Bio R

2019

Introduction

Bio R

Welcome to the **Research Methods in Biology (BIO3011)** guide to using R to analyse biological data. We will be using RStudio during practical demonstrations, but the code in this book is set up so that you can also use the basic R program if you need to.

Changing working directory in R

Immediately on opening R, the first thing you need to do is change your working directory to the folder where your data is currently kept.

Mac: Select 'Misc' > Change working directory...
> *navigate to folder where your data is stored (i.e. your csv files)*

Windows: Select 'File' > Change dir...
> *navigate to folder where your data is (i.e. your csv files)*

Changing working directory in RStudio

Changing the working directory in RStudio is the same for Windows and Mac.

Both: Select 'Session' > Set working directory > Choose Directory...
> *navigate to folder where your data is stored (i.e. your csv files)*

Create a working R script file

In RStudio we can create a script file to work from. You also have the option of creating an R Workbook (which some people prefer), although we'll start simpler and just work with a script file for now.

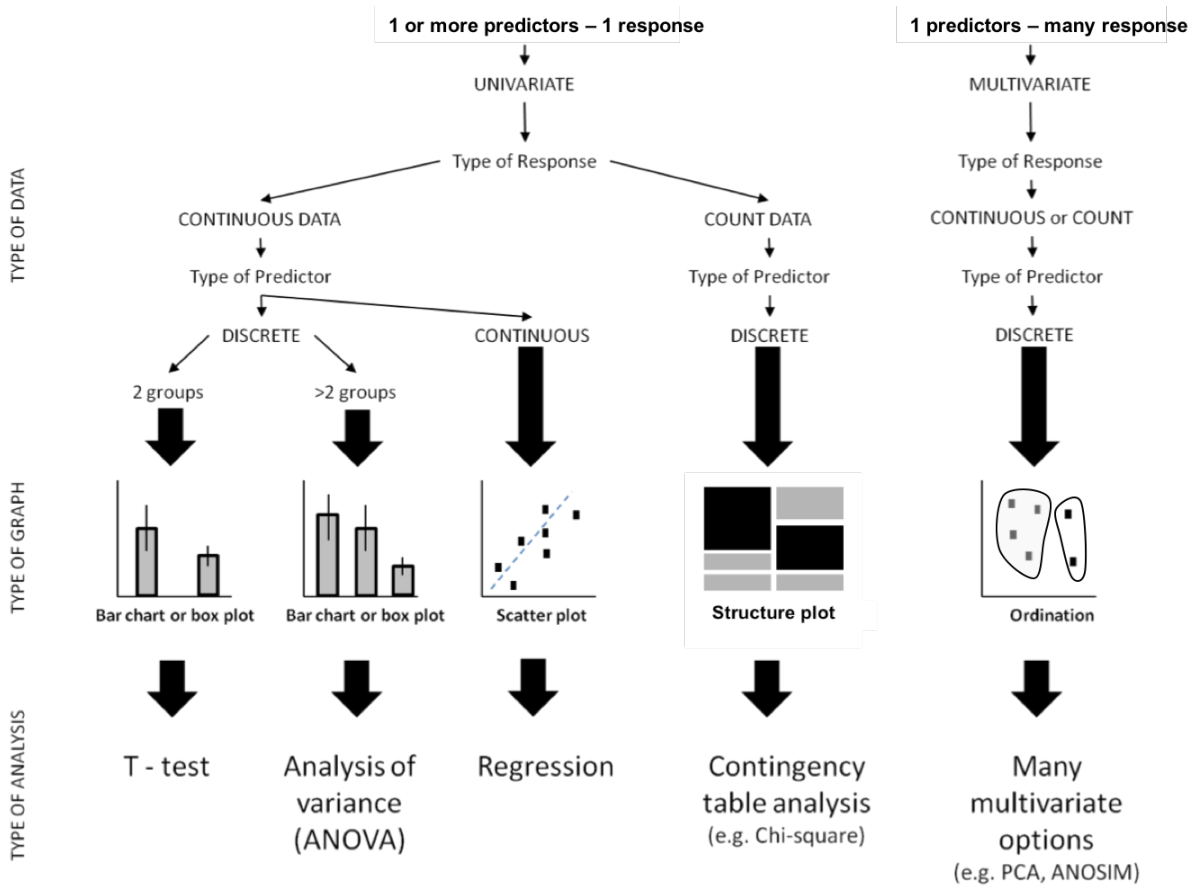
Select File > New File
> R Script

Then, either click the little blue save disc symbol or select **File > Save**. Give your R script a name and save it where you will be able to find it. It's usually a good idea to save your R script in the same folder as your data files.

Colour coding

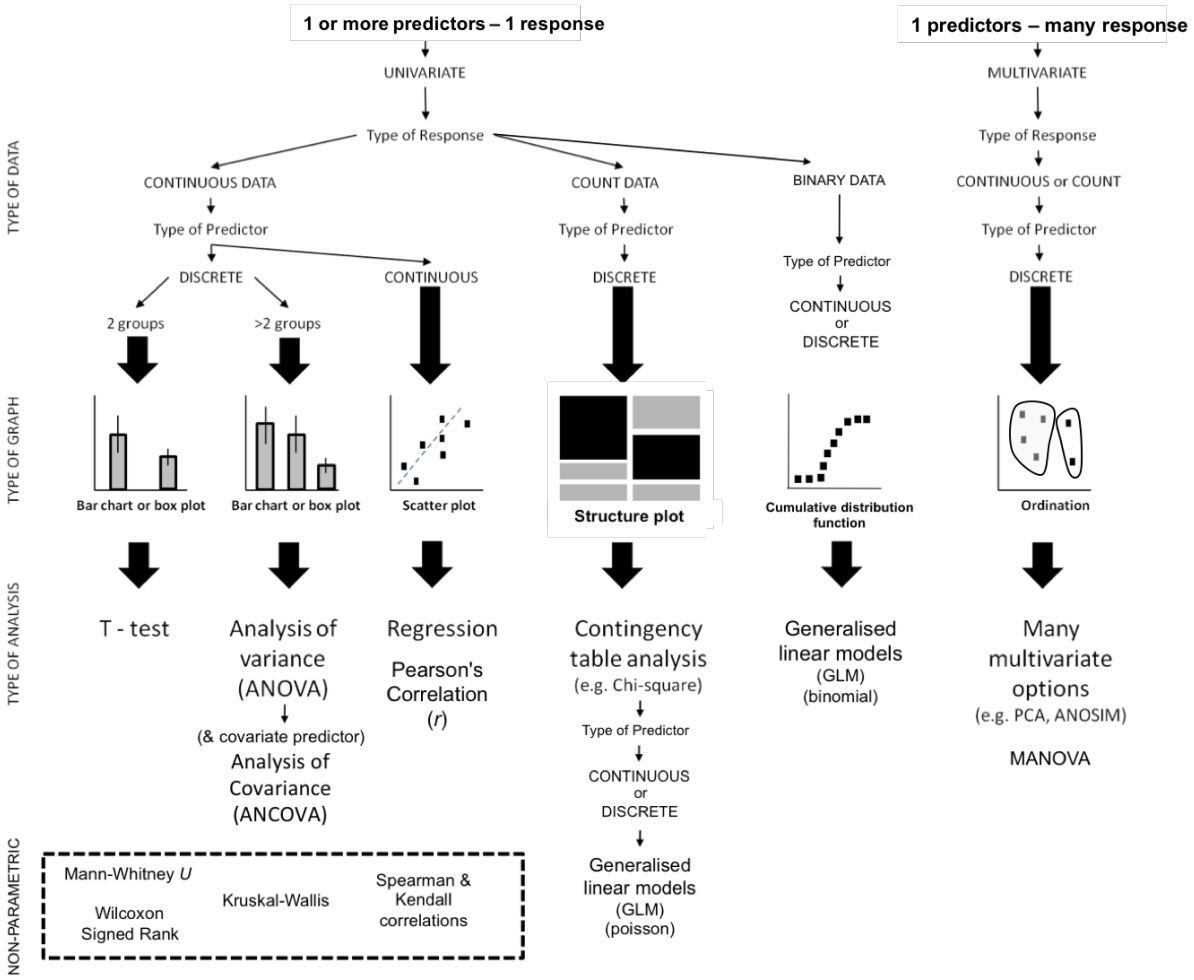
Code will be colour coded to make it easier to understand. Orange will be used for functions (commands), blue for anything that you can change or name yourself (the 'moving pieces' of the code, if you like), black for basic syntaxt requirements (arrows, brackets, commas mostly), green for comments (R doesn't read anything after a hash tag), and purple for libraries. For example:

```
library(vcd) # you need to install if you don't have it
attach(skink) # attaches the skink dataset
```

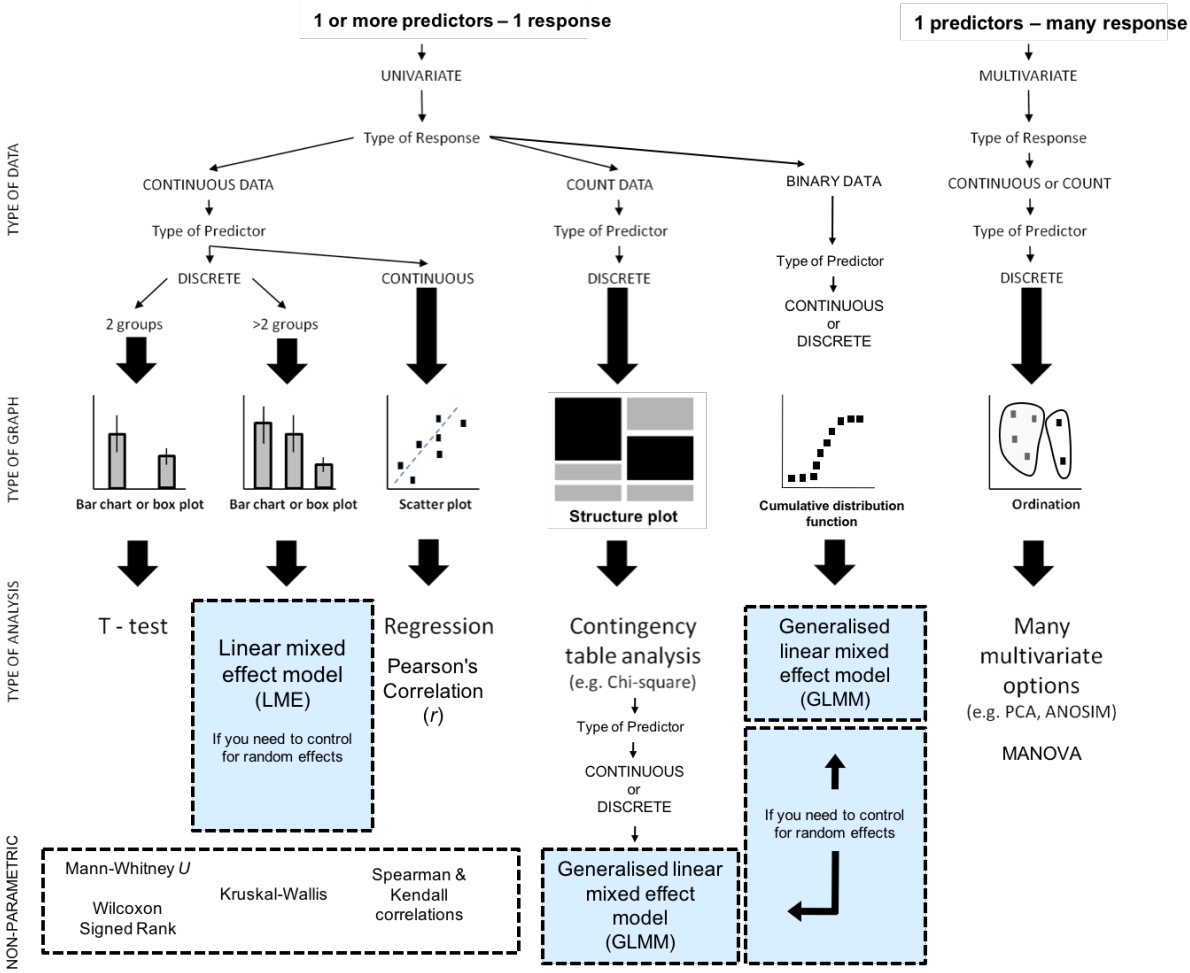


Basic Flow Diagram for Deciding Which Test to Use.

Predictors are also called independent variables. Responses are also called dependent variables.



Expanded Flow Diagram for Deciding Which Test to Use



Expanded Flow Diagram for Deciding Which Test to Use

Mixed Effects models are shaded blue

CODE QUICK REFERENCE SHEET

HISTORGRAMS

```
hist(yourdata$VARIABLE)
```

BOXPLOTS

```
boxplot(RESP ~ PRED, data = yourdata)
boxplot(RESP ~ PRED1*PRED2, data = yourdata, las = 2)
boxplot(RESP ~ PRED, data = yourdata, xlab="xlabel", ylab="ylabel", main="title")
```

SCATTERPLOTS

```
plot(RESP ~ PRED1, data = yourdata)
abline(lm(RESP~PRED1, data = yourdata))
plot(RESP ~ PRED1, data = yourdata, xlab="xlabel", ylab="ylabel", main="title")
```

STRUCTURE PLOTS

```
mosaicplot(your.xtab, shade=T)
```

INTERACTION PLOT (attach dataset first)

```
interaction.plot(XFACTOR, TRACE, RESPONSE)
```

SUBSETTING & CHANGING INTO A FACTOR

```
newdataset <- subset(yourdata, subset=yourdata$VARIABLE=='LEVEL')
yourdata$FACTOR <- as.factor(yourdata$NUMBER)
```

SUMMARIES

```
head(yourdata)
str(yourdata)
summary(yourdata)
mean(yourdata$VARIABLE)
sd(yourdata$VARIABLE)
sd(yourdata$VARIABLE) / sqrt(length(yourdata$VARIABLE)) )
```

CORRELATION

```
cor(yourdata$VARIABLE1, yourdata$VARIABLE2, method = "pearson")
cor(yourdata$VARIABLE1, yourdata$VARIABLE2, method = "spearman")
cor(yourdata$VARIABLE1, yourdata$VARIABLE2, method = "kendall")
round(cor(yourdata[,2:4]),2) # correlation grid: columns 2-4, round to 2 dp
```

COUNT DATA

```
your.xtab<-xtabs(RESP~PRED1+PRED2,data=yourdata)

fisher.test(your.xtab)

chisq.test(your.xtab, correct=F)
chisq.test(your.xtab, sim=T)
chisq.test(your.xtab)

chisq.test(your.xtab, correct=F)$res
chisq.test(your.xtab)$res

chisq.test(your.xtab, correct=F)$exp
chisq.test(your.xtab)$exp
```

ONE SAMPLE T-TEST

```
t.test(yourdata$VARIABLE)
```

TWO SAMPLE T-TEST

```
t.test(RESP~PRED,data=yourdata, var.equal=T)
t.test(RESP~PRED,data=yourdata)
t.test(yourdata$GROUP1,yourdata$GROUP2,paired=T)
```

GENERAL LINEAR MODELS

```
model.lm <- lm(RESP~PRED1*PRED2, data = yourdata)
model.lm <- lm(RESP~PRED1+PRED2, data = yourdata)
summary(model.lm)
anova(model.lm)
model.aov <- aov(RESP~PRED1*PRED2, data = yourdata)
model.aov <- aov(RESP~PRED1+PRED2, data = yourdata)
summary(model.aov)
TukeyHSD(model.aov, "FACTOR")
```

A note on 'rntransform'

The rank normal transformation from library GenABLE is used quite a bit in this book, however, the library doesn't appear to be being maintained at the time of writing this. I have salvaged the code out of an older working version of GenABLE and included it as the very end of this PDF.

If you want to create the rntransform function, just copy and paste the rntransform code included at the end of this file. You should find that 'rntransform' appears as a function in your working space.

A note on citations

This is not a peer-reviewed work. You should probably be hesitant about citing it in a peer reviewed journal. However, if you wish to, please use the following citation. This is updated each time a new version is released:

Johnstone, Christopher P. (2019) *Bio R: Statistical Stuff for the Biologically Minded* (version 2019.3). Self-Published.

Colour Coding

R code in this PDF is colour coded throughout. Functions (commands) are coloured **orange**. Names of objects or options (i.e. things that you will need to change or rename) are coloured **blue**. Basic syntactical signs (parenthesis, commas, plus or multiplication signs) are colour **black**. Package names are coloured **purple**. Note, which follow a hashtag (**#**) are not read by R, and serve as a place where you can make notes about your tests, code, and figures. These are coloured **green**.

In the following example, the orange words are functions that R will recognise as commands. Functions are typically placed to the immediate left of an opening parenthesis. The library '**car**' is denoted in purple, and is bracketed by black quote marks and parentheses. The object **swallows.lm** is an object that the biologist as created and named themselves.

```
install.pckages("car") # if not already installed
library(car)
crPlots(swallows.lm)
```

Significance Testing

For most biologists statistics is a tool. It is a means to an ends. What we're really interested in is the plants, animals or microbes, not the stats. As such, we try to keep things simple, while maintaining some rigor throughout the process of analysis.

What do we actually need to know?

1. We need to be able to identify 'classes' of data
2. We need to know how to pick an appropriate test
3. We need to know & check the assumptions of tests
4. We need to know how to run appropriate tests
5. We need to know how to present the results
6. We need to know how to interpret the results

Working backwards through these points

6. INTERPRETING RESULTS

A *P*-value can be thought of as the probability that the result could have been obtained by chance given the noise in the data.

- **Is there a difference between A and B?** $P < 0.05$ = Yes. $P > 0.05$ = No.
- **Are A and B showing a trend?** $P < 0.05$ = Yes. $P > 0.05$ = No.

5. PRESENTING RESULTS

Usually what we present is:

- Degrees of freedom (often n-1 but check your test results)
- An *n* if no degrees of freedom is produced (some tests don't us *df*)
- A test statistic
    - This is informative about the shape of the distribution. It is usually a ratio of the signal to noise in the data (higher = more signal, less noise)
    - *F*, *z* or *t* values are examples of test statistics
    - Usually reported to 2 dp (decimal places)
    - Test statistics are usually not interpreted in any way. They are presented in the results but not commented on in the Discussion.
- A *P*-value
    - Informative about significance or non-significance
    - Usually reported to 3 dp.
    - If less than 0.001, write < 0.001.

Why don't we just use a test statistic?

If the test statistic is a ratio of signal to noise, why use a *P*-value at all? The *P*-value depends on both the test statistic and the degrees of freedom. Using an made-up example where the *F* ratio is the test statistic...

Table 1. Example of how changes in the test statistic (ratio of signal to noise) and degrees of freedom (information in the sample) can alter the P value (probability of having obtained a result if the null were actually true)

t value	df	*P*	
2	5	0.501	
4	5	0.005	*Higher t but same df*
2	10	0.037	*Same t but higher df*

You can try these examples in R. Note how the P value changes as the degrees of freedom changes. The test statistic has been kept the same at 2.3. Note also that we need 1 minus the probability (`pt`) because R returns the beta rather than the alpha by default.

```
# t-tests
1-pt(2.3, 5)
1-pt(2.3, 10)
1-pt(2.3, 20)
1-pt(2.3, 200)

# t-tests
1-pt(1.3, 6)
1-pt(2.3, 6)
1-pt(3.3, 6)
1-pt(4.3, 6)
```

Degrees of freedom, test statistics and P-values can also be presented as tables:

Table 2. Example of how to present results in a table form. Instead of writing P < 0.05 in the table you could include a line here in the caption stating: alpha < 0.05 indicated by *

Response	Predictor	df	*F*	*P*	*P* < 0.05
Mass (g)	Fledging date	11	6.76	0.010	*
	Habitat type	1	1.62	0.205	
Hct (%)	Fledging date	11	15.31	< 0.001	*
	Habitat type	1	2.10	0.150	
Hb (%)	Fledging date	11	7.80	0.005	*
	Habitat type	1	3.28	0.072	

Test statistics can be summarized as a group in the text, especially if they are non-significant:

All tests of the effect of habitat on mass and blood variables for adult house swallows were non-significant (all $df = 1$, $F < 3.28$, $P > 0.072$). All tests of the effect of fledgling date on mass and blood variables for adult house swallows were significant (all $df = 11$, $F > 6.76$, $P < 0.010$).

Note how the greater than and less than signs are arranged. If a set of tests are significant we are interested in the smallest *F* (test statistic) and largest *P*. If a set of tests are non-significant we are interested in the largest *F* and smallest *P*.

How to write and talk about significance

- There is no such thing as 'more' or 'less' significant. A *P*-value either meets a significance level (usually 0.05) or it doesn't.
- If a test is not significant we say that it is 'non-significant' not 'insignificant'.
- Being significant doesn't make something true. Significance only makes it more likely given the evidence.
- Non-significant results can be informative too.

4. RUNNING APPROPRIATE TESTS

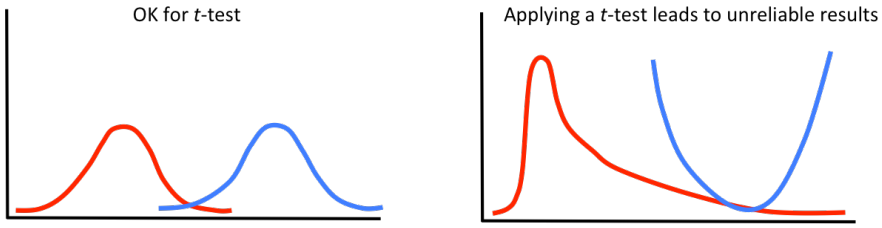
This will make up the bulk of the statistical sections of this pdf.

3. TESTING ASSUMPTIONS

Each statistical test has a set of assumptions about the data that must be checked before running the test. Assumptions are not always the same among tests. For example, for a *t*-test the response data must be normally distributed, for an ANOVA the residuals of the model must be normally distributed, and for a chi-squared test there is no requirement for the data to be normal at all.

Why test assumptions?

Statistical tests make assumptions about the 'shape' of the data and/or the residuals of the model... for example, a t-test assumes that the data will consist of two (approximately) normally distributed sets for comparison:



The example of the left would probably be acceptable for a *t*-test, but the example on the right would generate a *P*-value that would be meaningless nonsense. The key problem is that often a test will still be able to generate results when the data is unsuitable and if you don't stop to check the suitability of the data ahead of time your results may be extremely misleading. Remember also that assumptions are different for different tests. The assumptions for a t-test are not the same as a generalised linear model or a chi squared test.

Do some tests have no assumptions?

Some tests have very few (although independence of observations is always a given). A Kruskall-Wallis test for example has almost no assumptions about the underlying data, but still requires independence of observations.

How do we test independence of observations?

Although tests do exist to identify whether a set of values might have a correlation structure (perhaps spatial or temporal autocorrelation, for example), the key to avoiding the problem of pseudoreplication is good experimental design. Care needs to be taken to think through the nature of the design and in what ways samples might be interacting.

2. PICKING AN APPROPRIATE TEST

This will depend on whether your data meets the assumptions of a test. At its most fundamental, you need to first look at the response data and decide if it is continuous, binomial or count data. This will determine what sorts of tests you look at as potential first options.

1. IDENTIFY CLASSES OF DATA

You have been provided with a handy flow chat to help you work out which test to use under different circumstances but the first step is identifying the class of data you have. These are the basic types of data you may be working with:

Discrete data. This is an umbrella category that includes **categorical**, **binomial**, **ordinal** and also **count** data (although counts tend to be handled differently to the others).

Categorical. Also called **nominal data**. R classifies categorical data as a **factor**. Categorical data consists of categories that have no relative positions to each other. Usually there are three or more categories. If there are two categories then the data becomes **binomial**. Categorical data will typically be a predictor (independent) in an analysis. An example of categorical data would be if you wanted to test bee pollination of three species of flowers. The flowers don't exist on a scale or spectrum, they are simple Species A, B and C forming three categories. In this example R would identify this as a factor (flower species) with three levels (species A, B and C).

Binomial. Also called **dichotomous data**. Binomial data consists of two discrete categories. Yes and No responses, Male and Female or species site Presence and Absence data are all examples of binomial data. If you enter binomial data using words or letter R will identify it as a **factor**. If you enter Binomial data as 0 and 1 R will identify binomial data as an **integer**.

Ordinal. Data that is ordered or ranked. Instead of measuring the length of fallen logs in a field site, you might decide to rank them from largest to smallest. Ordinal data can accept ties. If you decided two logs were so similar they were in effect the same size you could rank them 1, 2, 3, 3, 4, 5 and 6. R will identify ordinal data as an **integer**.

Count. Count data is a sub-type of ordinal data where you have counted instances of something. R will identify count data as an **integer**.

Continuous. Also called **interval data**. Continuous data is measured on a scale. It can be both positive and negative (e.g. temperature) or only positive (e.g. mass). If your data has (or could have) decimal places, R will identify it as a **number**.

Percentages. Percentage data is a sub-type of **continuous data**. Percentages are often generated from observations of count data or continuous data. Percentages are bounded and non-normal, which breaks assumptions of a number of tests. They often need to be transformed. The standard transformation for a Percentage is an arcsine square root transformation. R will identify percentage data as a **number**.

Ratios. Percentage data is a sub-type of **continuous data**. Ratios are often generated from observations of count data or continuous data, especially where one variable is measured against another variable. Ratios are bounded by zero and they are non-normal, which breaks assumptions of a number of tests. They often need to be transformed. The standard transformation for a Ratio is a square root transformation (if zeroes are present) or a log transformation (if no zeroes are present). R will identify Ratio data as a **number**.

