK]
    y <- y[OK]


    slope <- sign(cor(x, y))*sqrt(var(y)/var(x))
    df <- (n1 <- (n <- length(x))-1)-1
    SEintercept <- sqrt((MSE <- (var(y)-cov(x, y)^2/var(x))*n1/df)*
      (1/n+mean(x)^2/var(x)/n1))

    CL95intercept <- (intercept <- mean(y)-(slope)*mean(x))+c(1, -1)*
      qt(0.025, df)*(SEintercept)
    CL95slope <- slope+c(1, -1)*qt(0.025, df)*(SEslope <- sqrt(MSE/var(x)/n1))

    result1 <- c("slope"=slope, "SE[slope]"=SEslope, "95% LCL"=CL95slope[1],
                 "95% UCL"=CL95slope[2])
    result2 <- c("intercept"=intercept, "SE[int.]"=SEintercept,
                 "95% LCL"=CL95intercept[1], "95% UCL"=CL95intercept[2])
    list(RMAslope=result1, RMAintercept=result2)
}
```

We can use this new function to get a RMA slope and intercept for this relationship.

```
RMA (Width, Length)
```

```
## $RMAslope
##     slope SE[slope]   95% LCL   95% UCL
## 3.4443903 0.4018599 2.6301448 4.2586358
##
## $RMAintercept
##  intercept   SE[int.]    95% LCL    95% UCL
##  -2.407085   5.600919 -13.755624   8.941455
```

Note that the input here is similar to the plot command (i.e. function (x,y))

We will end here for now, but will save our workspace before quitting.

```
save.image("glm workspace.RData")
```