Assumptions

Most (nearly all) statistical analyses have **assumptions**. An assumption is a requirement that the data must satisfy for the test to be valid. Assumptions vary among tests but most frequently encountered assumptions are:

- **Independence:** The samples must be independent of one another. This needs to be considered during the design phase. This is a consideration of almost all tests. Where it isn't a consideration, the test will probably be intended to examine the degree of departure from independence (i.e. spatial autocorrelation tests).
- **Normality:** The data and variances of the data should be (reasonably) normally distributed. This can be tested in R.
- **Homogeneity and Homoscedasticity of variance:** Are the variances of groups (reasonably) similar? In particular, there will be problems if the variance is larger in groups with larger means. This can be tested in R.
- **Linearity:** In analyses of **continuous data**, such as in a regression analysis, linearity of the continuous data is important (i.e. the response data should not plot a curve against the explanatory data). This can be tested in R.
- **No correlation of predictors:** Explanatory or predictor variables should not **correlate**. Multicolinearity occurs when predictors co-occur or co-correlate. For example, most smokers carry cigarettes and a lighter. Using both 'cigarette carrying' and 'lighter carrying' as predictors is invalid because they are correlating and 'explain' the same thing: the behaviour of smoking. Correlating predictors are sometimes referred to as **confounding variables** or **nuisance variables**. Sometimes the best option is to acknowledge in the methods that you collected data for A and B, but found that A and B were correlating (typically we worry if a correlation coefficient for two predictors is > 0.6), so the variable of least interest was dropped prior to analysis.

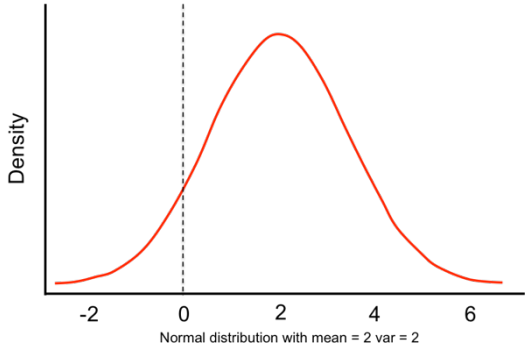
Expectations for data

Most assumptions apply to response data:

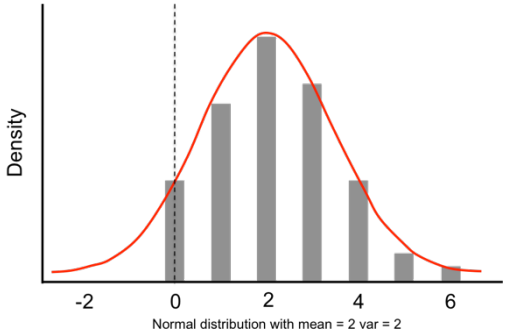
- What sort of response data do you have?

CONTINUOUS DATA	BINARY DATA	COUNT DATA
↓	↓	↓
Expect normality	Can't be normal	Can't be normal
↓	↓	↓
Normal distribution	Binomial distribution	Poisson distribution
↓	↓	↓
Test for normality	Don't test for normality	Don't test for normality

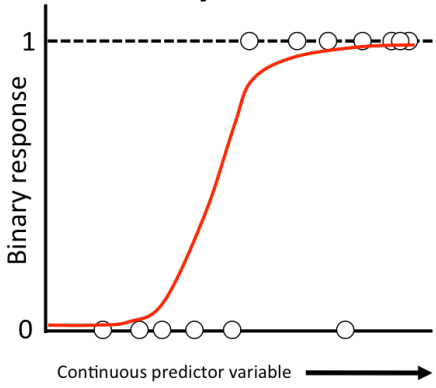
We expect continuous data to follow a normal distribution.



But count data cannot be negative, so we can't expect counts to be normally distributed. Counts will run up against the zero line.



And we wouldn't expect a binomial distribution of 0s and 1s to look anything like a bell shaped curve.



Frequency analyses

Goodness of Fit :: Count Data

Count data is produced when you have a response variable that is simple a count of items (trees in a plot, count of behaviours in a ten-minute period, insects in a trap). All of these variables could potentially be analysed with ANOVAs (mostly because ANOVAs are robust tests and can tolerate reasonably non-normal data), but usually if you did want to apply a linear model to count data, the count data will need to be scaled and/or centred in some way (either via percentages or applying transformations in R).

Frequency analyses are a more appropriate choice. However, these require counts to be measured against a category, rather a continuous predictor. Typically this will be a count that is classified and then summed up, but depending on your level of independence you may need to average courts by category instead. A frequency analysis does accept fractions, so that an average of 5.3 counts for a category (for example) could be used without problems.

To start us off, imagine you've done a field count of plants and have found 50 females and 40 males. We might want to ask whether this differs from the expected ratio of 1:1. To answer this we can calculate a chi-squared value and then derive a P value from it.

Chi Squared = Sum (observed - expected) ^{2} / expected

A chi-squared is an example of a test statistic. However, instead of being a ratio of **signal to noise** (like a *t*-value or an *F*-statistic) a chi-squared value is a ratio of the divergence of observed values from the expected values. This makes it something closer to being a ratio of **signal to expectation**, and it can be thought of as addressing the question: how much does this set of counts diverge from expected values?

A straightforward chi-squared tested is sometimes called a **goodness of fit** test, but 'goodness of fit' is a general term for any of a number of methods that test how well data fists a model. For example, an **Akaike information criterion (AIC)** is also a **goodness of fit** index, as is an R^2 value, though these work quite differently to a chi-squared. For now, it is enough to know that a chi-squared is just one approach to questions about the **goodness of fit** of data to the expectations of an underlying model.

ASSUMPTIONS OF CHI-SQUARED TEST

(1) Categories are independently classified (collected randomly)

(2) No more than 20% of expected values can be < 5

Using our example above, let's test whether 50 female plants and 40 male plants departs from the expected 1:1 ratio.

Chi-squared goodness of fit tests against an expected ratio. A ratio of 1:1 is the default.

```
chisq.test(c(40,50))
```

RESULT

```
     Chi-squared test for given probabilities

data:  c(40, 50)
X-squared = 1.1111, df = 1, p-value = 0.2918
```

The percentage of males and females:

```
40/90
50/90
```

RESULT

```
> 40/90
[1] 0.4444444
> 50/90
[1] 0.5555556
```

What if we collected the same ratio, but from 900 bushes?

```
chisq.test(c(400,500))
```

RESULT

```
> chisq.test(c(400,500))

     Chi-squared test for given probabilities

data:  c(400, 500)
X-squared = 11.111, df = 1, p-value = 0.0008581
```

But remember: never use percentages of the data or reduce the data down to a basic ratio, because this alters the results. You lose information about the weighting of the number of samples used to generate the data.

```
chisq.test(c(44,56))
```

RESULT

```
     Chi-squared test for given probabilities

data:  c(44, 56)
X-squared = 1.44, df = 1, p-value = 0.2301
```

What if on the other hand we expected a 30:70 male to female difference? Maybe another researcher found a 30:70 male to female ratio in a closely related species and we want to check whether our species is different to this. Remember that order matters. Male observed = 40. Male expected = 30% (i.e. a proportion of 0.3). Female observed = 50. Female expected = 70% (i.e. a proportion of 0.7).

```
chisq.test(c(40,50) ,p=c(0.3,0.7))
```

RESULT

```
     Chi-squared test for given probabilities

data:  c(40, 50)
X-squared = 8.9418, df = 1, p-value = 0.002787
```

Chi-squared tests using a table of observations

A chi-squared test can test divergence from expectation in two dimensional arrays of numbers too. Let's construct a simple 2x2 matrix. Let's say we have data that looks like this:

Table 1. Contingency table showing counts of male and female plants of the same species with pink and white flowers.

	Male	Female
Pink	23	45
White	34	67

To run a chi-squared test we need to create a matrix. We will do this in R and call it 'flowers'.

```
flowers <- matrix(c(23,34,45,67), nrow=2)
flowers
```

RESULT

```
> flowers
     [,1] [,2]
[1,]   23   45
[2,]   34   67
```

Apply a chi-squared test

```
chisq.test(flowers)
```

RESULT

```
     Pearson's Chi-squared test with Yates' continuity
correction

data:  flowers
X-squared = 8.3917e-31, df = 1, p-value = 1
```

This applies 'Yate's continuity correction' as a default, but this correction has fallen out of fashion among statisticians. We don't need to use it, so let's remove the correction.

```
chisq.test(flowers, correct=F)
```

RESULT

```
     Pearson's Chi-squared test

data:  flowers
X-squared = 0.00046639, df = 1, p-value = 0.9828
```

Note that you could run the entire matrix inside the chi-squared test, but this approach can be confusing. It's best to take things a step at a time until you are used to how the code should look and where errors might happen. However, the whole code string as one line would look like this:

```
chisq.test(matrix(c(23,34,45,67), nrow=2), correct=F)
```

RESULT

```
> chisq.test(matrix(c(23,34,45,67), nrow=2), correct=F)

     Pearson's Chi-squared test

data:  matrix(c(23, 34, 45, 67), nrow = 2)
X-squared = 0.00046639, df = 1, p-value = 0.9828
```

We can also use piping from the 'tidyverse' family of libraries. Some people find the piping method easier to use, and if you are used to piping, you can certainly make use of it here. Remember to load the packages using `library("tidyverse")` first.

```
matrix(c(23,34,45,67), nrow=2) %>% chisq.test(correct=F)
```

RESULT

```
     Pearson's Chi-squared test

data:  .
X-squared = 0.00046639, df = 1, p-value = 0.9828
```

Bootstapping a Chi-squared Test

If the assumption that no more than 20% of expected values less than 5 is not met, you will see an error stating that the P value may not be correct. For example, try this…

```
flowers_2 <- matrix(c(5,7,45,2), nrow=2)
flowers_2
chisq.test(flowers_2, correct=F)
```

RESULT

```
> flowers_2
     [,1] [,2]
[1,]    5   45
[2,]    7    2

> chisq.test(flowers_2, correct=F)

     Pearson's Chi-squared test

data:  flowers_2
X-squared = 21.625, df = 1, p-value = 3.315e-06

Warning message:
In chisq.test(flowers_2, correct = F) :
  Chi-squared approximation may be incorrect
```

Maybe we want to look at the expected values to check where the problem is. We can use the **$** symbol to look inside a test, just like how you can use it to look inside a dataset.

```
chisq.test(flowers_2, correct=F)$expected # $exp will also work
```

RESULT

```
          [,1]      [,2]
[1,] 10.169492 39.830508
[2,]  1.830508  7.169492

Warning message:
In chisq.test(flowers_2, correct = F) :
  Chi-squared approximation may be incorrect
```

One of the expected values is <5 (1.83). Because there are only four values, this means that 25% of the values are <5. The assumption that no more than 20% of expected values be 5 or less has not been met.

You can use the **$** to look at other variables in the test as well. To find out what you can view, use the structure command (**str**) on the test.

```
str(chisq.test(flowers_2, correct=F))
```

Or you could turn the test into an object and look at the structure this way…

```
flowers_2_test <- chisq.test(flowers_2, correct=F)
str(flowers_2_test)
```

Or you could use tidyverse piping, like this…

```
flowers_2 %>% chisq.test(correct=F) %>% str()
```

These should all give the same result (note that the warning will keep repeating because the people who have programmed R are rather thorough, and they don't want you to miss the warning by accident):

RESULT

```
> str(chisq.test(flowers_2, correct=F))
List of 9
 $ statistic: Named num 21.6
  ..- attr(*, "names")= chr "X-squared"
 $ parameter: Named int 1
  ..- attr(*, "names")= chr "df"
 $ p.value  : num 3.31e-06
 $ method   : chr "Pearson's Chi-squared test"
 $ data.name: chr "flowers_2"
 $ observed : num [1:2, 1:2] 5 7 45 2
 $ expected : num [1:2, 1:2] 10.17 1.83 39.83 7.17
 $ residuals: num [1:2, 1:2] -1.621 3.821 0.819 -1.931
 $ stdres   : num [1:2, 1:2] -4.65 4.65 4.65 -4.65
 - attr(*, "class")= chr "htest"
```

Try looking at the residuals. The approach is the same as for expected values.

```
chisq.test(flowers_2, correct=F)$residuals # $res will also work
```

So, what do you do if the assumption that no more than 20% of the values should be 5 or less? One potential workaround is to compute a p-value by Monte Carlo simulation and use this for the chi-squared test. R has a built-in function that allows you to do this:

To use a Monte Carlo simulation, try this:

```
chisq.test(flowers_2, correct=F, simulate.p.value=T)
# note that sim=T will also work
```

RESULT

```
     Pearson's Chi-squared test with simulated p-value
(based on 2000 replicates)

data:  flowers_2
X-squared = 21.625, df = NA, p-value = 0.0004998
```

The above test result was generated from 2000 simulations of the data using Monte Carlo picking. We might want to increase this value. Let's try 10,000 simulations. The test uses **B** to indicate number of simulations for the Monte Carlo test. Usually 10,000 is considered a suitable number for any kind of bootstrapping or simulation. If you don't see a significant result at 10,000 simulations, it probably isn't there to see.

```
chisq.test(flowers_2, correct=F, simulate.p.value=T, B=10000)
```

RESULT

```
     Pearson's Chi-squared test with simulated p-value
(based on 10000 replicates)

data:  flowers_2
X-squared = 21.625, df = NA, p-value = 9.999e-05
```

A Fisher Exact Test will also allow you to get around problems of not meeting the assumption that no more than 20% of expected values be 5 or less, but this test doesn't work with all configurations of data. The Monte Carlo simulation (above) is more flexible.

```
fisher.test(flowers_2)
```

But how do we interpret this?

The P value for the Chi Squared test is telling us whether there is any significant departure at all of observed from expected values. But what if we want to understand which categories are driving a significant result. We can do this visually using mosaic plots, but a mosaic plot is simply a visual representation of residuals. What are the residuals and how do we interpret them?

You can check residuals in the same way that you checked expected values... let's start by going back to our original flowers matrix.

```
flowers <- matrix(c(23,34,45,67), nrow=2)
flowers

chisq.test(flowers, correct=F)$residuals
```

RESULT

```
     > chisq.test(flowers, correct=F)$residuals

             [,1]          [,2]
[1,]  0.01359119 -0.009695849
[2,] -0.01115196  0.007955723
```

Any residual that is between -2 and -4 or between +2 and +4 is significant at the 0.05 level. Any residual that is less than -4 or above +4 is significant at the 0.01 level. If you remember, our first flowers dataset did not significantly depart from expected values (P = 0.983), so it is no surprise to discover that there are no significant differences in the residuals.

Let's try a different dataset, one with more rows and columns and more extreme differences in values.

Table 2. Contingency table showing counts of male and female plants of the same species with pink and white flowers.

	Male	Female
Red	2	10
Pink	15	45
White	38	7

```
flowers_3 <- matrix(c(2,15,38,10,45,7), nrow=3)
flowers_3
```

RESULT

```
> flowers_3 <- matrix(c(2,15,38,10,45,7), nrow=3)
> flowers_3

     [,1] [,2]
[1,]    2   10
[2,]   15   45
[3,]   38    7
```

Chi Squared Steps

1) Check the assumptions (i.e. look at the expected values)
2) Decide, can you use a standard Chi Squared test or Monte Carlo simulations?
3) Run the appropriate test
4) Check the residuals
5) Produce a mosaic plot

```
# 1) are 20% of values < 5?
chisq.test(flowers_3, correct=F)$expected

# 2) No expected values are 5 or less. Assumptions are met.

# 3) Run the test. There is a significant result.
chisq.test(flowers_3, correct=F)

# 4) Check the residuals.
chisq.test(flowers_3, correct=F)$residuals

# 5) Produce a mosaic plot.
mosaicplot(flowers_3, shade=TRUE)
```

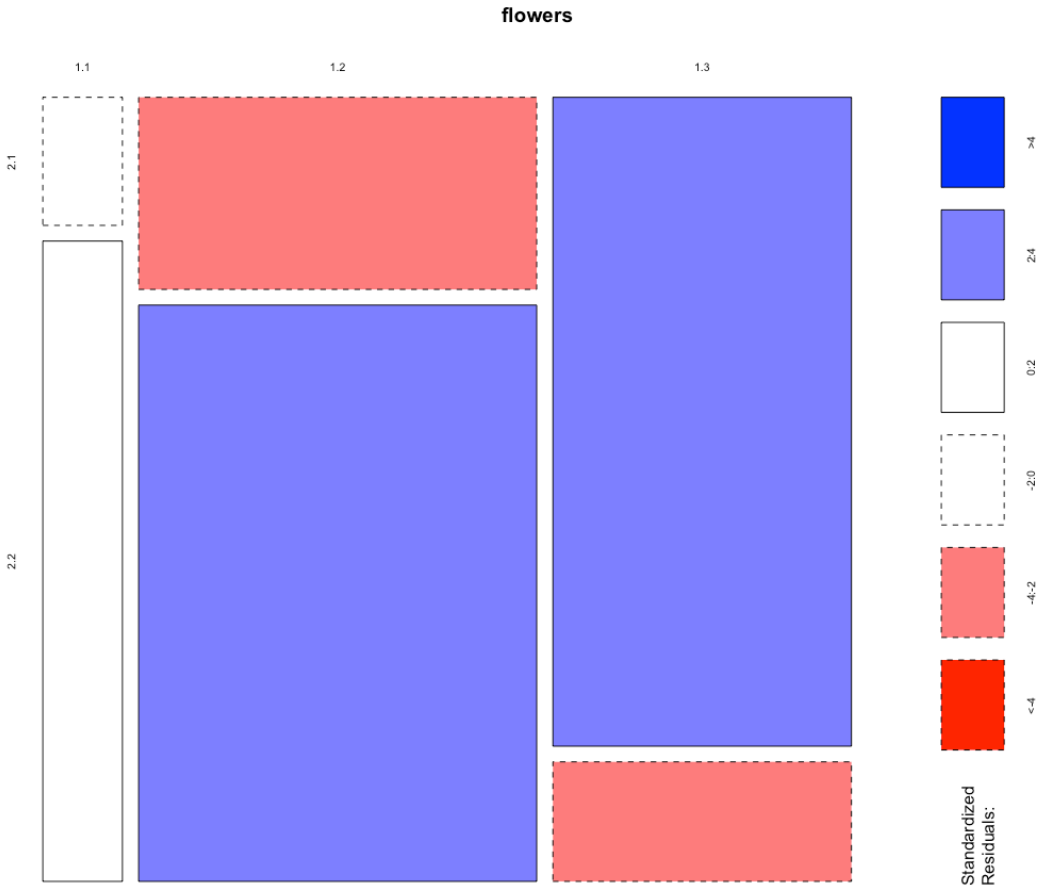


fig. 1. A mosaic plot showing values above and below expected for distributions of flower colours in a species of dioecious angiosperm ($\chi^2 = 41.4$, $df = 2$, $P < 0.001$).

Note that the chi letter (χ) is used to denote the chi squared test statistic. Now, one obvious problem with this plot is that R has no idea what the rows and columns should be called. This makes it hard for us too. But we can rename the row and column names in R. Try this…

```
rownames(flowers_3) <- c("Red", "Pink", "White")
colnames(flowers_3) <- c("Male", "Female")
flowers_3
```

RESULT

```
      Male Female
Red      2     10
Pink    15     45
White   38      7
```

```
mosaicplot(flowers_3, shade=TRUE)
```

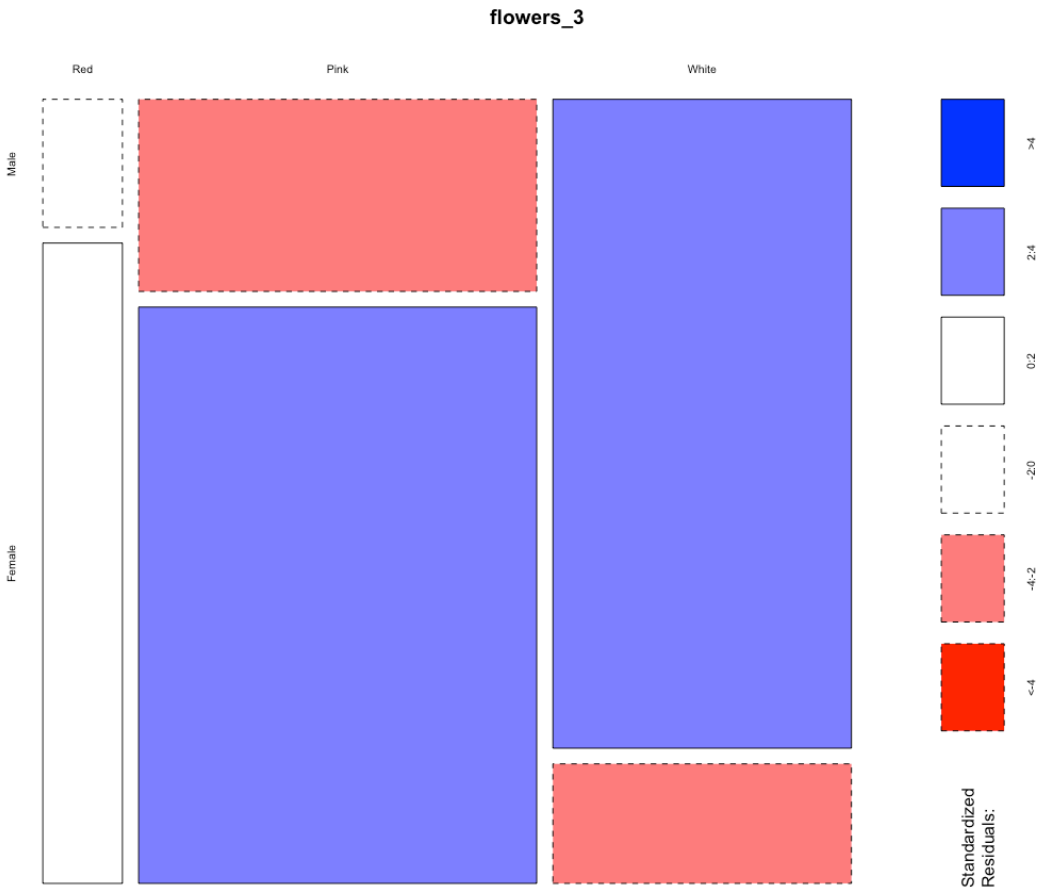


fig. 2. A mosaic plot showing values above and below expected for distributions of flower colours in a species of dioecious angiosperm (χ^2 = 41.4, *df* = 2, P < 0.001). Red flowers in males (res = -1.5) and females (res = 1.4) did not differ from expected. Pink flowers occurred at numbers significantly below expected in males (res = -2.5 and above expected in females (res = +2.4). Contrastingly, white flowers were at numbers significantly above expected in males (res = +3.7) and below expected in females (res = -3.4)

The **shade=TRUE** component of the code is telling R that you want it to produce boxes that show blue for significantly above expected and red for values significantly below expected. If we compare the box colours to the residuals, we will see that the colours will match. Note that **col=TRUE** simply paints half the boxes dark or light grey, and isn't meaningful in terms of interpretation.

It's also worth being aware that the residuals are in the unit of the original count too. So, that there were 3.4 fewer white flowers than expected in the dataset, given the distribution of all other flower colours by plant sex.

Using data from a comma delineated (csv) file

Let's look at an example using real data. Import the myna pecking data set (myna_peck.csv). This dataset contains a count of the number of times common mynas were pecking in a 10 second observation period. There is a set of explanatory variables included as well, but the primary variable of interest is whether the common mynas were observed in urban, suburban, periurban or rural landscapes.

```
myna <- read.table('mynas_peck.csv',header=T,sep=',')
str(myna)
```

Because this is a balanced design (there are 40 observations per landscape category) we can (arguably, see below) sum the values together to generate a contingency table.

```
myna.xtab <- xtabs(PECK.rate~REGION,data=myna)
myna.xtab
```

Check expected values.

```
chisq.test(myna.xtab, correct=F)$exp
```

Run a chi squared test.

```
chisq.test(myna.xtab, correct=F)
```

We can look at the residuals.

```
chisq.test(myna.xtab, correct=F)$res
```

```
REGION
 Peri-urban       Rural    Suburban       Urban
-0.50529115  0.08421519  0.50529115 -0.08421519
```

The residuals indicate that pecking was slightly below expected in peri-urban and urban environments and slightly above expected in rural and suburban environments, but the difference isn't significant. Library vcd has a nice looking mosaic plot option:

```
library(vcd)
strucplot(myna.xtab,shade=T)
```

You will have received an error message after running the structure plot. This hasn't worked because there is only one predictor. You need to take the residuals from the chi-squared test and use them in the structure plot. This code takes the residuals and drops them into a new object we've created and called RES.

```
RES <- chisq.test(myna.xtab, correct = F)$res
strucplot(myna.xtab, shade = T, residuals = RES)
```

fig. 3. A mosaic plot showing values above and below expected for rates of mynas pecking per 10 seconds (a measure of foraging activity) (χ^2 = 0.53, *df* = 3, P = 0.913). The foraging activity has been summed across four regions: urban, suburban, rural and peri-urban. No values were found to be significantly above or below expected. All residuals were between -0.51 and +0.51.

One confusing element of this graph is that there is a **p-value = < 2.22e-16** down at the bottom of the residuals bar. This is false and has been generated because we forced the mosaic plot to accept residuals to get around the error message above. Because nothing is significant, it may be preferable to just present the plot without a residual bar (i.e. remove **`shade = T`**). If you were determined to use this plot, you'd need to either include a note in the caption that the P value shown is the maximum possible significance value in R and in this instance it is not indicating significance here, or, simply remove it in an image manipulation program.

Summing or averaging?

Arguably, we may have broken the assumption of independence by summing the peck counts just now. The pecks are counts per bird per hour. As a bird is likely to be pecking at a given rate because it is feeling more or less safe in the environment (i.e. a bird that isn't worried about predation is more likely to spend time feeding), then the peck counts are probably not strictly independent. How might we solve this? One obvious (and perhaps the most straightforward) approach is to create a new csv file, average out all the pecks by geographic area, and use that. However, we can do something similar in R, although it takes a bit of code. Have a go at this...

```
# library plyr is part of the tidyverse, so if you already have
tidyverse loaded, plyr should already be running. If not, load it:

library(plyr)

attach(myna)
myna.average <- ddply(myna, .(REGION), summarise,
  PECK.MEANS = mean(PECK.rate))

# Check that this worked
myna.average
```

RESULT

```
> myna.average
       REGION PECK.MEANS
1 Peri-urban      3.375
2       Rural      3.550
3    Suburban      3.675
4       Urban      3.500
```

```
# Turn into a dataframe
myna.average <- as.data.frame(myna.average)

myna.xtab <- xtabs(PECK.MEANS~REGION,data=myna.average)
myna.xtab
```

Check expected values (note that R recognised 'exp' as an abbreviation of 'expected').

```
chisq.test(myna.xtab, correct=F)$exp
```

Run a chi squared test.

```
chisq.test(myna.xtab, sim=T, B=10000)
```

We're getting an error message. What does this mean?

RESULT

```
chisq.test(myna_average_pecks, sim=T, B=10000)

     Chi-squared test for given probabilities with simulated
p-value (based on 10000 replicates)

data:  myna_average_pecks
X-squared = 0.013121, df = NA, p-value = 1.007

Warning message:
In matrix(sample.int(nx, B * n, TRUE, prob = p), nrow = n) :
  data length [141000] is not a sub-multiple or multiple of
the number of rows [14]
```

This warning simply means that there are a small number of 'recycled' values in the matrix of 10,000 simulations that is generated. This is because the total length of the data is not an even multiple of the number of rows. To all intents and purposes, you can ignore this message, especially where the result is highly non-significant (as here) or you have a P <0.01. If you do get a marginal significance (i.e. around P = 0.04 to 0.05), then you might want to worry about resolving this mathematically, but in that case you may need to talk to a mathematician.

Note that you could do the same manually with the averages, generating a matrix:

```
myna_average_pecks <- matrix(c(3.375, 3.550, 3.675, 3.500))

rownames(myna_average_pecks) <- c("Peri-urban", "Rural",
"Suburban","Urban")

colnames(myna_average_pecks) <- c("Average_Pecks")

myna_average_pecks
```

Check expected values.

```
chisq.test(myna_average_pecks, correct=T)$exp
```

Run a chi squared test.

```
chisq.test(myna_average_pecks, sim=T, B=10000)
```

Further Example: Another Flowers Matrix

Have a go at these examples if you want some practise.

Let's create the following matrix.

	Male	Female
Red	2	4
Pink	5	16
White	18	1

```
flowers <- matrix(c(2,5,18,4,16,1), nrow=3)
flowers
```

Chi squared steps

1) Check the assumptions (i.e. look at the expected values)
2) Decide, can you use a standard Chi Squared test or Monte Carlo simulations?
3) Run the appropriate test
4) Check the residuals
5) Produce a mosaic plot

Further Example: Wildebeest Dataset

Import the wildebeest.csv dataset. Some researchers looked at whether there was a sex difference in the number of wildebeest calves that died from predation versus other causes. Let's see if we can identify any significant departures of observed from expected values.

```
wildebeest <- read.table('wildebeest.csv',header=T,sep=',')
str(wildebeest)
```

Chi squared steps

1) Create a contingency table (use the **xtab** function). You need to use a + because we want to divide the predictor variables along two categories:

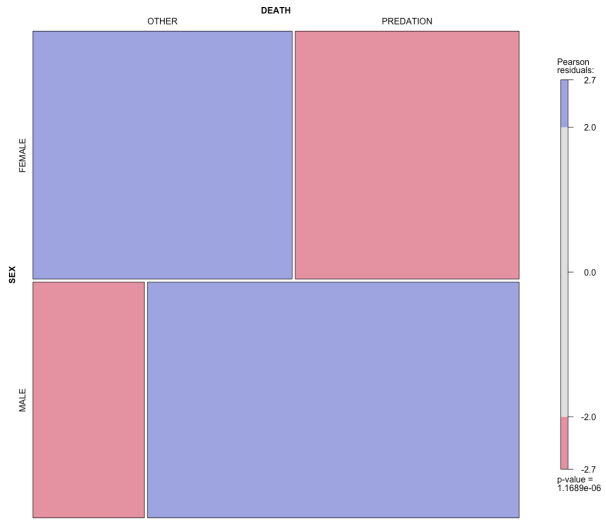
```
wildebeest.xtab <- xtabs(COUNT~SEX+DEATH,data=wildebeest)
```

2) Check the assumptions (i.e. look at the expected values)
3) Decide, can you use a standard Chi Squared test or Monte Carlo simulations?
4) Run the appropriate test
5) Check the residuals
6) Produce a mosaic plot

Library vcd

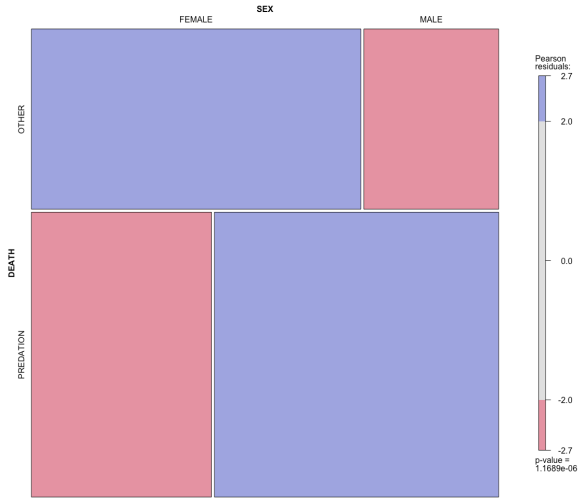
Library vcd has a couple of nice visual options for presenting contingency tables. You can generate a mosaic plot using the **strucplot** command, and another option, **assoc** will split up a mosaic plot and lay it out so that it is easier to read.

```
wildebeest.xtab <- xtabs(COUNT~SEX+DEATH,data=wildebeest)

library(vcd)

strucplot(wildebeest.xtab, shade=T)
```

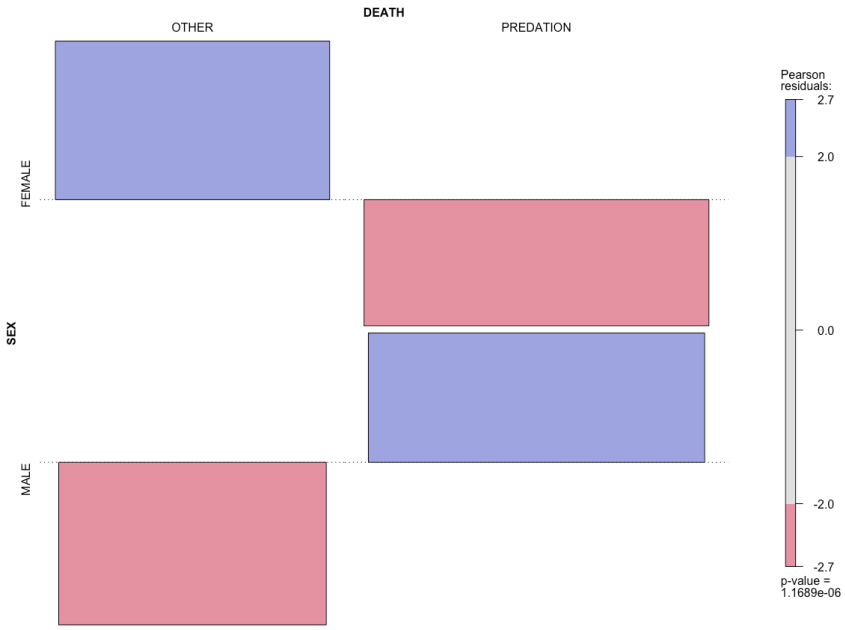


If you reorder the predictors in the contingency table, the axes will flip:

```
wildebeest.xtab <- xtabs(COUNT~DEATH+SEX,data=wildebeest)

strucplot(wildebeest.xtab, shade=T)
```



```
wildebeest.xtab <- xtabs(COUNT~SEX+DEATH,data=wildebeest)

assoc(wildebeest.xtab, shade=T)
```



When you are only dealing with a couple categories, splitting up a mosaic plot into an association plot probably isn't necessary, but if you have an array of three or more rows by three or more columns, it can be helpful.

Correlation

Goodness of Fit :: Continuous Data

Correlation coefficients and **tests of correlation** are used to assess the degree of association between two sets of paired data. In essence, if have a set of *X* (predictor) and paired *Y* (response) data, we can then use correlation to examine the degree to which *X* predicts *Y*.

Correlation Coefficients

A correlation coefficient is a type of effect size. It is presented as a number (from -1 to +1) that measures the degree of association between two sets of numbers. A value of -1 indicates a perfect negative relationship where all variation in *Y* is explained by *X*. A value of +1 indicates a perfect positive relationship where all variation in *Y* is explained by *X*. The most common correlation coefficient is Pearson's *r*. Pearson's *r* assumes *X* and *Y* to have a linear relationship. The other two types of correlation coefficient we will look at are non-linear. These are spearman's rho (ρ) and Kendall's tau (τ).

Linear Correlation: Pearson's *r*

Pearson's correlation is defined as the ratio of the covariance of *X* and *Y* to the geometric mean of the variances of *X* and *Y*. Correlations can range from -1 to 1. The closer the number is to -1 or 1, the stronger the correlation is. The closer the number is to 0, the weaker the correlation is. Parametric correlation assumes that both *X* and *Y* are normally distributed. Non-parametric (non-linear) correlation does not make this assumption. Pearson's correlation coefficient (*r*) is the most commonly used correlation coefficient. It assumes linearity of association and is defined as:

r = covariance of *x* and *y* / square root of the variances of *x* and *y*

ASSUMPTIONS OF CORRELATION (Pearson)

(1) Both variables are continuous
(2) Both variables are approximately normally distributed
(3) There is a linear relationship between the variables
(4) There is homoscedasticity of the data (equal variance)
(5) Observations must be independent (collected randomly)

Note that the total percentage of variance explained (R^2), is literally just Pearson's *r* squared. This is important because the correlation coefficient itself does not represent the percentage of variation explained. If you have $r = 0.45$ then the percentage of variation explained is 0.45^2, which is an R^2 of 0.2025, or 20.25%.

(1) Are the variables continuous? (or at least numeric)

Import the dataset swallow-nestlings-blood.csv (remembering to change your working directory if needed). This is an actual dataset from an honours project.

```
nestlings <- read.table('swallows-nestlings-blood.csv',
header=T,sep=',')
```

Check the data:

```
head(nestlings)
str(nestlings)
```

We are going to work with Haematocrit (Hct) and Haemoglobin in grams per decilitre (Hb.g.dL). Both are numeric and continuous. There may be missing data because this is a real dataset and sometimes birds escape before they are fully measured. A quick **but drastic** way to remove all lines that have missing data uses this code:

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

Note that applying an na.omit to a dataset without checking what it did to the dataset is dangerous. Have a look at the dataset and compare. Did it remove any observations?

```
head(nestlings)
str(nestlings)
```

We're now going to use this dataset to check whether haemoglobin (the oxygen carrying pigment in red blood cells) correlates with haematocrit (packed red blood cell volume as a percentage of total blood volume). It would be astoundingly strange if these two variables didn't correlate to some degree, but let's check how strong the correlation might be.

(2) Testing for violations of normality of the underlying data

Are both variables approximately normally distributed? A Shapiro-Wilks test is a test of normality where the null is that the data follows a normal distribution. If P < 0.05 then the data is not normally distributed.

```
shapiro.test(nestlings$Hct)
shapiro.test(nestlings$Hb.g.dL)
```

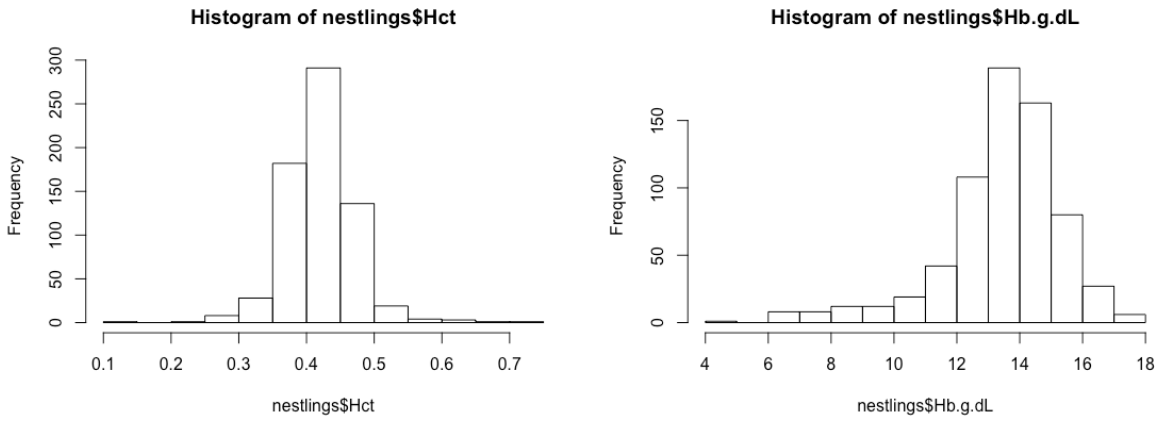
RESULT

```
     Shapiro-Wilk normality test

data:  nestlings$Hct
W = 0.94934, p-value = 1.895e-14


     Shapiro-Wilk normality test

data:  nestlings$Hb.g.dL
W = 0.92072, p-value < 2.2e-16
```

This implies that neither of the two datasets are normally distributed (both P < 0.05). We'd like to compare the results for a Pearson's r (which expects normality and linearity) to the non-parametric Spearman and Kendall correlation values for this dataset, so we will simply proceed. However, if we were planning to publish these results we wouldn't use the Pearson's *r*. It isn't appropriate because the assumption that the datasets are (at least roughly) normally distributed has not been met.

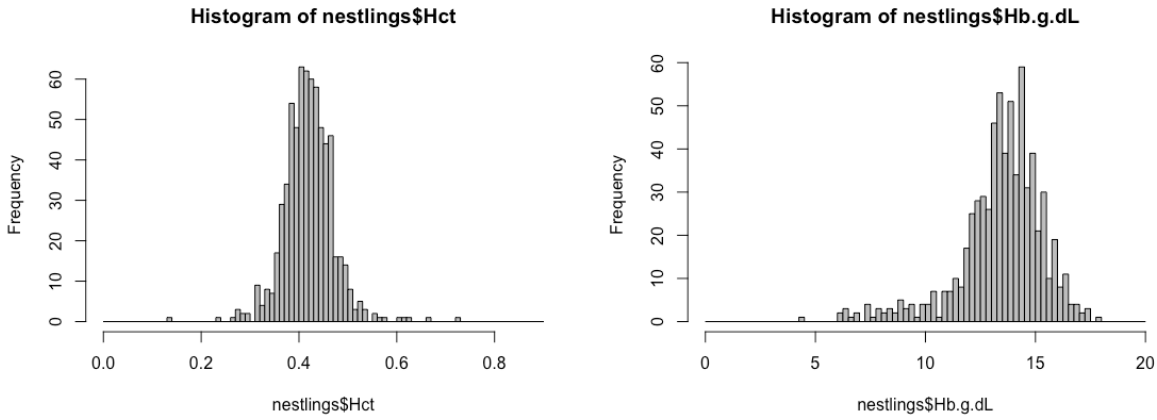
Let's also look at histograms of the data:

```
hist(nestlings$Hct)
hist(nestlings$Hb.g.dL)
```



We can adjust the min, max and break values like so:

```
# plot range from 0.0 to 0.9, at 0.01 increments
hist(nestlings$Hct, breaks=seq(0.0,0.9,0.01), col="grey")

# plot range from 0 to 20, at 0.25 increments
hist(nestlings$Hb.g.dL, breaks=seq(0,20,0.25), col="grey")
```



The plot of haemoglobin (Hb.g.dL) looks especially skewed. The plot of haematocrit is not as skewed, but is much too condensed around the mean to be strictly normal.

(3) Testing for violations of linearity

We need to construct a linear model to test assumptions of linearity. We use the `lm` function to construct a linear model. You can read the tilde (small squiggly line) to mean 'as a function of'. So, this code asks for a linear model (`lm`) of haemoglobin (`Hb.g.dL`) as a function of (~) haematocrit (`Hct`), using the nestlings dataset (`data=nestlings`).

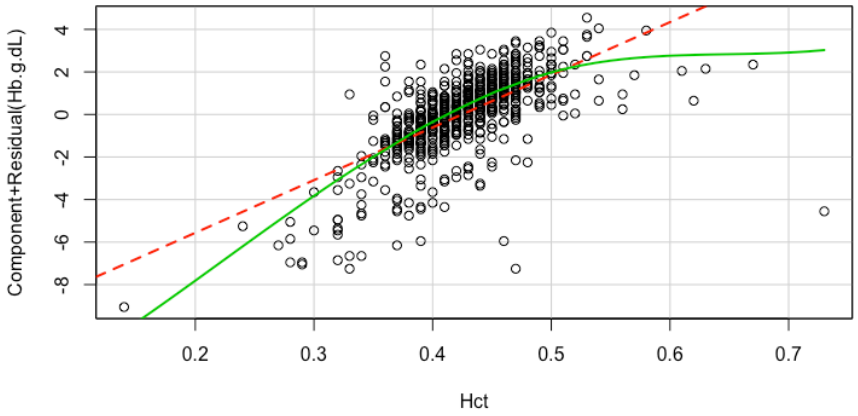
```
swallows.lm <- lm(Hb.g.dL ~ Hct, data=nestlings)
```

Now, load package `car`.

```
install.pckages("car") # if not already installed
library(car)
```

Produce a component residual plot to check linearity of the relationship:

```
crPlots(swallows.lm)
```



If the green line (representing the best non-linear curve of fit) departs from the red line (a straight line of best fit), then the assumption of linearity has probably not been met. We already know that the underlying data is not normal, so it is no surprise to discover that the component residual plot is indicating a violation of linearity.

(4) Testing for violations of homoscedasticity (equal variance)

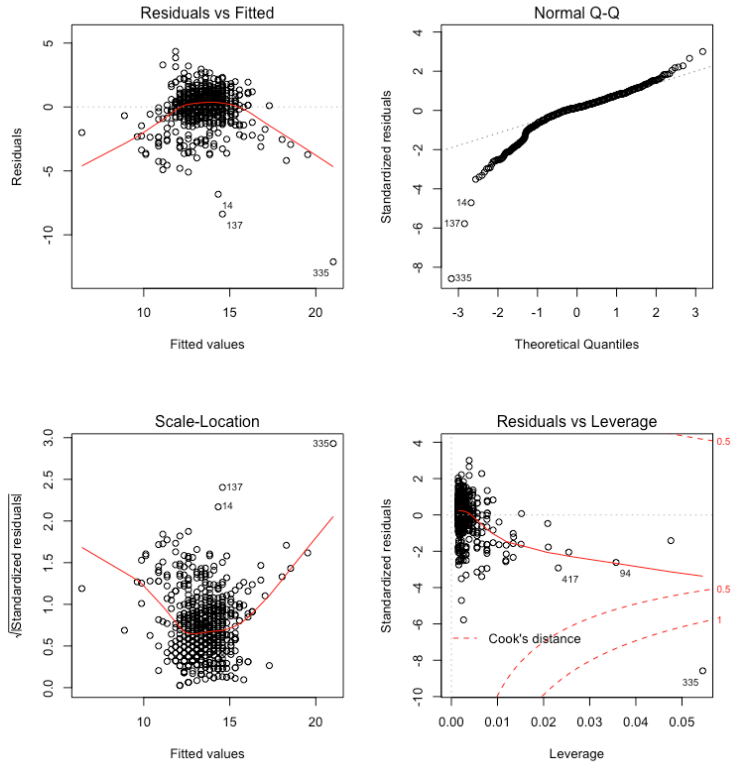
We can also use our linear model to check whether the variances of the residuals are larger at one end of the range of values than the other. We already created a linear model to check for linearity (above), and we can use the same model here.

Set the plotting window to a 2x2 array (i.e. four plots arranged in a square).

```
par(mfrow=c(2,2))
```

Plot four diagnostic plots.

```
plot(swallows.lm)
```



Return the plotting window to default settings.

```
dev.off() # this will also clear your plots!
```

The diagnostic plots check for linearity and equal variance of residuals (**Residuals vs fitted**), normal distribution of residuals (**Normal Q-Q**), linearity and equal variance of residuals (again) (**Scale-Location**), and whether there are any outliers in the residuals (**Residuals vs Leverage**). For now, let's just focus on the left two graphs, **Residuals vs Fitted** and **Scale-Location**. Both of these graphs should ideally show a (relatively) straight, horizontal red line (indicating linearity) and no 'wedge' shape in the data (indicating equal variance). In our case, the relationship is clearly not linear (both red lines are highly u-shaped), but there is no 'triangle', 'arrowhead' or 'wedge' shape to the data (it's just an amorphous cloud, which is what we want). So, at least the assumption of equal variances has probably been met.

(5) Testing for violations of independence

Although there are some statistical approaches to testing independence of observations, the most effective way to avoid problems of independence is good experimental design and clear thinking. Observations are independent if the value of one observation would not (partially or wholly) predict the value of another observation. In our dataset, we might actually have a problem with independence. There are haemoglobin and haematocrit values for multiple nestlings per nest in the study area. Because two or more nestlings would be under the same conditions (i.e. same parents, same food resources in the environment), these values probably do represent pseudo-replication of the results. That is, if we know the Hct of a nestling, then we probably could make a guess as to the likely range for its siblings in the same nest. If we were planning to publish this data we would have to either average values to the nest, or use a linear mixed effects model with **NEST.ID** as a random effect.

For our purposes, we will simply proceed, but keep in mind that we really are simply examining the dataset for teaching purposes now. It has violated almost all of our assumptions, and the results of a Pearson's *r* linear correlation applied to it will not be reliable, and probably not even meaningful.

Plot the data for haematocrit and haemoglobin:

```
plot(nestlings$Hct, nestlings$Hb.g.dL)
```

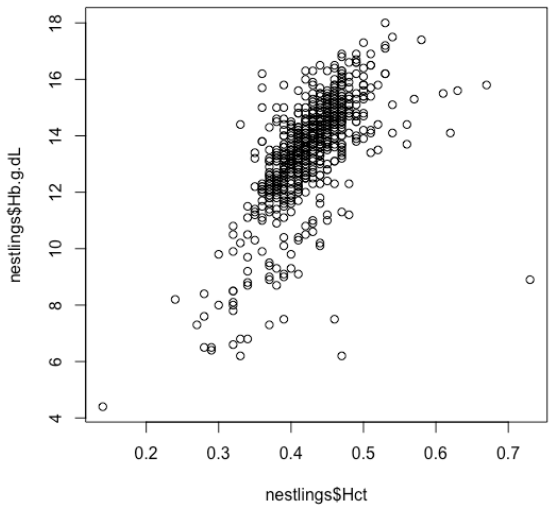


fig 1. Haematocrit (Hct) and haemoglobin (Hb) (g/dL) in 12 day old nestling welcome swallows.

The default R plot isn't very attractive. We can adjust colours and point characters (**pch**), as well as add a line of best fit using the linear model we created earlier. The scatterplot will appear first, and then the line of best fit will be added over the top using the **abline** function.

```
plot(nestlings$Hct, nestlings$Hb.g.dL, pch = 20, col =
"grey50", xlab="Haematocrit (%)", ylab="Haemoglobin (g/dL)")

abline(swallows.lm, col = "red", lwd=2)
```

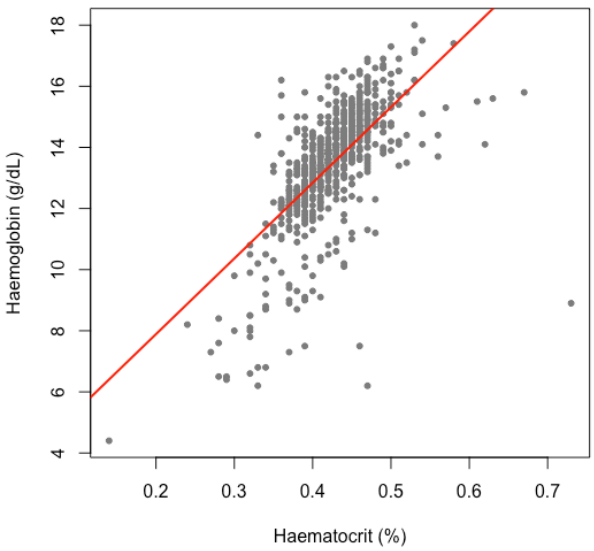


fig 1. Haematocrit (Hct) and haemoglobin (Hb) (g/dL) in 12 day old nestling welcome swallows. Line in red is a line of best fit assuming linearity of the relationship.

```
pch = 20                          # point shape… try other numbers
col = "grey50"                    # Colour… try other colours
xlab = "Haematocrit (%)"          # Label for the x-axis
ylab = "Haemoglobin (g/dL)"       # Label for the y-axis
lwd = 2                           # Line width… try other numbers
```

For example:

```
plot(nestlings$Hct, nestlings$Hb.g.dL, pch = 17, col =
"hotpink2", xlab="Haematocrit (%)", ylab="Haemoglobin (g/dL)")

abline(swallows.lm, col = "seagreen4", lwd=5)
```

Calculating a Pearson's *r*

Pearson's correlation coefficient (*r*) is the most commonly used correlation coefficient. It assumes linearity of association and is defined as:

r = covariance of *x* and *y* / square root of the variances of *x* and *y*

We will attach the dataset to make life easier. Attaching a dataset means that R will always be looking at the dataset and you don't need to keep stating that we are working with the nestlings dataset.

```
attach(nestlings)
```

Calculate variances and covariance:

```
var(Hct)
var(Hb.g.dL)
var(Hct,Hb.g.dL)
```

Now we will calculate a Pearson's *r* correlation coefficient:

```
var(Hct,Hb.g.dL)/sqrt(var(Hct)*var(Hb.g.dL))
```

Now we can calculate the Pearson's correlation coefficient using R.

```
cor(Hct, Hb.g.dL, method = "pearson")
```

RESULT

```
> var(Hct,Hb.g.dL)/sqrt(var(Hct)*var(Hb.g.dL))
[1] 0.6571337

> cor(Hct, Hb.g.dL, method = "pearson")
[1] 0.6571337
```

A correlation test using Pearson's *r*

Correlation tests are not used very often in biological sciences, although especially for non-linear associations (Spearman and Kendall's), these tests should perhaps be used more often. Correlation tests are so infrequently used in biology that if you do use a correlation test, you may have to reference it. Otherwise a reviewer may not even know what it is.

```
attach(nestlings)
cor.test(Hct, Hb.g.dL, method = "pearson")
```

RESULT

```
> cor.test(Hct, Hb.g.dL, method = "pearson")

     Pearson's product-moment correlation

data:  Hct and Hb.g.dL
t = 22.616, df = 673, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6120200 0.6979844
sample estimates:
      cor
0.6571337
```

The P value for a Pearson's correlation test should be exactly the same as for a simple regression analysis with one predictor and one response (which are more frequently used in biology). The test above gives a test statistic (**t**), degrees of freedom (**df**) and P value (**p-value**). There is also a 95% confidence interval for the range of the Pearson's r value (in this case we have 95% confidence that it is between 0.610 to 0.700). The estimated Pearson's r is also provided. If you reported this in a results paragraph it might read like this:

There was a significant positive relationship ($r = 0.657$) between haemotocrit (%) and haemoglobin (g/dL) in 12 day old house swallows (correlation test: $t = 22.6$, $df = 673$, $P < 0.001$).

Remember that a P value of $2.2e^{-16}$ is actually 0.00000000000000022. There is no point in reporting P values below 0.001, so we just state this was significant at < 0.001.

Referencing the test? How do we find the appropriate reference? Luckily for us, the help function in R will usually have academic references listed. Use this command:
`?cor.test`

Non-linear correlation coefficients

Spearman and Kendall correlation coefficients are both forms of ranking coefficients that allow us to obtain correlation coefficients for non-linear and non-normally distributed relationships.

ASSUMPTIONS OF CORRELATION (Spearman & Kendall)

(1) Both variables are at least ordinal (counts & continuous are ok)

(2) Scores of variable *X* must be montonically related to variable *Y*

(3) Kendall is preferred to Spearman (better statistical properties)

(4) Observations must be independent (collected randomly)

---

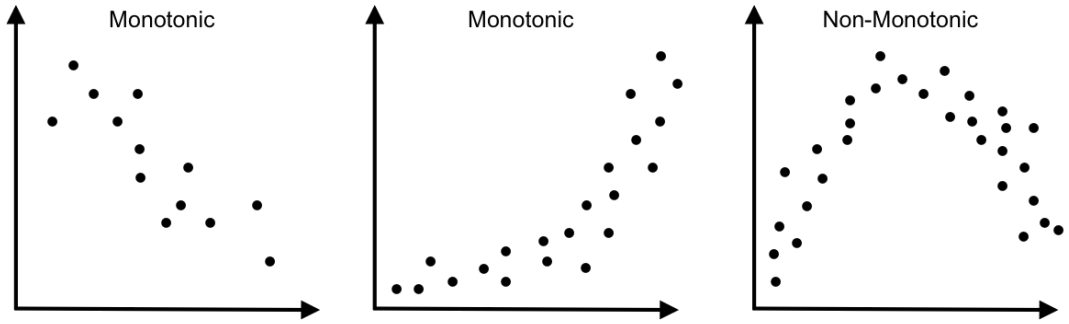
(1) Testing for violations of ordinal numbers

What does 'at least ordinal' mean? Many non-parametric tests (that is tests that make no assumptions about distributions of data), do require data to be 'at least ordinal'. In an ordinal series of data numbers don't necessarily need to be on the same scale but a higher number does need to represent a higher value. So, for example, if we have five forest fragment of a = 5 ha, b = 6.1 ha, c = 10 ha, d = 12.2 ha and e = 300 ha we could convert this to an ordinal series of a = 1, b = 2, c = 3, d = 4 and e = 5. Obviously the information in the original measurement has been lost, but non-parametric tests often don't care about relative differences anyway. They usually are only interested in which of two values is higher. Before conversion to a scale, the difference between fragments e and d was a 25x difference, and the difference between c and d was a 1.2x difference. Reframing these values as an ordinal scale removes the scale differences between variables but retains their ordinal positions. That is, d is still smaller than e but larger than a, b and c.

Anyway, for your purposes, you are extremely unlikely to ever have datasets that are not ordinal. You would need to have a quite strange set of data where 8 might be higher than 5 in one instance but lower than 5 in another instance.

(2) Testing for violations of a monotonic relationship

What does monotonic mean? A monotonic relationship does not need to be linear but it does need to be the case that as variableX increases, variable *Y* also increases or decreases **in a consistent way**. Distributions that are u, or n shaped are not monotonic.



(3) Kendall is preferred to Spearman

Really? Why are we bothering looking at Spearman then? Although arguably Kendall's tau is simply mathematically preferable to Spearman's rho, some statisticians will advise that for small datasets (<50) it is better to use Spearman's, whereas if a dataset is 50 or larger, we would use Kendall's. This may be a matter that your supervisor (or an editor or reviewer) may have an opinion on, so both methods are presented. Realistically, however, differences between Spearman and Kendall correlation coefficients tend to be reasonably small. If you do find that the difference between the coefficients is large (more than about 0.1), then look at the size of the dataset, and opt for Spearman for smaller (n < 50), and Kendall for larger (n > 50) datasets.

(4) Violations of independence

This again? Yes. Independence of observations is an assumption of all statistical tests. It is best to be sure your experimental designs are well thought out.

Pearson's, Spearman's and Kendall's correlations

Let's obtain all three correlation coefficients and compare them.

```
attach(nestlings)
```

When you attach data for the second (third, fourth etc) time, R will save a bit of memory space by 'masking' columns from previous attaches. The long screed of red 'error' message isn't an error at all. It's just telling you that some columns are being masked from an earlier attach.

```
cor(Hct, Hb.g.dL, method = "pearson") # parametric
cor(Hct, Hb.g.dL, method = "spearman") # non-parametric
cor(Hct, Hb.g.dL, method = "kendall") # non-parametric

detach(nestlings) # to keep our working space clean
```

RESULT

```
> cor(Hct, Hb.g.dL, method = "pearson") # parametric
[1] 0.6571337

> cor(Hct, Hb.g.dL, method = "spearman") # non-parametric
[1] 0.7048725

> cor(Hct, Hb.g.dL, method = "kendall") # non-parametric
[1] 0.5542297
```

We already know that this dataset isn't highly suitable for the linear (Pearson) correlation coefficient. But the coefficient for Spearman and Kendall are quite different (over 0.1 apart). Which should we choose? As this is a large set of observations (>50), the Kendall is preferable. This leaves us with the Kendall's tau of 0.554.

Spearman's and Kendall's correlation tests

This is similar to the code we used for the Pearson's correlation test, but with the method replaced with `"spearman"` or `"kendall"`.

```
attach(nestlings)

cor.test(Hct, Hb.g.dL, method = "spearman")
cor.test(Hct, Hb.g.dL, method = "kendall")
```

RESULT

```
> cor.test(Hct, Hb.g.dL, method = "spearman")

     Spearman's rank correlation rho

data:  Hct and Hb.g.dL
S = 15128000, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.7048725

Warning message:
In cor.test.default(Hct, Hb.g.dL, method = "spearman") :
  Cannot compute exact p-value with ties

> cor.test(Hct, Hb.g.dL, method = "kendall")

     Kendall's rank correlation tau

data:  Hct and Hb.g.dL
z = 20.712, p-value < 2.2e-16
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.5542297
```

The way Spearman's rho works is by ranking the observations and looking for a trend in the ranks. This means that this coefficient will encounter (slight) problems if there are ties in the ranks. However, unless your P value is close to the marginal 0.04-0.05 range you probably don't need to worry too much about this error message.

In this instance, though, we've already established that we prefer the Kendall test anyway, so the ties are fine (although, be warned that Kendall's tau will also produce errors for ties when $n < 50$).

How would you present this in a results section? It's preferable to use the tau (τ) or rho (ρ) symbols rather than the words spelled out, but otherwise, it is similar to how we presented the Pearson results above. In this case:

There was a significant positive relationship (τ = 0.554) between haemotocrit (%) and haemoglobin (g/dL) in 12 day old house swallows (Kendall's correlation test: z = 20.7, n = 675, P < 0.001).

We included an *n* because Kendall's correlation test doesn't use degrees of freedom. We can obtain this by checking the length of one of the columns (any column will do).

```
length(nestlings$Hct)
```

RESULT

```
> length(nestlings$Hct)
[1] 675
```

Correlations: strong, weak, positive and negative

We talk about correlations being 'strong' or 'weak' and 'positive' and 'negative'. The dataset corr_data has been set up to demonstrate what these terms mean:

```
corr.data <- read.table('corr_data.csv',header=T,sep=',')

par(mfrow=c(3,2)) # set the plotting window to 3x2 plots

plot(Y1~X1, data = corr.data, ylim = c(0,30))
abline(lm(Y1~X1, data = corr.data), col = "red")
cor(corr.data $X1, corr.data $Y1, method = "pearson")
mtext("High r, positive slope: r = 0.996", side = 3, adj = 0)

plot(Y2~X2, data = corr.data, ylim = c(0,80))
abline(lm(Y2~X2, data = corr.data), col = "red")
cor(corr.data $X2, corr.data $Y2, method = "pearson")
mtext("Low r, positive slope: r = 0.331", side = 3, adj = 0)

plot(Y3~X3, data = corr.data, ylim = c(0,30))
abline(lm(Y3~X3, data = corr.data), col = "red")
cor(corr.data $X3, corr.data $Y3, method = "pearson")
mtext("High r, positive slope: r = 0.785", side = 3, adj = 0)

plot(Y4~X4, data = corr.data, ylim = c(0,160))
abline(lm(Y4~X4, data = corr.data), col = "red")
cor(corr.data $X4, corr.data $Y4, method = "pearson")
mtext("High r, negative slope: r = -0.999", side = 3, adj = 0)

plot(Y5~X5, data = corr.data, ylim = c(0,400))
abline(lm(Y5~X5, data= corr.data), col = "red")
cor(corr.data $X5, corr.data $Y5, method = "pearson")
mtext("Moderate r, negative slope: r = -0.666", side = 3, adj
= 0)

plot(Y6~X6, data = corr.data, ylim = c(0,160))
abline(lm(Y6~X6, data = corr.data), col = "red")
cor(corr.data $X6, corr.data $Y6, method = "pearson")
mtext("Low r, negative slope: r = -0.104", side = 3, adj = 0)
```

The terminology can be a bit confusing because when someone talks about a 'strong' correlation, usually what is meant is that the correlation coefficient (*r*) is high. A strong positive correlation would be one with a very tight relationship (the data points all cluster around the line of best fit) but the slope might not be very steep so that the biological effect might not be especially strong.

```
dev.off() # return plotting window to defaults
          # note that this will 'blank out' your figures
          # only run dev.off to clear your plotting screen
```

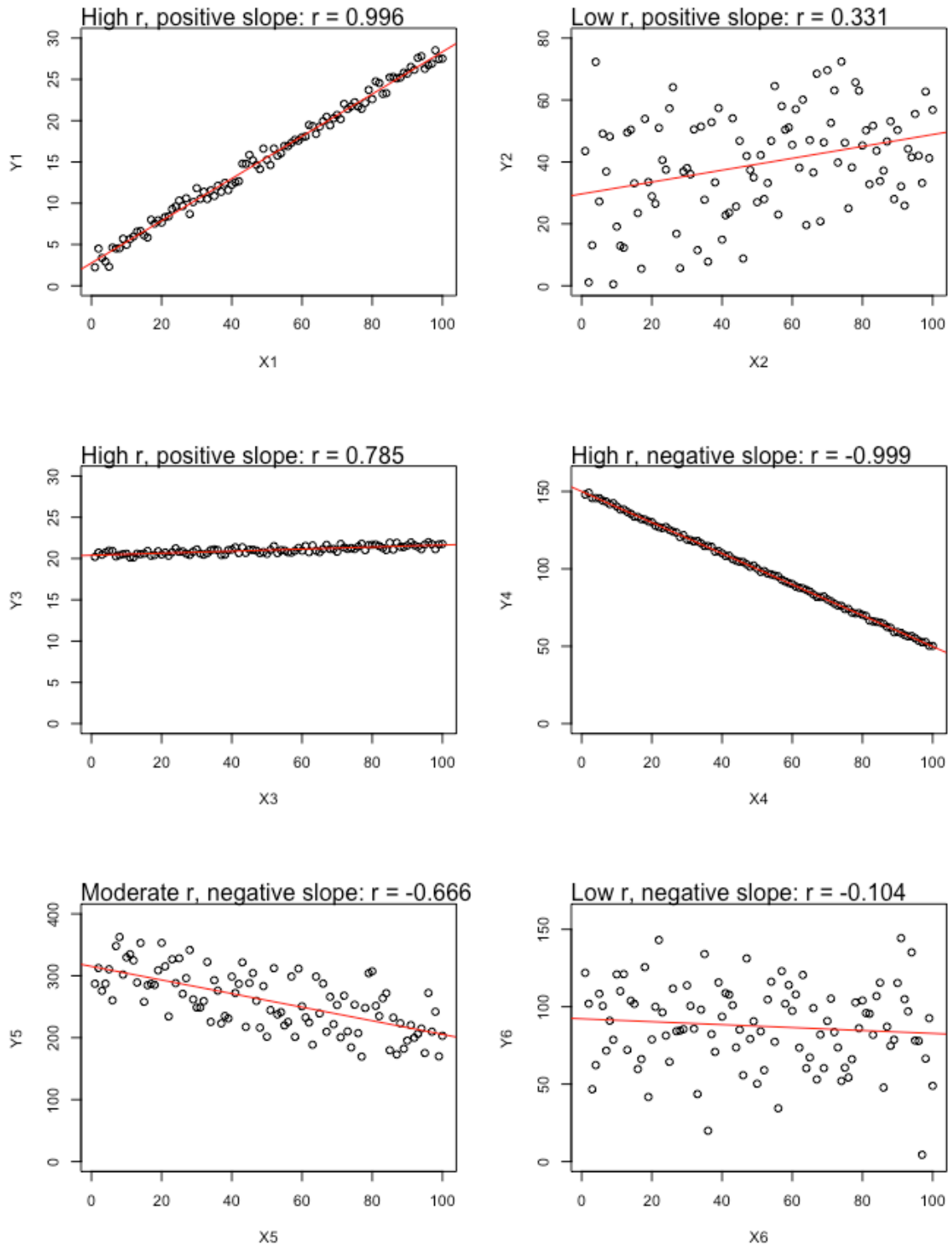


fig 2. Scatterplots demonstrating strong and weak correlations. Note that a 'strong' correlation only indicates that the points fall close to a line of best fit. A strong correlation could have quite a weak slope.

Variability Tests

Regression Analysis :: t-tests :: ANOVAs etc

Tests that use variability as a measure of 'noise' in the data are among the most commonly used and reported types of statistical tests. These include regression analysis, *t*-tests and Analyses of Variance (ANOVAs). These tests are often treated as distinct tests, but actually they all fundamentally act in the same way. All of them use variance as a measure of uncertainty in the data to decide whether a pattern might have occurred by chance. A Student's *t*-test will give you exactly the same P value as an ANOVA using the same data. Regression analyses and ANOVAs (and related tests) are all just forms of linear model, and you'll find they also give the same P values for the same datasets.

A quick note on terminology: *Just a warning. I am using 'variability family of tests' here to mean tests that are based on a signal:noise ratio, where the 'noise' is some measure of variance. There are also 'variance tests' or 'tests of variance', that are used to test* ***for differences in variance****, such as a Bartlett test or a Fligner-Kileen test. Such tests can be used to test hypotheses of differing variances, but are more often used for testing the assumption of equal variance. I've added this note just to try and head off confusion about what I mean exactly by 'the variability family of tests' (i.e. we're not talking about Bartlett tests, or similar, here).*

Table 1. Types of Predictor, Covariate and Response variables and the appropriate test for each. A covariate is usually not the predictor of interest but is included to control for some other variable that is likely to be important. Covariates are typically numeric, and often continuous. ANOVAs and ANCOVAs are both forms of general linear model (LM). Because of computing power today, highly complicated GLMs can be generated using many Predictors that are both discrete and continuous.

Predictor	Levels	Covariate	Response	Test
Continuous	NA	Numeric	Continuous	Regression
Discrete	2	No	Continuous	*t*-test
Discrete	2+	No	Continuous	ANOVA
Discrete	2+	Numeric	Continuous	ANCOVA

What is a covariate?

A covariate is a numeric (often continuous) variable included in a general linear model that helps explain variation, but is usually not the variable of interest. Here's an example of how a covariate can be helpful. Let's imagine you measured stress hormones in people living in Calcutta and in the Indian countryside with the hypothesis that living in the city would be more stressful (predictor is discrete with two levels: city or country). Possibly, you might not find a relationship unless you took into account a third variable, the time a person has been living in a city. People who have just moved to a city might find it stressful, whereas people who were born in the city and grew up there might not find it stressful at all. In this case, the covariate helps explain the dependant variable (stress hormones in the blood) and it might allow us to see whether there is an effect of city living on stress, albeit one that is time dependent.

What is a discrete variable?

Discrete variables is an umbrella term for any variable that exists as a set of discrete parts. For t-tests and ANOVAs, the discrete variable we would most typically be working with would be nominal (also called categorical), where the predictor is entered into a spreadsheet as a set of words: **Low**, **Medium**, **High**, or, **Male**, **Female**, or **Present**, **Absent**. In R these should appear as the 'factor' data type, although they may import as 'characters' instead which can cause problems. After importing a dataset that has nominal data it is sensible to make sure all the nominal columns are actually factors by looking at the structure of the data. If something is not a factor, you can tell R that it should be a factor, as below.

Check the structure of the dataset:

```
str(yourdata)
```

Change a variable to a 'factor' in R:

```
yourdata$your_variable <- as.factor(yourdata$your_variable)
```

Change a variable to a 'number' in R:

```
yourdata$your_variable <- as.numeric(yourdata$your_variable)
```

Note that the second line of code will only work if the variable could actually be a number. If a column of data contains letters, words or symbols, R will not be able to turn the data into numbers.

How is a *t* test comparison made?

The key point of the *t*-distribution is that it changes shape with different degrees of freedom. By comparing two *t*-distributions we can calculate the difference between the means and divide this by the noise (variance) of the distributions. Classically, especially before computers, t-tests were calculated using the assumption that both populations had the same variance (an equal variance **Student's t-test**), as that makes the equation much easier to work out. Now, t-tests are more sophisticated and the degrees of freedom are typically adjusted to account for differences in variance between the two samples (an unequal variance **Welch's t-test**).

signal / noise t = difference of group means / variability of groups

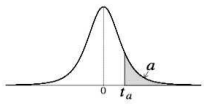
$t = X_{mean} - Y_{mean} / SE(X - Y)$

The *t*-test signal to noise ratio is called a *t*-statistic. Whether it is positive or negative isn't important because that simply depends on which mean happened to be larger. In the past, you would take the non-signed *t*-statistic, decide on a significance level (usually, alpha = 0.05) and look up significance on a table. Current statistical programs can provide exact *P*-values for a test instead of estimates from a table.

So, in essence, what the test is doing is **1)** working out a signal to noise ratio (the *t*-value = how strong is the pattern here?); **2)** establishing what the distribution for possible *t*-values should look like based on your sample size (higher *df* = thinner tails, which implies greater confidence about where our *t*-value falls); **3)** checking our actual *t*-value against the distribution of possible t-values. If there is a 5% or less chance of getting out t-value by chance (given the null were true), then we take that to be a significant result.

Critical Values of the t Distribution

A table entry is the value of t_a, having an area to the right of a under a t distribution with df degrees of freedom.



df	$t_{0.20}$	$t_{0.15}$	$t_{0.10}$	$t_{0.05}$	$t_{0.025}$	$t_{0.01}$	$t_{0.005}$	$t_{0.001}$	$t_{.0005}$
1	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.3	636.6
2	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.33	31.60
3	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.21	12.92
4	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781

Figure 1. An example of a table for *t*-test results with different critical levels for significance at the 0.20, 0.15, 0.10, 0.05, 0.01, 0.005, 0.001 and 0.0005 levels (reading across the columns) and degrees of freedom from 1 to 9 (reading down the rows). If a *t*-value equals or is larger than the critical value for a given *df* and significance level, then the result would be significant at that level. Typically, the critical value α = 0.05 would be used, which corresponds to the column under $t_{0.05}$.

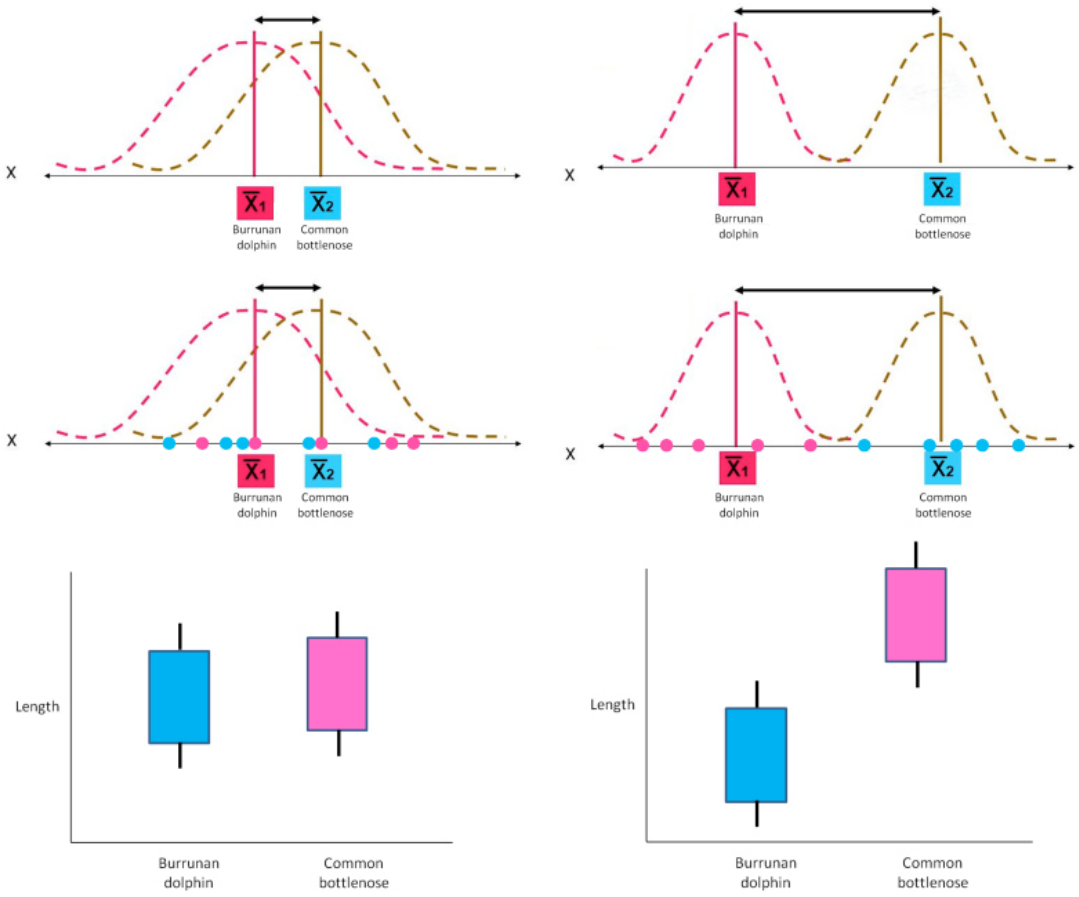


Figure 2. Two populations of dolphin measured for morphometric traits. On the left, difference between the means of groups is small and the variability (noise) is large. Probably, we would find the difference is non-significant ($P > 0.05$) if we ran a *t*-test. On the right the difference between means of groups is large and in comparison the variability (noise) is relatively small. Probably, we would find the difference is significant ($P < 0.05$) if we applied a *t*-test.

Assumptions of variability tests

Because t-tests, regression analyses and ANOVAs all function in the same way, they are all concerned with assumptions of normality and equal variance. Although you can get away with slight departures from normality (these tests are actually relatively robust to small problems with normality), you will very much run into trouble if you have increasing variances with higher means. So, what do you do, if assumptions are not met?

Usually, the first step is to try transforming the response variable. Any mathematical operation that manipulates the data as a whole so that it retains its overall order and sequence but the distribution is changed (usually into a normal distribution). The object of transformation is usually to change non-normal or non-linear data into normal or linear data. A log transformation for example, logs all the response values. A square root transformation takes the square root of all the response values. Sometimes we transform predictors as well, but that is unusual and probably if you think you need to transform predictors actually you need to rethink if you are using the correct statistical analysis.

Once you apply transformations, new columns will be created and these new columns become your response variables. A transformation changes the 'shape' of the data but not the 'order' of the observations. So, if we had a series of observations that looked like this...

1
2
5
14
106
2205

We might decide they need to be transformed to be more easily analysed. They might be transformed like this, using a log transformation:

Original value	Log transformed value
1	0.00
2	0.30
5	0.70
14	1.15
106	2.03
2205	3.34

Notice how the numbers have been changed so that they are closer together, but the order hasn't changed. The largest value is still the largest. The smallest value is still the smallest.

Transformations

Researchers talk about 'applying a transform' or 'applying a transformation', 'transforming the data' or 'working with transformed data'. These all mean the same thing: the data has been changed mathematically to make it more normal and/or more linear.

What do we need to know at a basic level?

- Typically, only the RESPONSE (dependent variable) is transformed
- Transformation involves loss of information
- Better not to do if you can avoid it (bit of a trade-off)
- Never graph transformed data in a final report (ok to do this in an exploratory analysis for your own benefit) (even if you use transformed data in your tests)
- Sometimes transformation is unavoidable

Code for Transformations

To make life easier we can set up a dummy variable called **x** and drop a **RESPONSE** into it like so:

```
x <- yourdata$RESPONSE
```

Now apply various transformations

```
yourdata$INVR <- 1/x                # Reciprocal transformation
yourdata$NEGINV <- -1/x             # Negative reciprocal transformation
yourdata$SQUARED <- x^2             # Power transformation
yourdata$CUBERT <- x^(1/3)          # Cube root transformation
yourdata$SQRT <- sqrt(x)            # Square root transformation
yourdata$LOG <- log(x)              # Log base e. Cannot be applied to zero
yourdata$LOG10 <- log10(x)          # Log base 10 Cannot be applied to zero
yourdata$ASQRT <- asin(sqrt(x))     # Arcsine square root transformation
                                    # Only useful for percentages (0.01-0.99)
yourdata$LOGIT <- log(x/(1-x))      # Logit transformation
                                    # Only useful for percentages (0.01-0.99)
yourdata$FOLD1 <- sqrt(x)/sqrt(1-x) # Square root folded transformation
                                    # Only useful for percentages (0.01-0.99)
yourdata$FOLD2 <- log(x)/log(1-x)   # Log folded transformation
                                    # Only useful for percentages (0.01-0.99)
yourdata$FT <- sqrt(x) + sqrt(x+1)  # Freeman-Tukey transformation
```

Now look at your dataset:

```
head(yourdata)
View(yourdata)
str(youdata)
```

Note how the transformations will have been added as columns to the end of your dataset? These are your potential response variables now. You need to check if they have met assumptions (by checking boxplots and residuals of models etc). Generally speaking you'll want to pick the transformation that best meets assumptions, although sometimes you might decide to pick a simpler transformation over a more complex transformation if the difference between them isn't substantial. Picking the best transformation is a bit of an art rather than something that has hard and fast rules. Often a lot of boxplots and histograms will be needed to work out which transformation is preferable.

Advanced Transformations

If all of the above transformations fail to meet the assumptions of the data you can try two advanced transformations:

BOXCOX TRANSFORMATION

```
install.packages("MASS") # install library from web
library(MASS) # load library

fit <-lm(RESPONSE~TREATMENT,data=yourdata)
bc <- boxcox(fit)
lambda<-with(bc, x[which.max(y)])
yourdata$bc <- ((x^lambda)-1)/lambda)
boxplot(bc ~ TREATMENT, data = yourdata)
```

RANK NORMAL TRANSFORMATION

```
install.packages("GenABEL") # install library from web
library(GenABEL) # load library

yourdata$RANK.NORMAL <- rntransform(yourdata$RESPONSE)
```

Are there any types of data that are always transformed?

- Percentages and ratios often need to be transformed
- Arcsine square root transformations are often used for percentages
- Log transformations are often used for ratios
- Square root transformations are also used for ratios, especially if some values are zero.

What if no transformation seems to help?

If none of the transformed values seem to fit assumptions of a test, then possibly you are using the wrong test. If you have a dataset where the response is a count and there are a lot of zeroes, then no amount of transformation will make it fit ANOVA assumptions and you need to start thinking about other tests such as GLMs.

t-tests

We usually think of a t-test as comparing two samples comprised randomly selected individuals from a population. However, **one-sample t-tests** and **paired t-tests** also exist. We will start with a one sample t-test.

One sample t-test

A one-sample t-test provides confidence intervals for a single set of data points. The test assumes the comparison dataset is normally disturbed and that the observations are independent. It can be used to test whether a single data point falls within a range of expected values.

ASSUMPTIONS OF ONE SAMPLE T-TESTS

(1) The comparison datasets is normal

(2) Observations must be independent (collected randomly)

Import the adult house swallows dataset:

```
swallows <- read.table('swallows-adults.csv',
header=T,sep=',')
```

Check the data:

```
head(swallows)
str(swallows)
```

Run a one sample t-test on Wing Lengths of the adult swallows:

```
t.test(swallows$WingL)
```

RESULT

```
     One Sample t-test

data:  swallows$WingL
t = 475.39, df = 110, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 112.9416 113.8872
sample estimates:
mean of x
 113.4144
```

So we appear to have a significant result. But significant by comparison to what? Exactly what is this test reporting?

```
        One Sample t-test

data:  swallows$WingL
```
This is simply a reminder what data we used for the t-test.

```
t = 475.39, df = 110, p-value < 2.2e-16
```
The t-value (signal to noise ratio), degrees of freedom, and p-value. Although all three of these values would be reported, typically, only the P value would be discussed.

```
alternative hypothesis: true mean is not equal to 0
```
This is telling us that the test is comparing the wing lengths to a value of zero. That is, it is asking if the true mean of the dataset is zero. This is not especially useful or interesting. We'll look at how to modify this below.

```
95 percent confidence interval:
 112.9416 113.8872
```
The 95% confidence interval for mean of wing lengths of swallows. We have a 95% confidence that the mean is no lower than 112.9 mm and no higher than 113.9 mm.

```
sample estimates:
mean of x
 113.4144
```
The estimated mean of the swallow wing lengths in mm.

How might we use a one-sample t-test. The test is useful if you only have a single value and you want to compare it to a sample. Imagine you caught a purple swallow. It looks like your blue welcome swallows except for feather colour. You are curious as to whether it is a vagrant species, or maybe a mutation, so you take measurements. Perhaps we want to compare the wing length of the purple swallow to the sample of blue swallows. We can use **mu** in the t-test code to change the value for comparison. Assume you obtained a wing length of 113.1 mm for the purple swallow.

```
t.test(swallows$WingL, mu = 113.1)
```

Try running the above test and determine whether the purple swallow had a significantly different wing length to the estimated mean wing length of welcome swallows. As well as looking at the P value, have a look at the confidence interval for the welcome sallow wing lengths. Did the confidence interval change from what we obtained above? Why or why not?

Two sample t-test (unpaired)

A two-sample t-test allows you to compare the means of two samples and determine which (if either) is higher. We will use this data to check masses of swallows by sex.

ASSUMPTIONS OF STUDENTS T-TEST

(1) Observations must be independent (collected randomly)

(2) Both datasets are normal

(3) Equal variances

The presence or absence of a broodpatch on a swallow indicates if it is a female or male. Females brood but males do not (or only seldom), so females acquire a discoloured brooding patch whereas males tend not to. Using the adult swallows dataset we want to investigate whether male or female adult welcome swallows are heavier on average. We will use a two-sample t-test to test this.

First, recheck your data. The **BROODPATCH** value needs to be factor. Use `str(swallows)` to check it.

The researcher has recorded the data as a set of binary numbers (1 = no and 2 = yes) but R doesn't know that these are supposed to be two levels of a single factor. We can change this using a line of code.

```
swallows$BROODPATCH <- as.factor(swallows$BROODPATCH)
```

Now rerun the `str(swallows)` command and check that **BROODPATCH** has changed to a factor with 2 levels. The next thing to do is check the assumptions. Now we will look at assumptions.

(1) Datasets must be independent (collected randomly)

There are no tests for independence that are especially useful. Really, this is a matter of good experimental design and understanding what problems are associated with pseduoreplication.

(2) Are the datasets normal?

Check boxplots of the data. If the boxplots are (reasonably) symmetrical they are probably at least bell-shaped, although may not be strictly normal.

```
boxplot(MASS~BROODPATCH,data=swallows)
```

There are a number of different ways to look at each level within the factor separately (i.e. look at birds with brood-patches and without brood-patches) but the most straightforward is simply to subset the data. We'll attach the dataset to make the code easier (when a dataset is attached R assumes you are working with that dataset unless told otherwise... this reduces the need to keep typing the name of the dataset over and over).

```
attach(swallows)
brood.yes <- subset(swallows, BROODPATCH=='2')
brood.no <- subset(swallows, BROODPATCH=='1')
```

Check histograms of the data.

```
hist(brood.yes$MASS)
hist(brood.no$MASS)
```

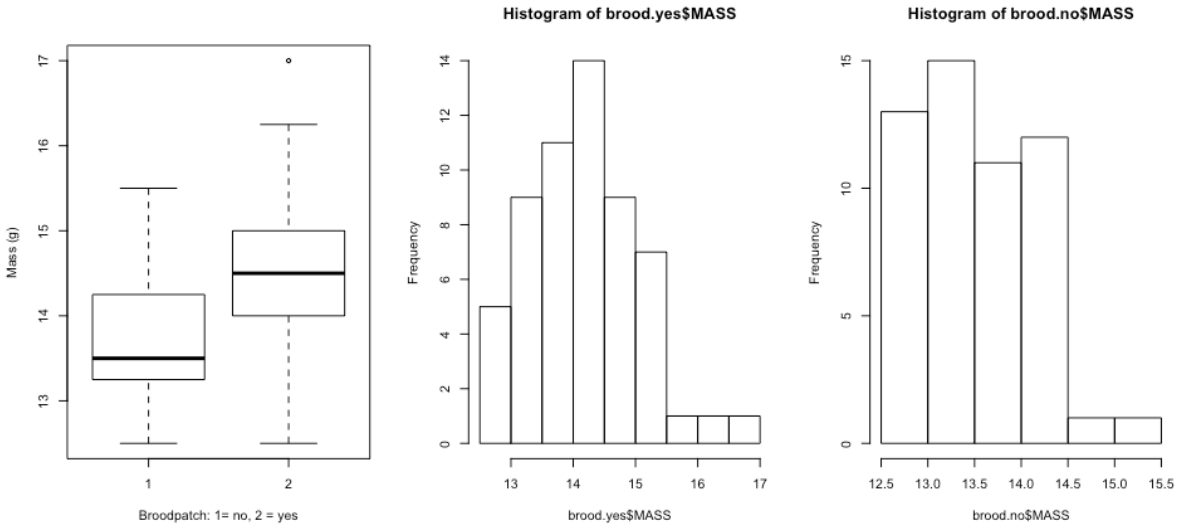
You can also use a Shapiro-Wilk's test of normality, where we would take $P < 0.05$ to indicate that the assumption of normality may have been broken. However, distribution tests, such as Shapiro-Wilks, tend to be quite aggressive and they often diagnose distribution problems when the data is probably fine for a t-test or similar. Visual assessments of boxplots and histograms is usually preferable.

```
shapiro.test(brood.yes$MASS)
shapiro.test (brood.no$MASS)
```

```
detach(swallows)
```

It is usually good practise to detach your dataset after working with it.

Let's look at our plots. The boxplots don't look too bad, but the histograms are potentially not normal.



Keeping in mind that Shaprio-Wilks tests can tend to indicate that most biological data is non-normally distributed , let's check these distributions and see what the results are. Where P<0.05, we would take this to be evidence that the distribution may not be normal.

RESULT

```
> brood.yes <- subset(swallows, BROODPATCH=='2')
> shapiro.test(brood.yes$MASS)

     Shapiro-Wilk normality test

data:  brood.yes$MASS
W = 0.96923, p-value = 0.1472


>
> brood.no <- subset(swallows, BROODPATCH=='1')
> shapiro.test(brood.no$MASS)

     Shapiro-Wilk normality test

data:  brood.no$MASS
W = 0.94083, p-value = 0.01106
```

The distribution of masses for swallows without a brood-patch is a bit suspect. At this point I would consider using a **non-parametric equivalent of a t-test** (Mann-Whitney U: **`wilcox.test`** in **R**). However, we are going to use this data for non-parametric tests later, and it would be good to compare the results to the t-test results. Let's proceed to checking equal variances.

(3) Equal variances?

Check boxplots of the data. If the boxplots are (reasonably) symmetrical and (about) the same height, then the variances are probably equal. You can also try using a test of equal variances. The Bartlett test is suitable for this purpose.

```
boxplot(MASS~BROODPATCH,data=swallows)
bartlett.test(MASS~BROODPATCH,data=swallows)
```

We already have the boxplot (above), so let's look at the result of the Bartlett Test.

RESULT

```
> bartlett.test(MASS~BROODPATCH,data=swallows)

     Bartlett test of homogeneity of variances

data:  MASS by BROODPATCH
Bartlett's K-squared = 5.0971, df = 1, p-value = 0.02397
```

The assumption of variances doesn't seem to be met here. A significant P value ($P < 0.05$) is usually taken to indicate that the variances may not be equal. However, rather than resort to a non-parametric test (or transformation), we can use a different sort of t-test, the **Welches unequal variance t-test**, which is actually the default in R.

ASSUMPTIONS OF WELCH'S T-TEST

(1) Observations must be independent (collected randomly)
(2) Both datasets are normal

The Welch's t-test penalised the degrees of freedom for unequal variances. Also, as it turns out, if the variances are perfectly equal a Welch's t-test collapses into a Student's t-test anyway. This means that you might as well just always use the Welch's t-test. We'll look at how to run both though.

Run a classic equal variances (Student) *t*-test.

```
t.test(MASS~BROODPATCH,data=swallows, var.equal=TRUE)
```

RESULT

```
     Two Sample t-test

data:  MASS by BROODPATCH
t = -4.6563, df = 109, p-value = 9.134e-06
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -0.9822838 -0.3957253
sample estimates:
mean in group 1 mean in group 2
       13.70755        14.39655
```

Run a Welch's *t*-test that allows for unequal variances. By leaving out the command var.equal=TRUE, the test will default to a Welches test.

```
t.test(MASS~BROODPATCH,data=swallows)
```

RESULT

```
     Welch Two Sample t-test

data:  MASS by BROODPATCH
t = -4.7206, df = 104.2, p-value = 7.353e-06
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -0.9784383 -0.3995708
sample estimates:
mean in group 1 mean in group 2
       13.70755        14.39655
```

You can see that there is a slight difference in t-value, and the degrees of freedom have been penalised by the Welch's t-test. They are 109 for the equal variance t-test, but 104.2 for the Welch's. The P value is different, but in both instances the P value is strongly significant. The Welch's t-test would be the preferable t-test to use here (variances are clearly not equal), and if reported in brackets in a Results section it would look something like this (t = -4.72, df = 104.2, P < -.001).

Effect sizes for t-tests

It is generally a good idea to report an effect size as well as the results of any statistical test. For t-tests, the most straightforward effect size is simply the difference between the two means (making sure to make it clear which mean is higher). A standardised effect size for t-tests, however, does exist, and is called a Cohen's *d*.

The Cohen's *d* is a measure of difference in means between two groups given the standard deviation as a whole. A Cohen's d isn't often used for publication in biological sciences journals (although psychologists like to use it), but it is certainly acceptable in university reports as a way to summaries the strength of difference between two populations.

```
install.packages("lsr")
# Download library from the internet.
# Only needed if you haven't already installed the package

library(lsr)
cohensD(MASS~BROODPATCH,data=swallows)
```

```
[1] 0.8848034
```

The author of the statistic, Cohen, suggested that *d* = 0.2 be considered a 'small' effect size, 0.5 represents a 'medium' effect size and 0.8 a 'large' effect size. This means that where you obtain a Cohen's *d* of <0.5, you should consider the difference to be biologically modest, even if the difference is also significant. The effect we obtained, where *d* = 0.88 would suggest that there is a large effect of mass as a function of swallow sex.

Further Example: Tails

Some researchers have suggested that swallow tails may serve a sexual selection function. If this were the case we might expect tail lengths to differ among males and females. Import the adult swallows dataset if you haven't already.

Import the adult swallows dataset:

```
swallows <- read.table('swallows-adults.csv',
header=T,sep=',')
```

Check the data:

```
head(swallows)
str(swallows)
```

t-test steps

1) Create of boxplot of tail length (mm) by brood-patch (a proxy for sex where 1 = no brooding patch (male), and 2 = brooding patch (female)).

```
boxplot(TailL~BROODPATCH,data=swallows)
```

2) Check the assumption of normal data
3) Check the assumption of equal variances
4) Decide which test is appropriate and run the appropriate test
5) Check the results

Two sample t-test (paired)

Paired t-tests are more powerful than unpaired tests because in a paired experimental design some of the noise in the background is eliminated by pairing controls and treatments together in the same conditions.

ASSUMPTIONS OF PAIRED T-TEST

(1) Observations must be independent (collected randomly)
(2) Both datasets are normal
(3) Equal variances (there is no Welch's version of a paired test)
(4) Data is from a paired experimental design

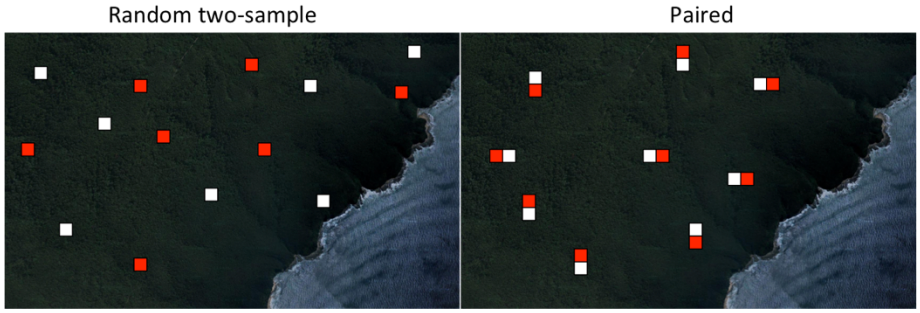


Figure 3. Paired experimental designs are more powerful than random two-sample designs and a paired t-test takes this into account. White boxes = control sites. Red boxes = treatments.

Import the mean seedling height paired and unpaired data sets. These contain mean measurements of seedlings in 400 m^2 quadrats measured in forest plots. The hypothesis being tested is that the plants in this forest are boron limited. Boron is an element that is essential to plant growth and some soils are low in boron. At treatment sites boron has been applied, whereas at control sites an inert powder has been applied.

Import the mean seedling height datasets:

```
msh.p <- read.table('msh-paired.csv',header=T,sep=',')
msh.u <- read.table('msh-unpaired.csv',header=T,sep=',')
```

Look at the way the data is laid out. Paired and unpaired datasets have quite different expectations around data layouts. We're going to skip assumption testing just for the sake of time, but remember that you must test assumptions if you plan to report results.

```
View(msh.p)
View(msh.u)
```

Run a Welch's *t*-test, allowing for unequal variances on the unpaired data:

```
t.test(MSH.400m2~TREATMENT, data=msh.u)
```

RESULT

```
     Welch Two Sample t-test

data:  MSH.400m2 by TREATMENT
t = -0.9364, df = 21.978, p-value = 0.3592
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -1.2296869  0.4646869
sample estimates:
  mean in group control mean in group treatment
               4.495833                4.878333
```

Now run a paired *t*-test on the paired data:

```
t.test(msh.p$CONTROL, msh.p$TREATMENT, paired=T)
```

RESULT

```
     Paired t-test

data:  msh.p$CONTROL and msh.p$TREATMENT
t = -3.4057, df = 11, p-value = 0.00587
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 -0.6296955 -0.1353045
sample estimates:
mean of the differences
                -0.3825
```

These data sets are exactly the same except that one is paired and the other is not. What are the *P*-values that you obtained? Is one result significant when the other is not? Why is this the case? What does pairing do to the 'power' of a test?

Regression Analysis

Simple linear regression generates a mathematical model that relates the magnitude of one variable to that of another. The general equation of a straight line is:

$$y = a + bx$$

...where a is the y–intercept (value of x when $y = 0$) and b is the slope of the line (rate at which y changes per unit change in x). Importantly, when the slope of the line (i.e., b) equals zero, there is no relationship between the response (dependant) (Y) and predictor (independent) (X) variables. Linear regression summarises how the average values of one variable (the dependent or response variable) vary across a range of subpopulations defined by a linear function of the other variable (the independent or predictor variable).

The value of y when x is zero is equal to a. This is the intercept on the vertical axis.

$$y = a + bx$$
$$y = a + b * 0$$
$$y = a + 0$$
$$y = a$$

The value of a when y equals zero is $-bx$. This is the intercept.

$$y = a + bx$$
$$y - bx = a$$
$$0 - bx = a$$
$$- bx = a$$

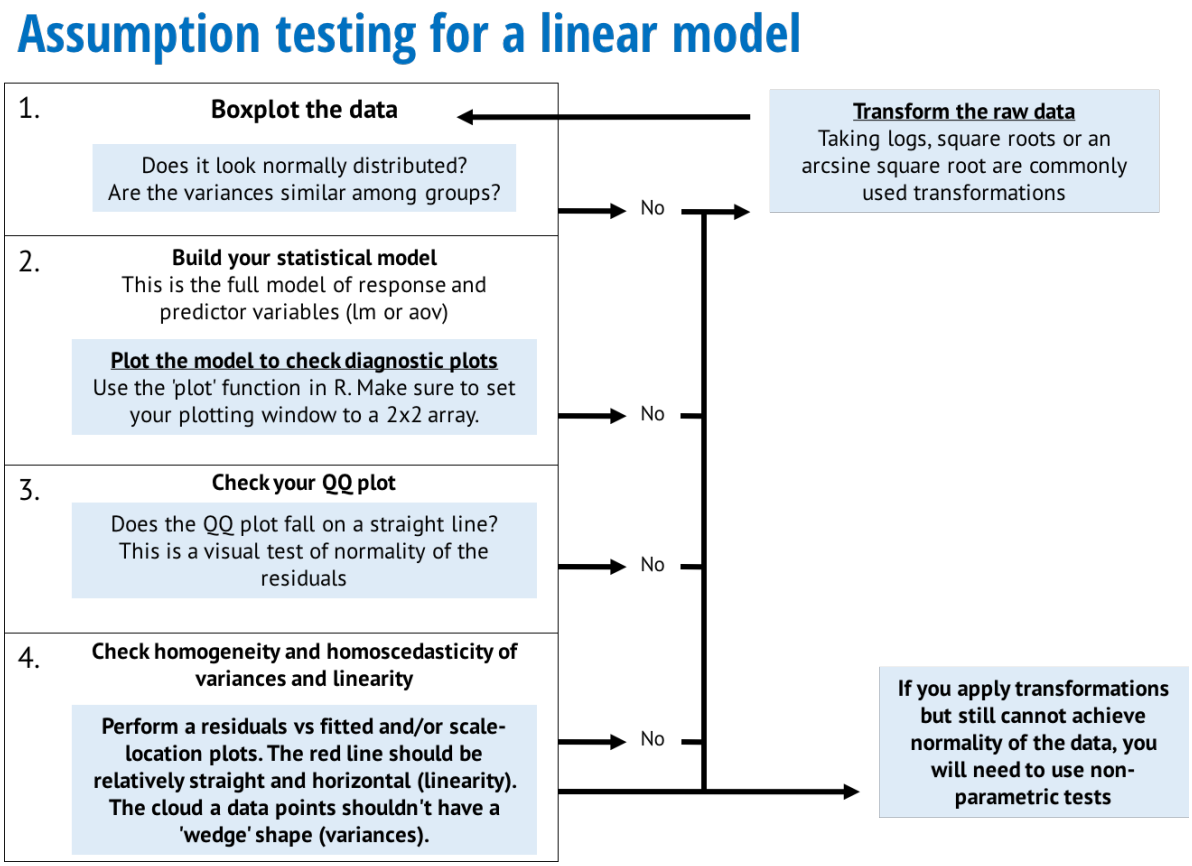
ASSUMPTIONS OF LINEAR REGRESSION

(1) Residuals are normally distributed
(2) Residuals must be independent (collected randomly)
(3) Residuals have equal variances (homoscedasticity)
(4) The relationship of x and y should be (relatively) linear

---

Assumptions of Linear Models

One set of tests that require fairly involved assumption testing are linear models. These include **regression analyses**, **ANOVAs** and **ANCOVAs**, and all the variants thereof. The following diagram provides a walk-through for testing linear model assumptions. Note that for a linear model it is **the residuals of the model** that need to meet assumptions of normality and equal variance, **not the original data**.



Note that the above diagram always applies to linear models, but if you have multiple predictors then you will have a couple of additional steps as well **1)** checking that predictors are not correlating and **2)** checking for significance of interaction terms.

Regression Analysis Example

We're going to use our nestling swallows dataset again, but this time we will test whether haemoglobin concentration in the blood (an indicator of health and parasite levels) has an effect on the mass of house swallow chicks at day 12 after hatching (an indicator of growth rate). Our hypothesis is that:

- House swallow chicks with a higher Hb will have a higher mass at day 12

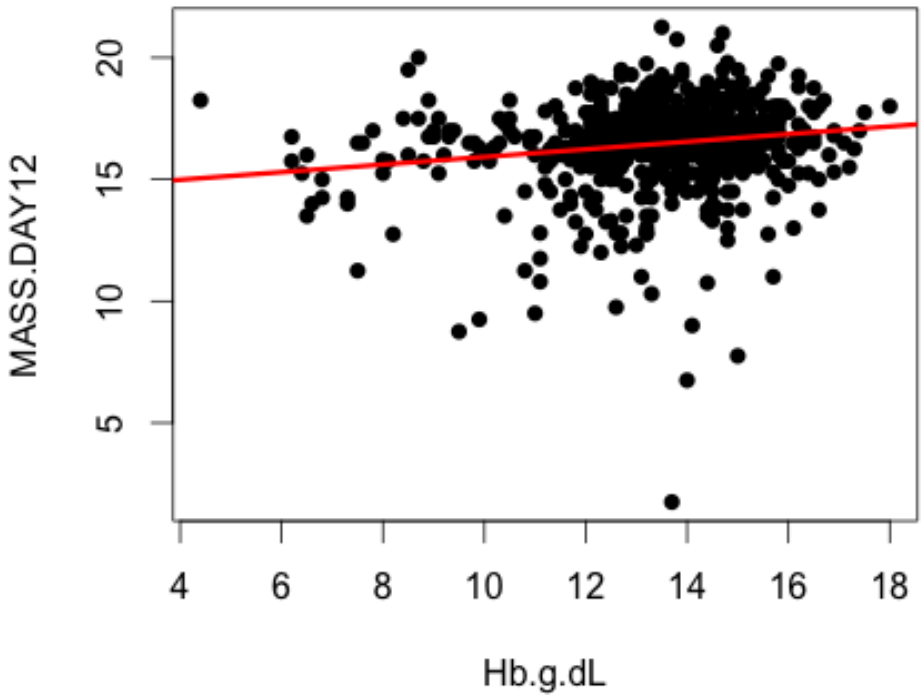


Figure 4. Plot of mass (g) of house swallow chicks at day 12 after hatching against haemoglobin concentration in the blood (g/dL).

Given the hypothesis we have proposed above, what would the null be?

Testing Assumptions

Sometimes you will find advice about testing assumptions of the underlying data for a regression analysis. Strictly speaking, the assumptions of a regression analysis apply to the residuals (values above and below expected) of the linear model, not the original data. Checking the residuals is most easily done in R by using diagnostic plots.

Wait... shouldn't we test residuals to the means for t-test then?

Actually, you could if you wanted to. A t-test is such a simple model, though, that in principal the result of assumption tests applied to the original data should be the same as the assumption tests applied to residuals taken to the means. Because testing residuals taken from the means would be a bit more involved, there is a tendency to just check the original data instead.

Diagnostic Plots

One very helpful feature of R is that it will generate diagnostic plots that help determine whether or not data is normal and whether there are outliers that are over-influencing the data:

```
nestlings <- read.table('swallows-nestlings-blood.csv',
header=T,sep=',')
```

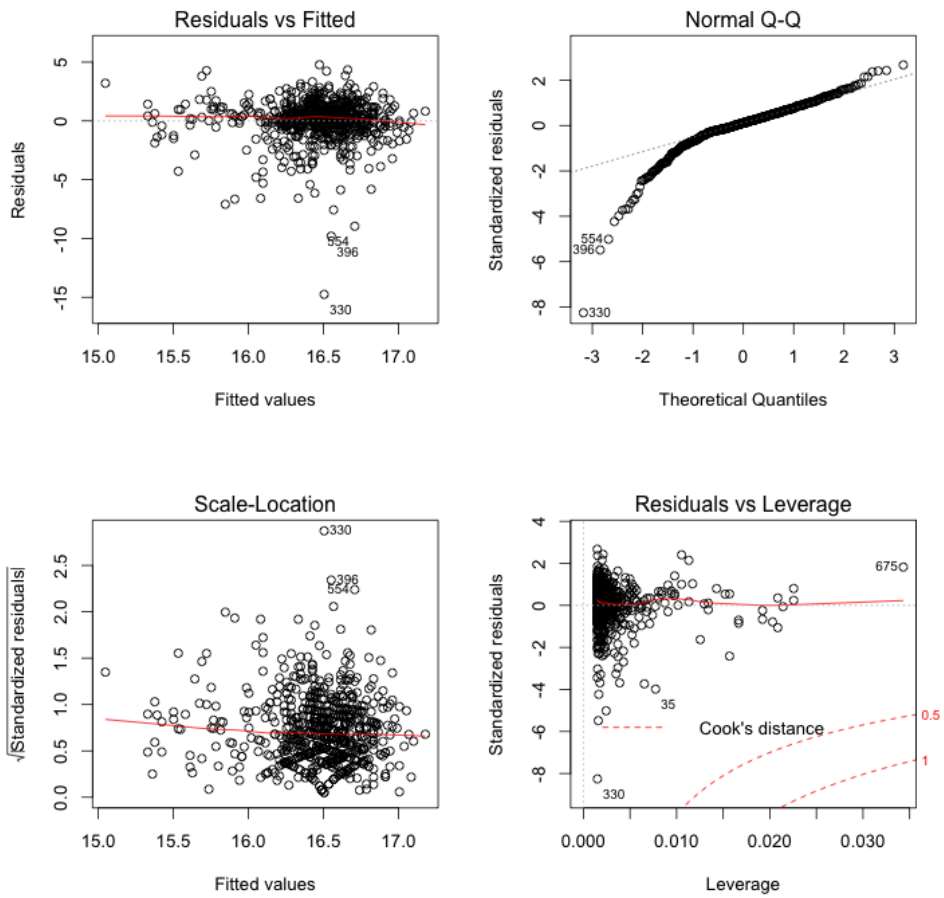
Set your window so that it will accept four plots at once (otherwise you need to click through the graphs):

```
par(mfrow = c(2, 2))
```

Plot the linear model diagnostic plots. This is done simply by plotting the model.

```
plot(lm(MASS.DAY12 ~ Hb.g.dL, data = nestlings))
```

You should get a plot that looks like this:



Reset your plotting window back to a single figure at a time to avoid future confusion:

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

Residuals vs Fitted

Used to check **linearity** and **homogeneity of variances**. The red line should be (relatively) straight and horizontal for the data to be linear. The cloud of data should be (relatively) amorphous (cloud like), and should not have a 'wedge' or 'arrowhead' shape. **Assessment:** *Our red line seems straight and horizontal. The cloud of data is perhaps a little 'pointy' to the right, but it isn't forming a clear arrowhead shape. This suggests the model is relatively linear and the variances of the model are relatively equal across the range of the predictor.*

Normal Q-Q Plot

Used to check **normality of residuals**. The horizontal axis plots the values that the residuals should show if the residuals were perfectly normal. The vertical axis shows the actual residuals. It follows then that if the data is departing from normality, the dots (real vs theoretical residuals) would wander off the red line, which represents a perfect 1:1 relationship. **Assessment:** *Our data seems to have some normality problems at the lower end of its range. Biological data is often non-normal, and we might decide to accept the departure from normality here (it's probably a bit borderline and depends on how conservative you want to be). If we were concerned about the normality we would use a non-parametric correlation test (e.g. Kendall tau) instead.*

Scale-Location

Used to check **linearity** and **homogeneity of variances**. You read this plot in the same way as the Residuals vs Fitted plot. The Residuals vs Fitted plot is consider a (slightly) better diagnosis plot for linearity, whereas the Scale-Location is considered a (slightly) better diagnosis plot for equal variances. **Assessment:** *Very similar to our interpretation for the Residuals vs Fitted, above. The two plots appear to be in agreement.*

Residuals vs Leverage

This is not strictly speaking an assumption test at all, rather it is checking whether there are any data points that would have a significant effect on the model if removed. This can be thought of as a test for statistical outliers. If a data point is on the other side of the 0.5 or 1.0 Cook's Distance (from the rest of the data), then it is having a substantial independent affect on the model as a whole. **Assessment:** *All our data points are sitting in a cloud to the left of the Cook's distance lines. All is okay then.*

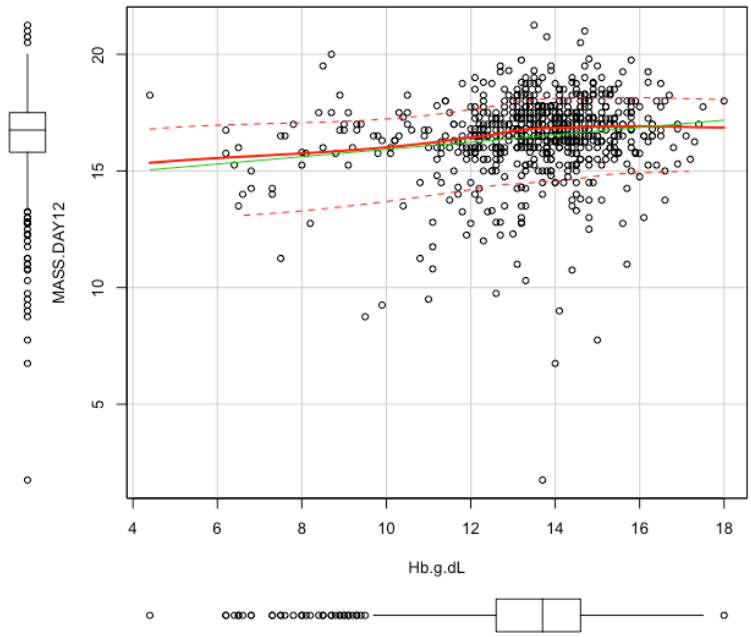
What to do with Outliers?

My standard advice is that unless you have a strong reason to believe that an outlier was due to human error or machine error, you should leave it in the data. The reason for this is that removing outliers can lead to a temptation to start 'adjusting' the data so that it fits expectations (i.e. the hypothesis), which is far from ideal.

Scatterplot in Library 'car'

The scatterplot function in library car also provides a clear visual assessment of linearity and whether variance (uncertainty) around a line of best fit might be increasing.

```
install.packages("car")
# Download library from the internet.
# Only needed if you haven't already installed the package

library(car)
scatterplot(MASS.DAY12 ~ Hb.g.dL, data = nestlings)
```



The plot fits a straight line (green, solid), and loess smoothed lines (red) with 50% confidence intervals above and below the line (dashed red). If the confidence intervals form a 'funnel' then there is probably a problem with variance changing across the range of the predictor. You can modify the plot using standard graphic parameters as well as some parameters built into the function. The boxplots are plotted automatically, but can be switched off by setting to false. Use the following code to bring up a help menu and check the options.

```
?(scatterplot)
```

Overall Assessment

Given the way the data points are wandering off the Q-Q line, it might be preferable to either transform the data to achieve normality of residuals, or use non-parametric tests. Nonetheless, it is a bit of a borderline call here. Although the QQ plot is not ideal, regression analyses are (somewhat) robust to (small) departures from normality.

For simplicity, we're going to leave the data as is, and proceed with the analysis. Run the following code.

```
nestlings.lm <- lm(MASS.DAY12 ~ Hb.g.dL, data = nestlings)
summary(nestlings.lm)
```

You could also 'pipe' the code using the `tidyverse` syntax if you like:

```
library(tidyverse)

lm(MASS.DAY12 ~ Hb.g.dL, data = nestlings) %>%
summary()
```

Or you can pile everything into a single line of code, but keep in mind you are risking annoying errors if you use this approach:

```
summary(lm(MASS.DAY12 ~ Hb.g.dL, data = nestlings))
```

RESULT

```
     Call:
lm(formula = MASS.DAY12 ~ Hb.g.dL, data = nestlings)

Residuals:
     Min       1Q   Median       3Q      Max
-14.7546  -0.5427   0.2014   0.9954   4.7767

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.35776    0.48662  29.505  < 2e-16 ***
Hb.g.dL      0.15670    0.03581   4.376  1.4e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.789 on 673 degrees of freedom
Multiple R-squared:  0.02767,    Adjusted R-squared:  0.02623
F-statistic: 19.15 on 1 and 673 DF,  p-value: 1.398e-05
```

```
Call:
lm(formula = MASS.DAY12 ~ Hb.g.dL, data = nestlings)
```

This is simply a reminder what linear model we have just run.

```
Residuals:
     Min       1Q   Median       3Q      Max
-14.7546  -0.5427   0.2014   0.9954   4.7767
```

These are the ranges, upper and lower quartiles and median of the model. These are the same values used to make a boxplot. You typically wouldn't report these values in a Result section.

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.35776    0.48662  29.505  < 2e-16 ***
Hb.g.dL      0.15670    0.03581   4.376  1.4e-05 ***
```

These are the effect sizes **Estimate)** *standard error of the effect sizes* **(Std Error)**, *t-value and P value* **Pr(>|t|)**. *These are the values that you would report in a Results section either in brackets or in a table. Remember that any value of P that is very small (as is the case here) would be reported as P < 0.001. We would not report these as P = 0.0000000000000002 or P = 0.000014, which are the actual P values that have been returned.*

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

These codes for symbols indicating different significance levels. For our purposes we are going to stick with the standard 0.05 significance level.

```
Residual standard error: 1.789 on 673 degrees of freedom
```

The residual standard error (unexplained variance) for the whole model and the degrees of freedom for the residual variance. You typically wouldn't report these values in a Results section.

```
Multiple R-squared: 0.02767,    Adjusted R-squared: 0.02623
```

The Multiple R^2 and Adjusted R^2 for the whole model. Typically, you would report the Adjusted R^2 (assuming you want to report an R^2), as this takes into account departure from parsimony effects caused by using multiple predictors.

```
F-statistic: 19.15 on 1 and 673 DF,  p-value: 1.398e-05
```

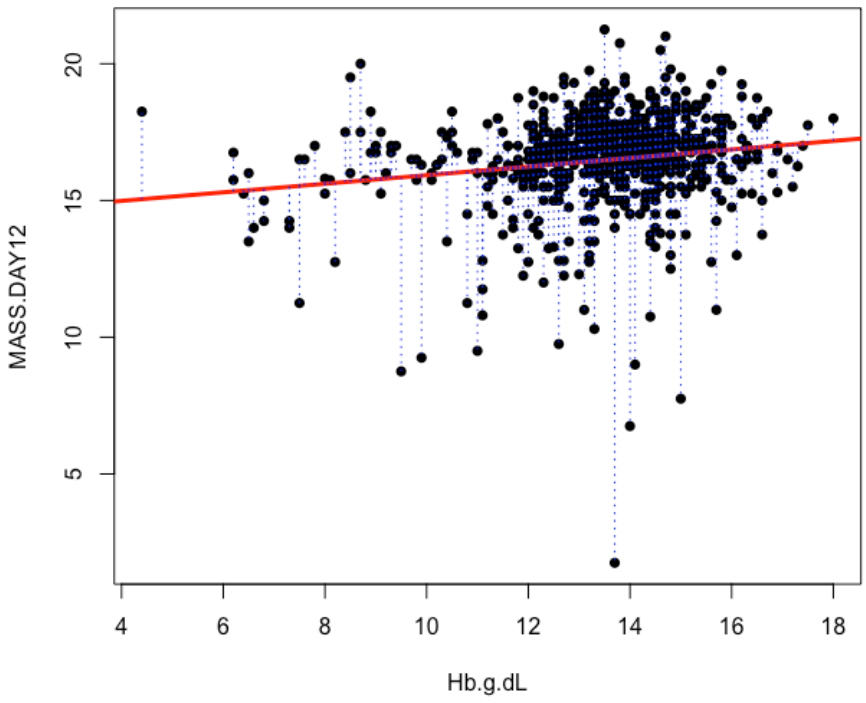
F-value, degrees of freedom and P for the whole model. Again, this is typically not reported in a Result section.

Plotting the regression model

You can have a go at plotting the residuals of the model, if you like. These residual plots are typically not reproduced in scientific papers, but it can be useful to know how to make one.

First, make sure that the model is constructed:

```
nestlings.lm <- lm(MASS.DAY12 ~ Hb.g.dL, data = nestlings)

par(mfrow = c(1, 1)) # reset to one graph per display

plot(MASS.DAY12~Hb.g.dL, pch = 16, data = nestlings)

abline(lm(MASS.DAY12~Hb.g.dL,
data = nestlings),col="red",lwd=3)
# lwd = line width

segments(nestlings$Hb.g.dL, fitted(nestlings.lm),
nestlings$Hb.g.dL, nestlings$MASS.DAY12,col="blue",lty=3)
# lty = line type. Try line types = 2, 4 or 5 if you like.
```



Multiple Regression

Multiple Regression is just a term for a regression analysis that has two or more continuous predictors. There is nothing about it that is much more complicated than a simple regression, except that now you need to check for interaction terms. In principal though, checking interaction terms is the same process as for ANOVAs, so we will deal with that in the next section.

ANOVA, ANCOVA & related models

An Analysis of Variance (ANOVA) is one of the most commonly used tests that you will see in the biological sciences literature. It is a form of variance analysis and is related to a *t*-test.

An ANOVA tests the null hypothesis that all means of response variables for groups are the same.

In practice, this means that an ANOVA can be used to tell whether different treatment levels are associating with different means of the response variable. As an example, consider an experimental set-up where we are growing tomato plants under full light, partial shade and heavy shade. This experiment has one response variable (height of the tomato seedlings at week 10) and three levels (light, shade, heavy shade) of a single factor (lighting treatment). Although the test is asking is there a difference in the mean values of seedling height among the groups, the usefulness is that from this we can determine whether a particular lighting treatment is associating with different levels of seedling growth.

It is perhaps a little complicated, but in a sense, an ANOVA asks a question about differences in order to address a question about trends.

An ANOVA compares residuals in a way that is similar to how a regression model works except that residuals are taken from the mean of each group rather than across a range of values. An *F*-ratio is derived from the variance of the residuals.

F = signal / noise
F = variance between treatments / variance within treatments
F = mean squares of treatment / means squares of the error
F = [sum of squares of treatment / [(T-1)] / [sum of squares of error / (n-1)]

The *F*-ratio is conceptually very similar to the *t*-value. It is a measure of the signal to noise in the data. It is equivalent to:

F = explained variance / unexplained variance

This means that an F ratio is somewhat similar to an R^2 value, except that instead of dividing by total variance (i.e. and then getting percentage of total variance explained), we obtain a ratio of the explained to unexplained variance.

ASSUMPTIONS OF ANOVAs (and ANCOVAs etc)

(1) Residuals are normally distributed

(2) Residuals must be independent (collected randomly)

(3) Residuals have equal variances

OTHER PROBLEMS MAY OCCUR IF...

(1) Predictor variables correlate (r > 0.6 considered problematic)
(two variables explain the same thing)

(2) The model is over-parameterised
(too many predictors given your data set size)

(3) You fail to check for significance of interaction terms
(your main effects could be meaningless)

ANOVAs are relatively robust to departures from the assumption of normality. However, if variances are not equal and if variances of residuals are proportional to the predictor variable(s), serious problems can develop. Independence of residuals must also be maintained. If not, pseudoreplication can result, and your *P*-values will likely be far stronger than they should be.

Terminology

ANOVAs have acquired a set of terminology that is confusing when first encountered and is arguably out-dated. The distinction between a one-way and two-way ANOVA was more important when ANOVAs were calculated using dedicated statistical calculators that might take hours or days to programme, and a two-way ANOVA might be all that was feasible. Now, it is sometimes preferable to refer to a test in the ANOVA family as a general linear model.

Figure 6.14. Wang 462 Statistical Programmable Calculator from about 1975.

Table 6.3. Experimental designs and appropriate formulas for ANOVAs in R. `y` is the response variable. `A`, `B` and `W` are explanatory factors (discrete variable). `x1` and `x2` etc are covariates.
All formula use: `aov(formula , data = your.data)`.

Design	Formula
One-way ANOVA	`y ~ A`
One-way ANCOVA with one covariate	`y ~ x * A`
Two-way Factorial ANOVA	`y ~ A * B`
Three-way Factorial ANOVA	`y ~ A * B * C`
Four-way Factorial ANOVA	`y ~ A * B * C * D`
Two-way Factorial ANCOVA with two covariates	`y ~ x1 * x2 * A * B`
Nested ANOVA	`Y ~ A/B/C`
Split-plot ANOVA	`Y ~ A * B * C + Error(A/B/C)`
Randomised Block (where B is blocking factor)	`y ~ B + A`
One-way within-groups (repeated measures) ANOVA	`y ~ Error(Subject/A)`
Repeated measures ANOVA with one within-groups factor (W) and one between-groups factor (B)	`y ~ B * W + Error(Subject/W)`

Some words of advice

Nested ANOVAs are mathematically complicated, and you should only be trying to construct a nested model once you have a very clear idea what you are doing. **Within-group ANOVAs** (also called a **repeated measures ANOVA**) work fine as long as the experimental design is **perfectly balanced**. As soon as your data moved away from balance (even a little bit), Within-group ANOVAs can develop substantial problems and it becomes far more preferable to control for groups using a linear mixed effects model with the group as a random effect in the model. All of the experimental designs that are **shown in blue (bottom half)** may be better approached through use of mixed effects models.

I still don't understand! Why are we 'testing assumptions'? None of this makes sense... can't we just do an ANOVA?

Statistical tests, like ANOVAs or t-tests or chi squared tests, all carry assumptions about the data. A t-test for example assumes the data consists of two sets of normally distributed data. If you tried to feed non-normal data into a t-test, you might get a result, but the result would be unreliable.

All we are doing is testing whether the data meets the test assumptions, and then (if it doesn't) we try to transform the data so that it does meet assumptions, or look for another more suitable test.

Assumptions are confusing. Non-parametric tests don't have assumptions. Why not just use them from the start?

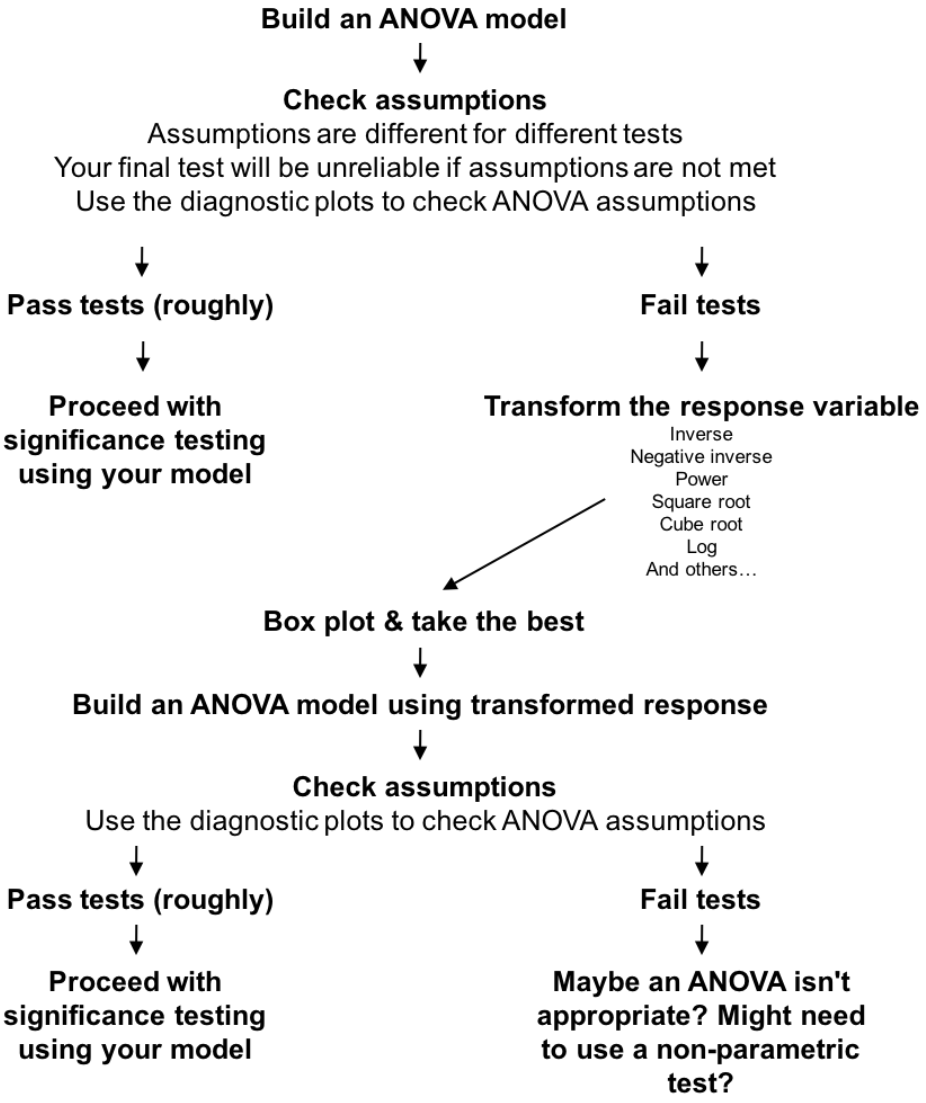
Most non-parametric tests work by ranking the data. This is a form of very drastic transformation, and it discards a lot of information from a model. A transformation, like a log transformation or a square root transformation, is an attempt at going half-way: we're trying to 'correct' the data, and we know we will lose *some* information, but hopefully most of the information in the distribution will remain. We're stuck between needing the data to meet the assumptions of a test, but being reluctant to transform the data really drastically because we know that could change the data to a point of losing too much information.

Also, non-parametric tests inflate Type II error (the risk of accidentally accepting the null when we shouldn't). Because scientists are obsessed with significance, they sometimes do seem to avoid non-parametric tests for this reason.

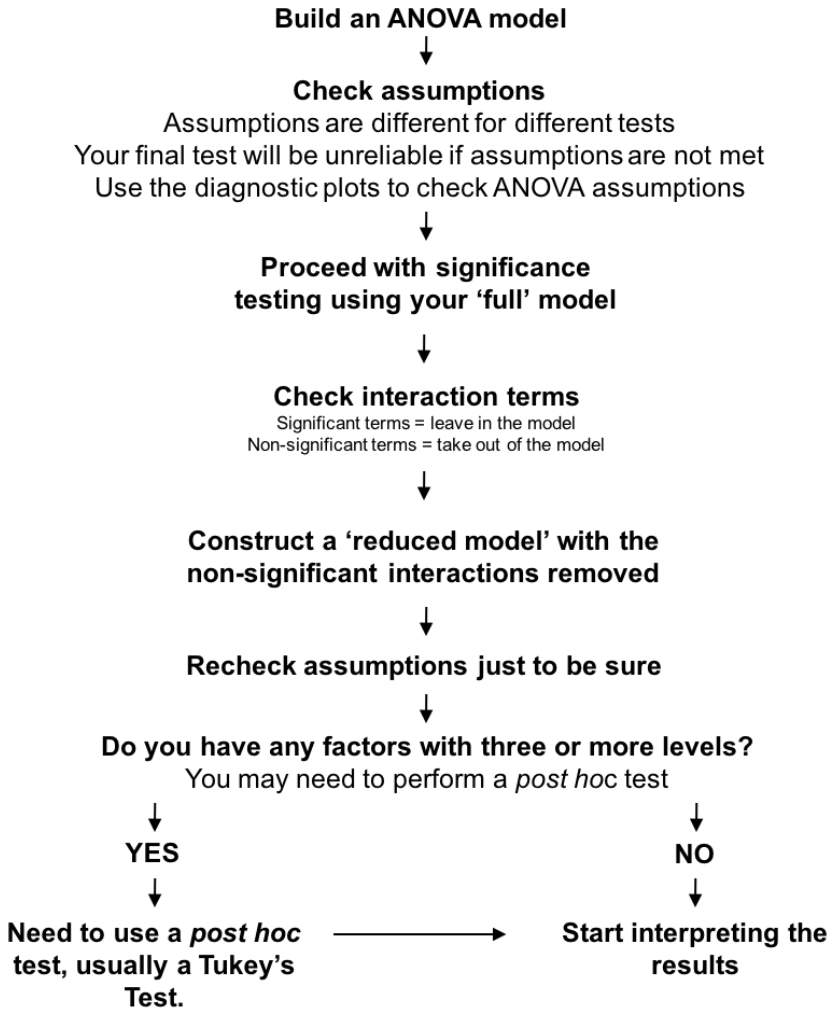
Testing assumptions for ANOVAs and other linear models is confusing. Can't you break it down a bit to make it simpler to follow?

Alright. Have a look at the flow diagram on the next page.

Testing assumptions in ANOVAs (and other ANOVA-like linear models) is quite involved. Here is a flow chart to help you visualize the process.



At a more fine-grained level, we also have to check the interaction terms and think about whether or not we will require a *post hoc* test, such as a Tukey's test.



We will leave this here for now, and examine what is meant by 'interactions' and *'post hoc'* a little way down the track.

For now, we can return to checking assumptions for an ANOVA. The assumptions are the same for ANOVAs, ANCOVAs and other ANOVA-like linear regression models. These all (effectively) work the same way under the hood, which means that the assumption testing is the same for all of these basic linear models.

DIAGNOSTIC PLOTS

You can use the plot command to look at diagnostic plots. The current advice is that it is better to use diagnostic plots to check assumptions in an ANOVA (or any regression-type linear model) rather than using assumption tests like Shaprio-Wilks or Bartlett tests.

```
par(mfrow = c(2, 2))
plot(fullmodel.aov)
```

Residuals vs Fitted (homogeneity of variance in residuals)

The Residuals vs Fitted plot can be used to check homogeneity of residuals and linearity of a model. If the residuals are equal (homogenous) there should be no pattern or shape to the scatter of points. If there is a wedge (arrow-head or side-ways triangle) shape to the data points (thin at one end, thick at the other) the residuals of the model are not equal across the model. If it looks like a random cloud of scattered points you're ok. As long as the red line is (relatively) horizontal and straight, the model is probably linear.

Normal QQ (normality of residuals)

The Normal QQ plot can be used to check the normality of residuals. If the residuals are normal the observations (circles) should fall on the line of normality

Scale-Location

This also can be used to test for homogeneity of residuals and linearity of a model. It is read in exactly the same way as the Residuals vs Fitted plot. If the plots disagree, the Residuals vs Fitted is considered (slightly) better for linearity, whereas the Scale-Location is considered (slightly) better for examining equal variances across the range of a predictor.

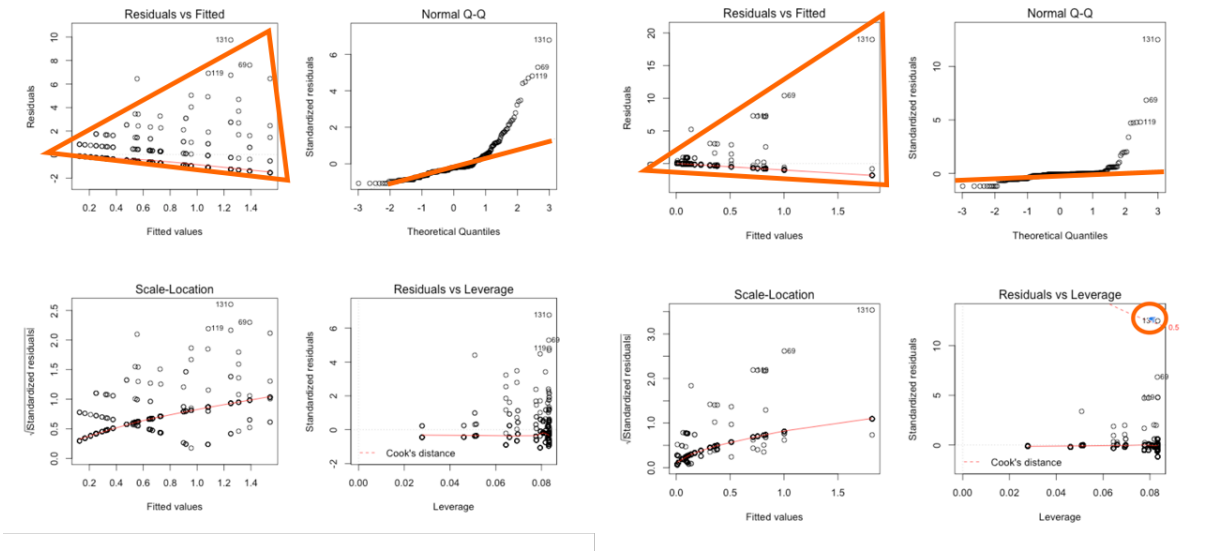
Residuals vs Leverage

This is a more easily interpreted check for outliers. If observations are scattered so that they are sitting beyond the 0.5 or 1 Cook's distance lines you have a potential problem with outliers.

Checking tests visually: *Most statisticians now seem to recommend you should check the assumptions visually rather than rely on significance tests. Significance tests for assumptions can provide a nice yes/no answer, but assumption testing is a bit more nuanced than this, and sometimes you need to make a judgment call whether the data looks basically ok.*

The following pages have some examples of diagnostic plots that have some problems in them.

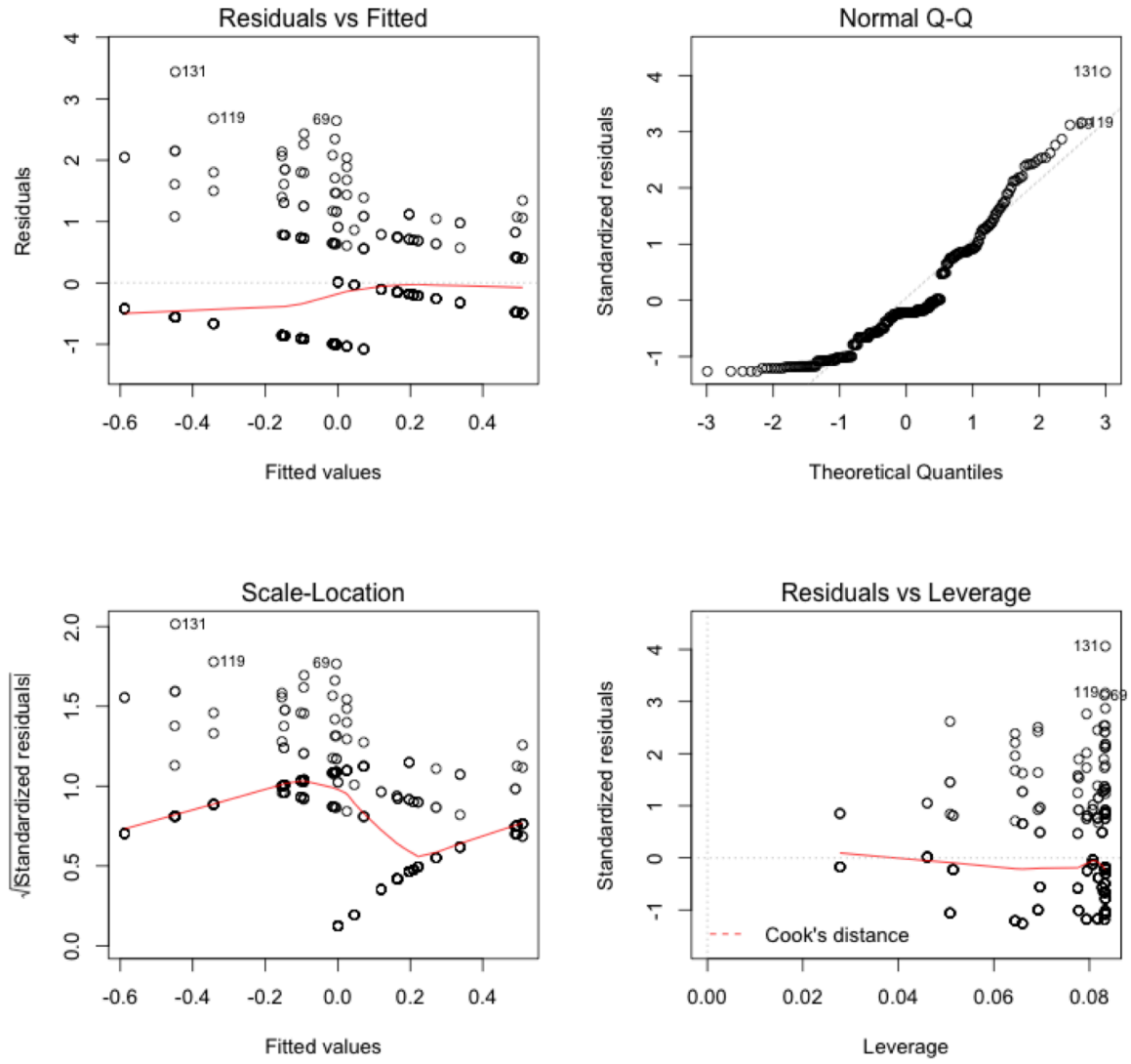
Untransformed **Transformed**



These plots have a couple of problems. For the untransformed data, you can see (on the left) that the residuals vs fitted plot has a clear 'wedge' shape (I've drawn a line around it in orange unless it is not clear) and the QQ plot is showing a fairly drastic tendency away from the line of normality (indicated in orage). The residuals vs leverage plot appears to be fine.

I've transformed the data and made it, if anything, worse (i.e. transformations don't always improve data). There is still a wedge shape in the Residuals vs Fitted and now the Scale-Location is developing a wedge as well. The QQ plot is maybe a little better, but a lot of points are still clearly departing from the line of normality. And now we have a clear outlier too (Residuals vs Leverage). Point 131 is on the other side of the 0.5 Cook's D line, which indicates it is having too heavy an effect on the model as a whole.

Let's try a more aggressive transformation and see if we can improve the fit of the data.



These look better. I achieved this by applying a rank normal transformation to the data and using the rank normal transformed data as a response variable. The rank normal transformation is a very drastic transformation that forces the data into a normal distribution. It won't work if you have too many ties or if there are a lot of zeroes in the data, but otherwise will tend to work with most data. In essence, using a rank normal transformation is like forcing your ANOVA to perform like a non-parametric test. It is far from ideal, but you might be stuck having to take an approach like this.

I've given the code below for a rank normal transformation, but here it is as well in case you want to try it straight off the bat:

```
install.packages("GenABEL")
library(GenABEL)

yourdata$RANK.NORMAL <- rntransform(yourdata$variable)
```

One-Way ANOVA

Import the adult house swallows dataset (if you haven't already):

```
swallows <- read.table('swallows-adults.csv',
header=T,sep=',')
```

Check the data:

```
head(swallows)
str(swallows)
```

Run the following code on the adult swallows data:

```
swallows.aov <- aov(MASS~BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
             Df Sum Sq Mean Sq F value   Pr(>F)
BROODPATCH    1  13.15  13.147   21.68 9.13e-06 ***
Residuals   109  66.10   0.606
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Check the *P*-values of your ANOVA result above with the *P*-values of the equal and unequal variance *t*-tests we just did on the same data. Does the *P*-values match?

```
t.test(MASS~BROODPATCH,data=swallows, var.equal=TRUE)
```

RESULT

```
     Two Sample t-test

data:  MASS by BROODPATCH
t = -4.6563, df = 109, p-value = 9.134e-06
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -0.9822838 -0.3957253
sample estimates:
mean in group 1 mean in group 2
       13.70755        14.39655
```

One-Way ANCOVA with one covariate

Run the following code on the adult swallows data:

```
swallows.aov <- aov(MASS~WingL*BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value   Pr(>F)
WingL              1   4.38   4.375   7.621  0.00679 **
BROODPATCH         1  12.63  12.626  21.991 8.13e-06 ***
WingL:BROODPATCH   1   0.81   0.809   1.409  0.23793
Residuals        107  61.43   0.574
```

Two-Way ANOVA

Run the following code on the adult swallows data:

```
swallows.aov <- aov(MASS~MONTH*BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value   Pr(>F)
MONTH              4   4.39   1.099   1.980    0.103
BROODPATCH         1  16.94  16.942  30.538 2.52e-07 ***
MONTH:BROODPATCH   3   1.32   0.440   0.793    0.501
Residuals        102  56.59   0.555
```

Effect sizes for ANOVAS

It is generally a good idea to report an effect size as well as the results of any statistical test. For ANOVAs the eta squared (η^2) is an effect size that is frequently reported in psychology and medical studies, but is perhaps underused in biological sciences. An is equivalent to an R^2 (i.e. it is a percentage of variance explained, but for each predictor). The eta squared function in the **lsr** library provides both the standard η^2 (which you would report) and the partial η^2 (which you wouldn't typically report). The partial η^2 is the variance explained as if none of the other predictors were in the model, which isn't highly interpretable if you are actually interested in hypothesis testing based on the ANOVA as a whole.

```
install.packages("lsr")
# Download library from the internet.
# Only needed if you haven't already installed the package

library(lsr)

swallows.aov <- aov(MASS~WingL*BROODPATCH,data=swallows)
etaSquared(swallows.aov)
```

RESULT

```
                  eta.sq        eta.sq.part
WingL             0.04864008    0.05903723
BROODPATCH        0.15933154    0.17048468
WingL:BROODPATCH  0.01020536    0.01299295
```

```
swallows.aov <- aov(MASS~MONTH*BROODPATCH,data=swallows)
etaSquared(swallows.aov)
```

RESULT

```
                  eta.sq       eta.sq.part
MONTH             0.1033432     0.12642313
BROODPATCH        0.2137978     0.23041207
MONTH:BROODPATCH  0.0166550     0.02279164
```

The η^2 is interpreted in the same was an R^2 is interpreted. A value of 0 = zero variance explained, whereas 1 = 100% of variance explained.

Changing the order of predictors

Now run these two sets of code and look at the P-values comparing them to the tests you just did on the previous page. Have the P-values changed? Have any of the results changed in terms of their significances?

```
swallows.aov <- aov(MASS~BROODPATCH*WingL,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value  Pr(>F)
BROODPATCH         1  13.15  13.147  22.898 5.5e-06 ***
WingL              1   3.85   3.854   6.713  0.0109 *
BROODPATCH:WingL   1   0.81   0.809   1.409  0.2379
Residuals        107  61.43   0.574
```

```
swallows.aov <- aov(MASS~BROODPATCH*MONTH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value   Pr(>F)
BROODPATCH         1  13.15  13.147  23.698 4.12e-06 ***
MONTH              4   8.19   2.047   3.690  0.00756 **
BROODPATCH:MONTH   3   1.32   0.440   0.793  0.50054
Residuals        102  56.59   0.555
```

What you may have noticed is that the P-values are changing depending on the order of the variables. This often causes students to have a mini-crisis of confidence when they first notice this. So, what is going on here?

Predictor order is important

When you changed the order of the predictor variables, the P-values may change. You might find when you do this that some variables will fall in and out of significance. What is going on?

There are three ways to partition the variance in **y** among the explanatory effects on the right side of the equation. The equation...

```
y ~ x1 + x2 + A * B
```

...is equivalent to...

```
y ~ x1 + x2 + A + B + A:B
```

If the design is balanced then all three types of variance partitioning (called the Type I, II and III sums of squares) will generate the same result and the order of predictors won't matter **(a good reason to keep your designs balanced)**. But if the design is unbalanced the different Types of sums of squares partition variation explained in different ways.

Type I (sequential)

Explanatory effects are adjusted for those that appear earlier in the formula. In the example above, **x1** is unadjusted, **x2** is adjusted for **x1**, **A** is adjusted for **x1** and **x2**, **B** is adjusted for **x1** and **x2** and **A**, and the interaction term **A:B** is adjusted for all other terms.

Type II (hierarchical)

Explanatory effects are adjusted for those at the same level. In Type II, **x1**, **x2**, **A** and **B** are all adjusted for each other but not for the interaction term. The interaction term **A:B** is adjusted for all the main effects (**x1**, **x2**, **A** and **B**). If there were two interaction terms, perhaps **x1:x2**, then **A:B** and **x1:x2** would also be adjusted for each other.

Type III (marginal)

Each explanatory effect is adjusted for every other effect in the model. This is the least powerful of the sums of squares approaches, and is arguably the most conservative, but it causes problems for *post-hoc* analyses like Tukey's tests which we will look at shortly.

R by default runs Type I sums of squares which are the best for *post-hoc* analyses. You can run Type II and Type III sums of squares tests using the **Anova** function in package **car** if you like, but it is often better to arrange the explanatory variables in an order that makes experimental sense. If you want to look at this function type:

```
library(car)
?Anova
```

In the example where we are looking at the effects of **Month** and **Broodpatch** on adult house swallow **Mass** it makes sense to write the model out as **Month**, then **Broodpatch**. This is because primarily we are interested in the effect of **Broodpatch**, but we have (presumably) included **Month** here because we know that swallows will gain and lose **Mass** over a breeding season and we want to control for this in the statistical model. In essence, by placing **Broodpatch** after **Month** we are asking: does **Broodpatch** (sex of swallows) associate with differences in **Mass** after we take into account the variation in **Mass** already explained by **Month**.

When working with Type I sums of squares models in R it is important to know what question you want to ask before writing out formulas. Some fundamental advice:

(1) Place a predictor that you wish to control for (especially if the predictor is not a part of your hypothesis) early in the sequence.

If you want to control for the effect of a predictor on another predictor, the controlled predictor must come first.

(2) Place predictor(s) that are key to your hypothesis late in the sequence.

This is placing your predictors that relate to your key hypothesis in a position where all the other variables have a go at explaining the variance first. This biases against finding significance in your predictor of interest, which is the most rigorous approach to take.

The key thing here is that placing your variables of interest early in the sequence is arguably a form of P-hacking because you are making a decision to favour them in terms of obtaining a significant *P*-value.

Also, although it may seem that adding additional variables will always make it harder to see significance in your variables of interest, this isn't necessarily true if you have been intelligent about your covariates. Covariates that are early in the sequence can also act to 'clear away' some of the confusion in the variance, leaving behind residuals that your predictor of interest is better able to explain. This is fundamentally what happens with the stress and city living example above, where including the covariate 'how long have you lived where you live', resolves some of the variation and allows for the variable of interest (city vs country living) to explain the remaining variation in a meaningful way.

Changing the interaction terms

Now run these three sets of code and look at the *P*-values comparing them to each other:

```
swallows.aov <- aov(MASS~MONTH*BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value   Pr(>F)
MONTH              4   4.39   1.099   1.980    0.103
BROODPATCH         1  16.94  16.942  30.538 2.52e-07 ***
MONTH:BROODPATCH   3   1.32   0.440   0.793    0.501
Residuals        102  56.59   0.555
```

```
swallows.aov <-
aov(MASS~MONTH+BROODPATCH+MONTH:BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                  Df Sum Sq Mean Sq F value   Pr(>F)
MONTH              4   4.39   1.099   1.980    0.103
BROODPATCH         1  16.94  16.942  30.538 2.52e-07 ***
MONTH:BROODPATCH   3   1.32   0.440   0.793    0.501
Residuals        102  56.59   0.555
```

```
swallows.aov <- aov(MASS~MONTH+BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                Df Sum Sq Mean Sq F value   Pr(>F)
MONTH            4   4.39   1.099   1.992    0.101
BROODPATCH       1  16.94  16.942  30.720 2.23e-07 ***
Residuals      105  57.91   0.551
```

Are there any changes in *P*-values (however small)? Did you notice that the interaction term `MONTH:BROODPATCH` has disappeared from the last model? What was different about the code in the third model?

Interactions are important

An interaction term is an additional factor or covariate in a model that is included to look for complex relationships between main effects (the ordinary non-interactive factors and covariates like **A** and **B** and **x1** and **x2**).

Because an interaction term changes the model, even in a Type I model, the *P*-values of the other terms will be (usually only slightly) changed.

What does a significant interaction mean?

There is a complex relationship and you can't interpret the main effects. Consider the following situation:

MASS ~	**MONTH**	***P* < 0.005**
	BROODPATCH	***P* < 0.005**
	MONTH:BROODPATCH	***P* < 0.005**

The significant interaction term means that you cannot interpret **Month** or **Broodpatch**. The interaction term could mean ***any*** of the following:

- Females are heavier than males, but only in August.
- Males are heavier than females but only in April.
- Females are heavier than males, but only in April and August

... and so on and so on

How to interpret a significant interaction term

If the interactions are between factors, then splitting the factors and analysing the levels separately is an acceptable approach. In the above example we might split up all the data by Month and then analyse Broodpatch separately for each Month.

Also, you can apply a *post-hoc* Tukey's test to an ANOVA model with factorial interactions. However, this generates a (very large) series of comparisons that may not be biologically meaningful and the power of your analysis will be substantially reduced. Only take this step if it makes some biological sense to compare all possible interactions.

What does a non-significant interaction term mean?

When an interaction term is non-significant it is reducing the explanatory power of the model. You should remove it, and re-run the model using **+** instead of ***** to separate the factors or covariates. If **`A:B`** is non-significant change the formulas from...

```
y ~ A * B
```

to

```
y ~ A + B
```

...and re-run the model.

However, keep in mind that:

1) You would keep a non-significant interaction term if the interaction term itself is testing the hypothesis of your study

2) For very large datasets removing non-significant interaction terms won't make much different to the P value

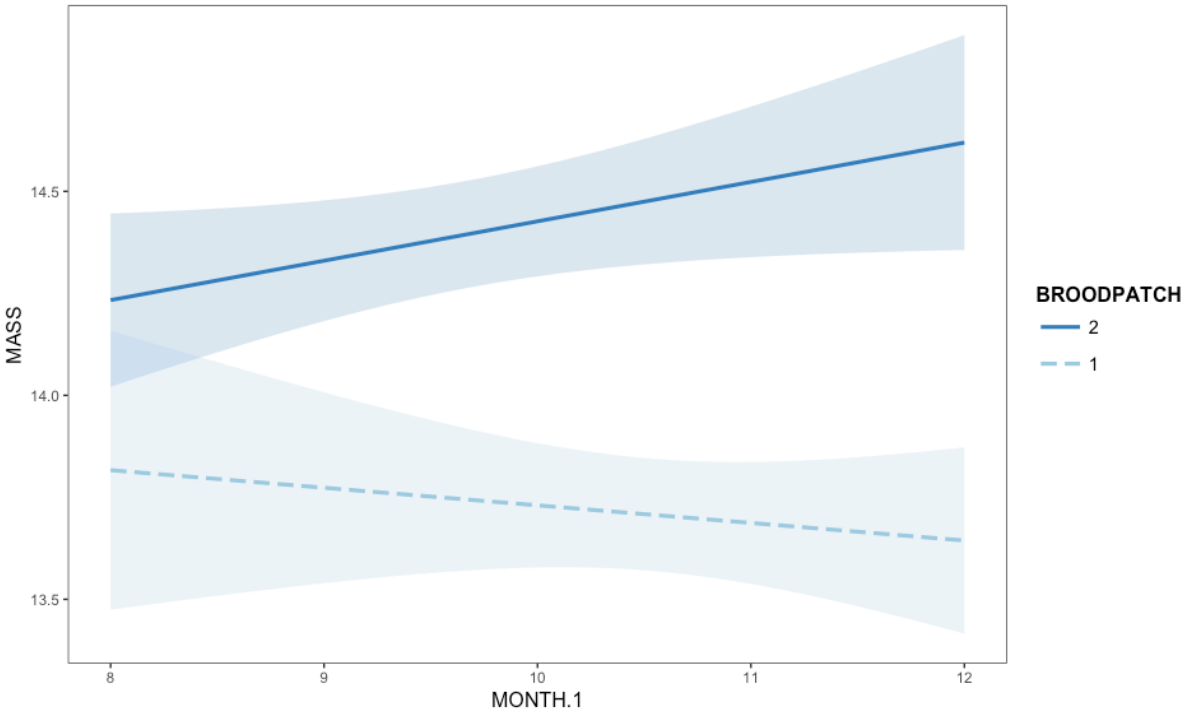
3) Some people prefer to leave non-significant interaction terms in a model, because removing them can look a bit like chasing after significance for main effects

4) Leaving a non-significant interaction term in a model is not a terrible statistical crime. Not checking for interaction terms in the first place may render your main effects meaningless without you even realising.

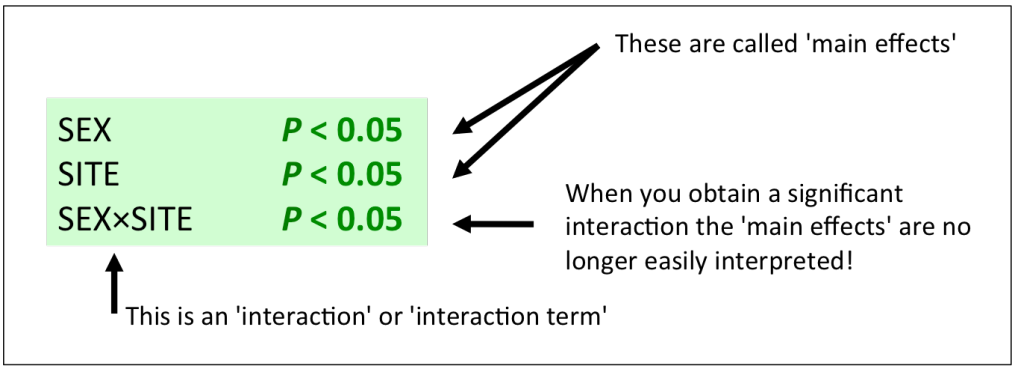
Interactions in linear models

Interaction terms can be one of the most difficult concepts to understand in statistics. A significant interaction happens when the effect of one predictor depends on the effect of another predictor. This is best explained visually.

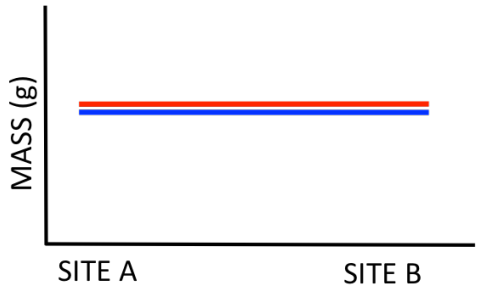
```
swallows.aov <- aov(MASS~MONTH.1*BROODPATCH,data=swallows)

library(interactions)
interact_plot(swallows.aov, pred = "MONTH.1", modx =
"BROODPATCH", interval = TRUE, int.width = 0.8)
```



I've used month as a numeric value (8 = August, 12 = December) to make it a bit easier to read the figure, but what you can see is that male (broodpatch = 1) and female (broodpatch = 2) have quite different profiles to their mass from August to December. Males seem to be losing mass, while females seem to be gaining mass.



The two sexes are no different in mass and there is no difference between sites A and B



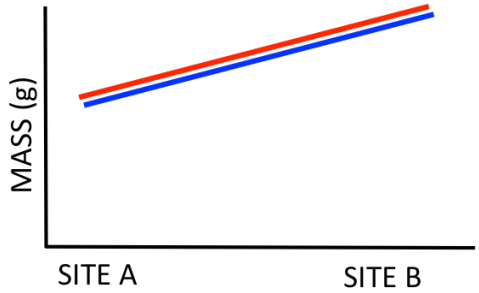
Testing MASS as a response to SEX and SITE

Female

Male

SEX	$P > 0.05$
SITE	$P > 0.05$
SEX×SITE	$P > 0.05$

The two sexes are no different in mass but there is a difference between sites A and B



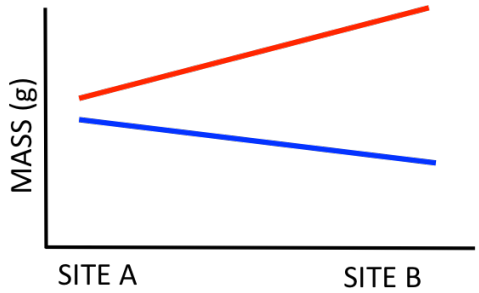
Testing MASS as a response to SEX and SITE

Female

Male

SEX	$P > 0.05$
SITE	**$P < 0.05$**
SEX×SITE	$P > 0.05$

The two sexes are different in mass, and there is a difference between sites A and B, but males and females are responding differently



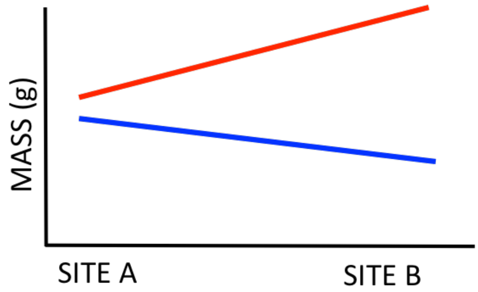
Testing MASS as a response to SEX and SITE

Female

Male

SEX	**$P < 0.05$**
SITE	**$P < 0.05$**
SEX×SITE	**$P < 0.05$**

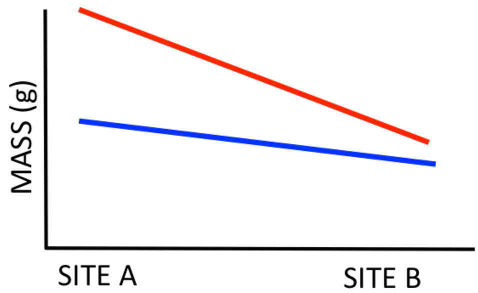
Note how the following examples have similar statistical results but the relationships are markedly different…



Testing MASS as a response to SEX and SITE

- Female
- Male

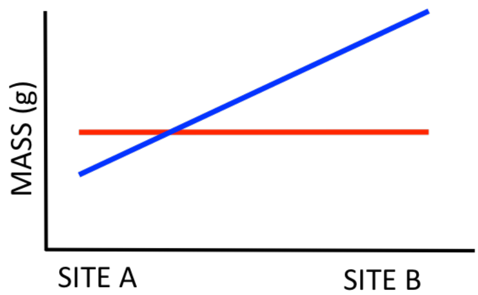
SEX	$P < 0.05$
SITE	$P < 0.05$
SEX×SITE	$P < 0.05$



Testing MASS as a response to SEX and SITE

- Female
- Male

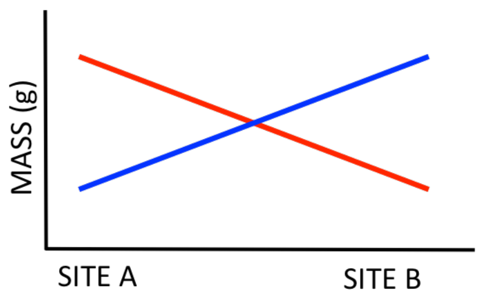
SEX	$P < 0.05$
SITE	$P < 0.05$
SEX×SITE	$P < 0.05$



Testing MASS as a response to SEX and SITE

- Female
- Male

SEX	$P < 0.05$
SITE	$P < 0.05$
SEX×SITE	$P < 0.05$



Testing MASS as a response to SEX and SITE

- Female
- Male

SEX	$P > 0.05$
SITE	$P > 0.05$
SEX×SITE	$P < 0.05$

Interaction Terms: Step-by-step

When you run any sort of linear model:

(1) Start by including all interaction terms
(2) Remove the interaction term that has the highest *P* value
(3) Run the model again… check interactions
(4) Remove another interaction term with a high *P* value
(5) Keep removing interaction terms until all you are left with are main effects and significant interactions

Why do we remove non-significant interactions from the model?

- Non-significant interaction terms interfere with the over-all predictive power of the model.
- By including them you are in effect telling the model to take into account an interaction and consider it important when it may not be
- When you remove non-significant terms, you will find the other *P*-values will change (slightly)
- But actually, it's usually no terrible problem leaving them in, and some researchers prefer to leave interactions in a model to show that they remembered to check them. Also, removing interaction terms can look like chasing significance (as P values will tend to decrease), but if significance depends on removing an interaction term, you need to seriously think about what that might mean in terms of overall effect. Relevant effect sizes will be an important next step.

Do we ever retain non-significant interaction terms in a linear model?

- Non-significant interaction terms are usually retained if they were part of your hypothesis.
    - For example, if your hypothesis specifically states *Male guppy preference for larger females depends on the temperature of the water* then the interaction term **`FEMALE.SIZE:WATER.TEMPERATURE`** is part of your experimental design and you would retain it in the model even if it is non-significant.
- Sometimes non-significant interaction terms are retained when higher order terms including the same variables are significant. So, if you find that the two-way interaction term **`AGE:SEX`** is not significant but the three-way interaction term **`AGE:SEX:HEALTH`** is significant then there is an argument for retaining the non-significant lower order interactions.
- When sample size is large (n = thousands of samples) the retention or removal of interaction terms makes less difference to the main effects and some researchers chose to leave the non-significant terms in the model. This last point is largely one of personal preference.

Exploring Interaction Terms in Linear Models

Johnson-Neyman Procedure

The Johnson-Neyman Procedure is a method for identifying the range over which two groups differ for a given response.

ASSUMPTIONS OF JOHNSON-NEYMAN

(1) Your model has already met the assumptions of a linear model
(2) There is a significant interaction term
(3) The response is continuous
(4) The interaction term is two-way or three-way (only 2 or 3 terms)
(5) The predictors need to be either numeric or sensibly coded as numeric dummy variables (i.e. male = 0, female = 1).

In some ways, the Johnson-Neyman procedure is quite limited in scope (there are quite a few limitations, but when it is applicable, it can be hugely useful for resolving and unravelling exactly how two terms are interacting with respect to a response variable,

Let's use our swallows dataset again.

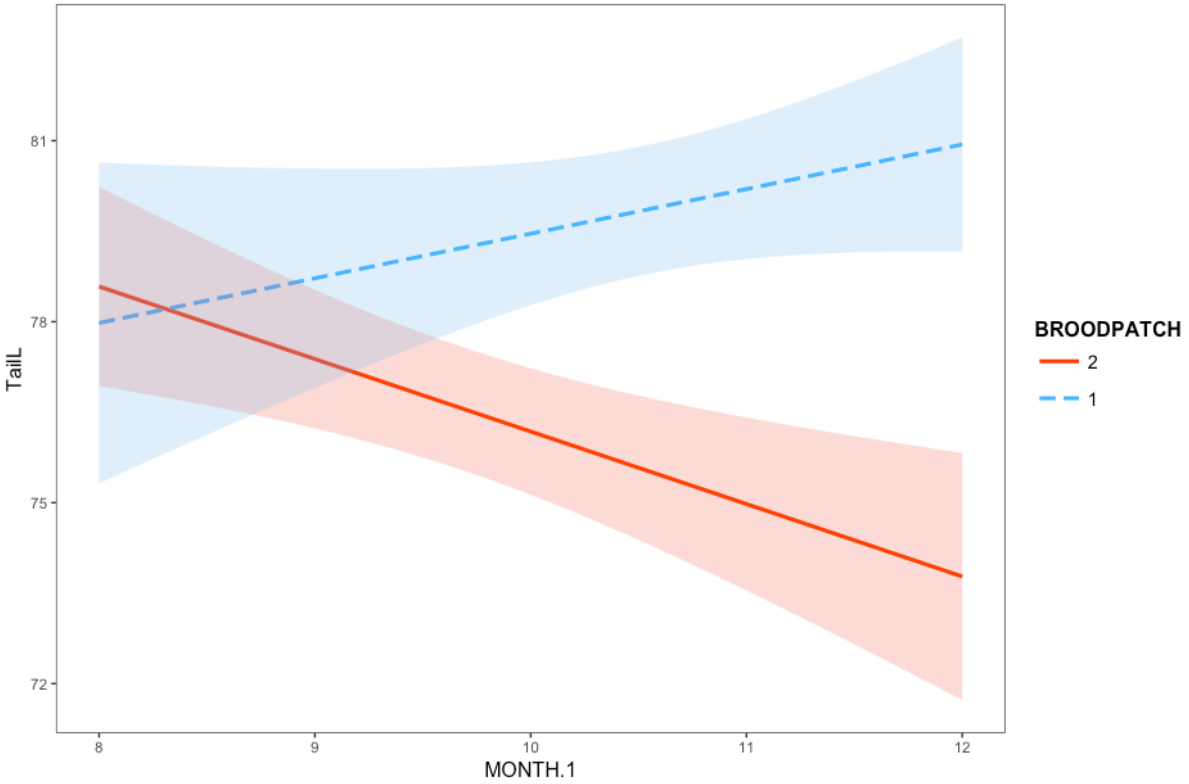
```
swallows.aov <- aov(TailL~MONTH.1*BROODPATCH,data=swallows)
summary(swallows.aov)
```

RESULT

```
                   Df Sum Sq Mean Sq F value  Pr(>F)
MONTH.1             1      0     0.0   0.000 0.98321
BROODPATCH          1    334   334.5   9.216 0.00301 **
MONTH.1:BROODPATCH  1    150   150.1   4.137 0.04444 *
Residuals         107   3884    36.3
```

So we have a significant interaction of MONTH:BROODPATCH for Tail Length (P = 0.0444). This suggests that the length of the tail is responding to month differently for the sexes. We'll make use of the visual and statistical tools in the interactions package. Let's start with an interaction plot.

```
library(interactions)
interact_plot(swallows.aov, pred = "MONTH.1", modx =
"BROODPATCH", interval = TRUE, int.width = 0.8)
```



That certainly looks like an interaction. It appears that females (Broodpatch = 2) have declining tail length from August to December, whereas males (Broodpatch = 1) have an increasing tail length. We can use the Johnson-Neyman Procedure to work out exactly for which months the difference is significant.

However, the Johnson-Neyman procedure won't work with an **aov** object. We need to create an **lm** object instead.

```
swallows.lm <- lm(TailL~MONTH.1*BROODPATCH,data=swallows)
summary(swallows.lm)
```

Note also, that if you have difficulty getting this procedure to work, you may need to use **as.numeric** to ensure that the predictors are numbers. You'd need to do this **before** building the model.

```
swallows$MONTH.1 <- as.numeric(swallows$MONTH.1)
swallows$BROODPATCH <- as.numeric(swallows$BROODPATCH)
```

The Johnson-Neyman functions are also in the interactions library, so you'll want to be sure it is loaded.

```
library(interactions)

sim_slopes(swallows.lm, pred = "BROODPATCH", modx = "MONTH.1",
johnson_neyman = TRUE)
```

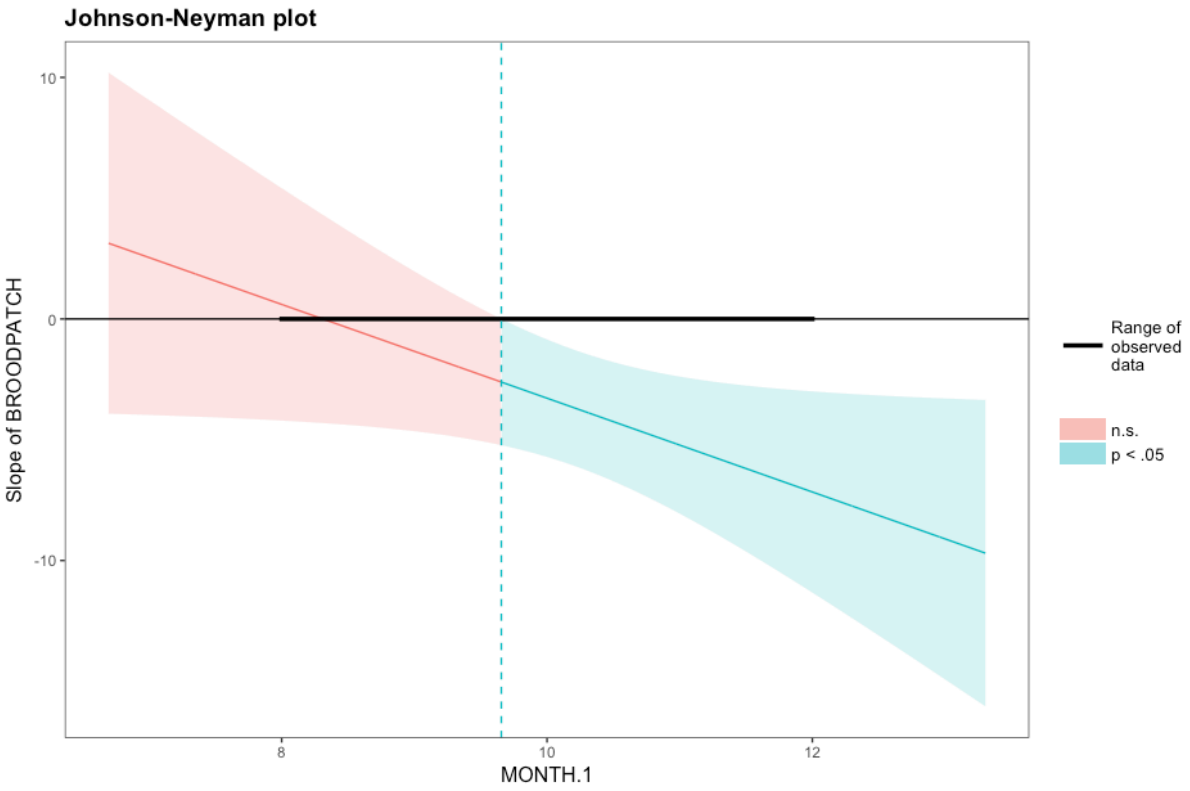
In this example, we are asking over what range of months is the slope of broodpatch (sex) different. Note that the predictor and moderator have been swapped here. You can try it the other way around, and see what result you obtain.

RESULT

```
JOHNSON-NEYMAN INTERVAL

When MONTH.1 is OUTSIDE the interval [-64.92, 9.66], the slope
of BROODPATCH is p < .05.

Note: The range of observed values of MONTH.1 is [8.00, 12.00]

SIMPLE SLOPES ANALYSIS

Slope of BROODPATCH when MONTH.1 = 11.39 (+ 1 SD):
  Est. S.E. t val.    p
 -5.99 1.66  -3.60 0.00

Slope of BROODPATCH when MONTH.1 = 10.09 (Mean):
  Est. S.E. t val.    p
 -3.45 1.21  -2.85 0.01

Slope of BROODPATCH when MONTH.1 =  8.79 (- 1 SD):
```

The interpretation would be that the two sexes have different slopes for the response of tail length to month before -64.92 (meaningless, there is no 'month' before zero), and after 9.66 (i.e. about half-way through September, the ninth month). We can also visualise this graphically:

```
johnson_neyman(swallows.lm, pred = "BROODPATCH", modx =
"MONTH.1", alpha = 0.05)
```



The thick black line shows you the actual range of the data. The red indicates the increments (here, months) over which the difference is not significant. The blue indicates the region in which there is significance. As with the result we had above, male and female adult swallows appear to have significantly different slopes of the relationship between month and tail length after mid-September, through to December.

Controlling for False Discovery Rate

The Johnson-Neyman Procedure in the interactions package allows you to control for false discovery rate. Try this and see whether the results differ to that above.

```
sim_slopes(swallows.lm, pred = "BROODPATCH", modx = "MONTH.1",
johnson_neyman = TRUE", control.fdr = TRUE)
```

Note that the result is now for the range inside two numbers, rather than outside a set of two numbers.

RESULT

```
JOHNSON-NEYMAN INTERVAL

When MONTH.1 is INSIDE the interval [9.78, 29.25], the slope
of BROODPATCH is p < .05.

Note: The range of observed values of MONTH.1 is [8.00, 12.00]

Interval calculated using false discovery rate adjusted t =
2.23

SIMPLE SLOPES ANALYSIS

Slope of BROODPATCH when MONTH.1 = 11.39 (+ 1 SD):
  Est. S.E. t val.    p
 -5.99 1.66  -3.60 0.00

Slope of BROODPATCH when MONTH.1 = 10.09 (Mean):
  Est. S.E. t val.    p
 -3.45 1.21  -2.85 0.01

Slope of BROODPATCH when MONTH.1 =  8.79 (- 1 SD):
  Est. S.E. t val.    p
 -0.92 1.81  -0.51 0.61
Slope of BROODPATCH when MONTH.1 =  8.79 (- 1 SD):
  Est. S.E. t val.    p
 -0.92 1.81  -0.51 0.61
```

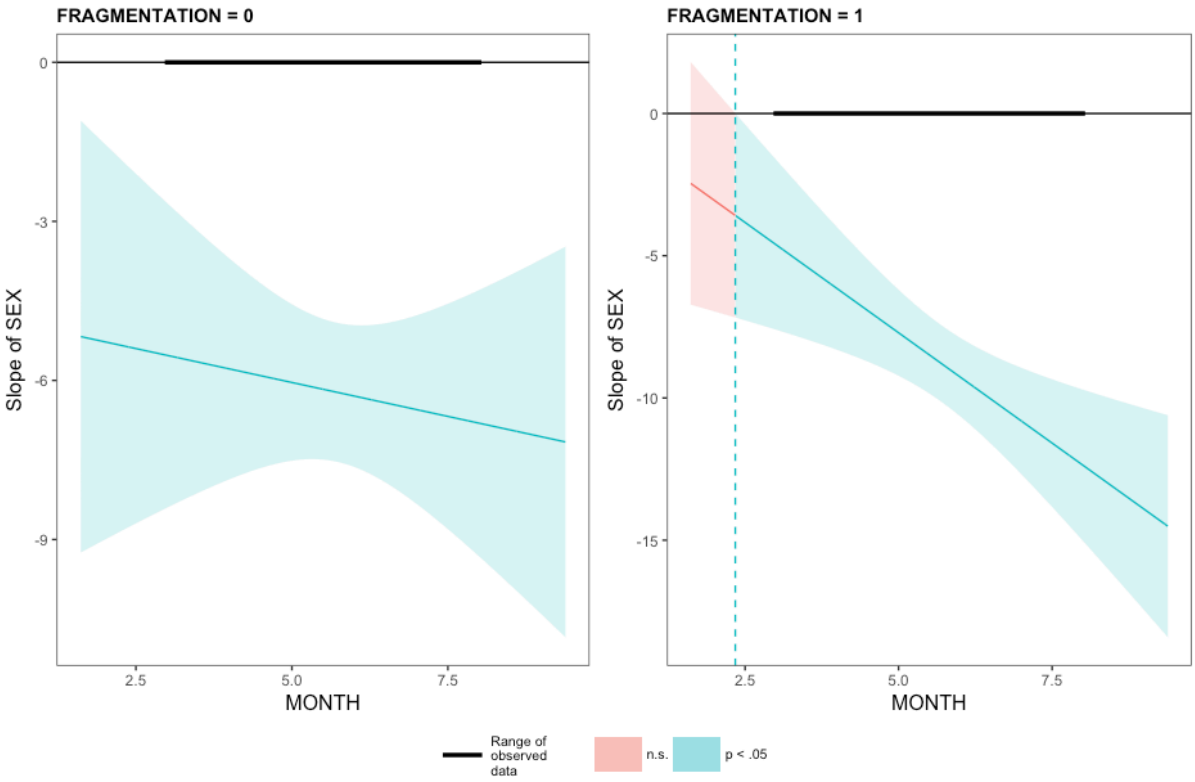
The interactions package does have some options for including a second moderator, although you will also need the library cowplot installed to make use of this. You can try this using the agilis_morphometrics.csv dataset.

```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')

install.packages("cowplot")
library(cowplot)

agilis.lm <- lm(MASS~SEX * FRAGMENTATION * MONTH , data=
agilis)
summary(agilis.lm)

sim_slopes(agilis.lm, pred = "SEX", modx = "MONTH", mod2 =
"FRAGMENTATION",jnplot = TRUE, control.fdr = TRUE)
```

RESULT

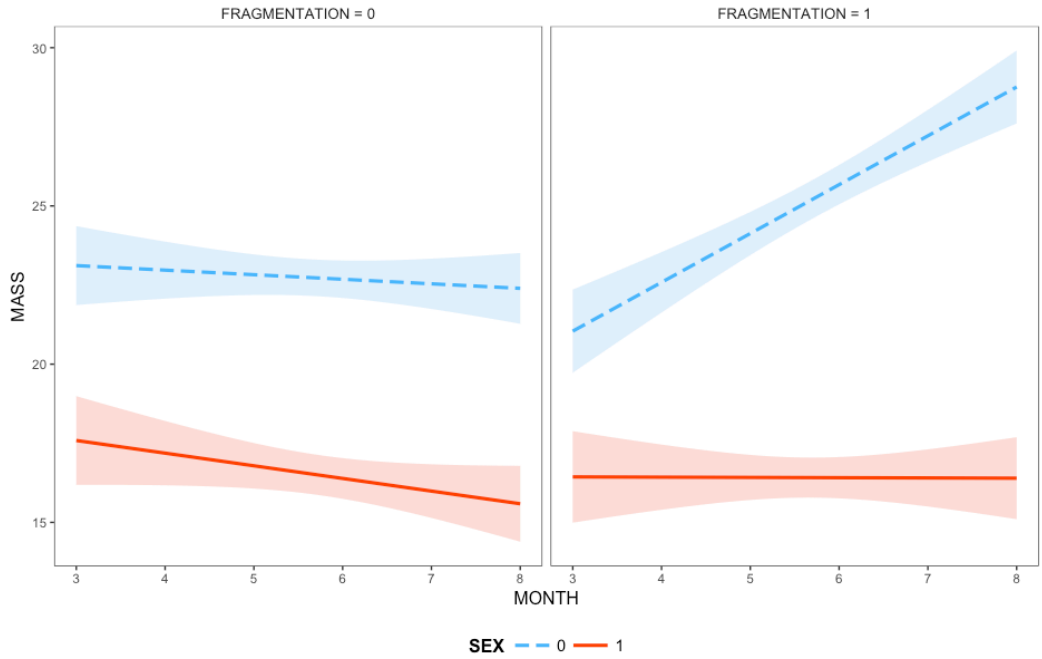
```
[redacted] While FRAGMENTATION (2nd moderator) = 0.00 (0) [redacted]

JOHNSON-NEYMAN INTERVAL

When MONTH is INSIDE the interval [0.66, 14.68], the slope of SEX is p <
.05.

Note: The range of observed values of MONTH is [3.00, 8.00]

Interval calculated using false discovery rate adjusted t = 1.97

SIMPLE SLOPES ANALYSIS

Slope of SEX when MONTH = 4.32 (- 1 SD):
  Est. S.E. t val.    p
 -5.87 0.94  -6.24 0.00

Slope of SEX when MONTH = 5.70 (Mean):
  Est. S.E. t val.    p
 -6.22 0.66  -9.36 0.00

Slope of SEX when MONTH = 7.09 (+ 1 SD):
  Est. S.E. t val.    p
 -6.57 0.93  -7.07 0.00

[redacted] While FRAGMENTATION (2nd moderator) = 1.00 (1) [redacted]

JOHNSON-NEYMAN INTERVAL

When MONTH is OUTSIDE the interval [-10.11, 2.34], the slope of SEX is p
< .05.

Note: The range of observed values of MONTH is [3.00, 8.00]

Interval calculated using false discovery rate adjusted t = 1.97

SIMPLE SLOPES ANALYSIS

Slope of SEX when MONTH = 4.32 (- 1 SD):
  Est. S.E. t val.    p
 -6.66 0.97  -6.87 0.00

Slope of SEX when MONTH = 5.70 (Mean):
  Est. S.E. t val.    p
 -8.80 0.68 -12.93 0.00

Slope of SEX when MONTH = 7.09 (+ 1 SD):
   Est. S.E. t val.    p
 -10.95 0.98 -11.19 0.00
```

In effect, the model has been split into fragmented (1)and continuous (0) populations, and then a difference for slope of mass by sex has been checked for each of the two populations seperately. The plot generated is on the next page, and you can see that the fragmented and continuous populations are graphed separately too.



There is a small region of non-significant difference in slopes (red) for fragmented populations, but if you look at the thick black line, you can see that this is outside our actual sampled range (March-August). We wouldn't want to infer anything about months outside of the sampling range anyway, so that's fine. Plotting the data rather than the slopes would be helpful for interpretation:

```
interact_plot(agilis.lm, pred = "MONTH", modx = "SEX", , mod2
= "FRAGMENTATION", interval = TRUE, int.width = 0.8)
```

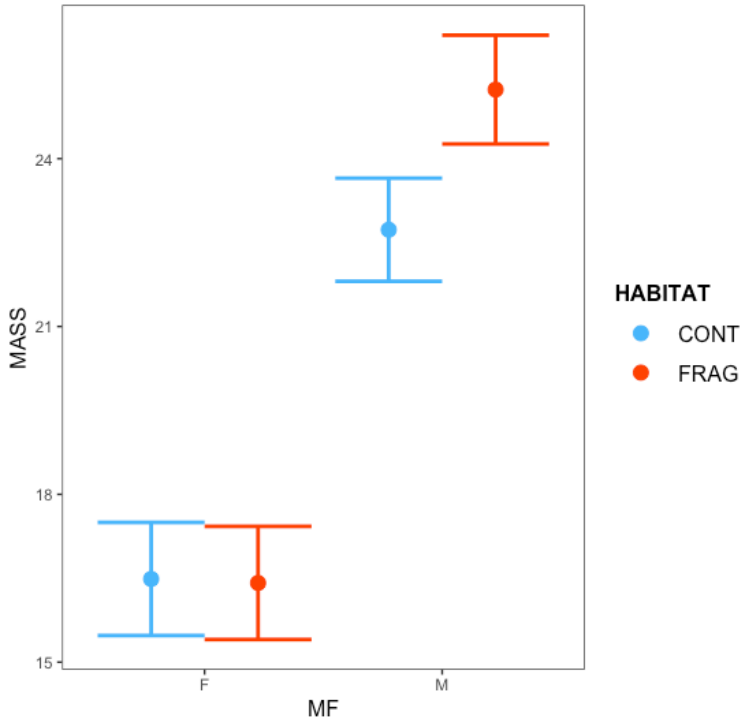


Fragmentation (0 = continuous forest, 1 = fragmented) Sex (0 = male, 1 = female)

The interactions package also allows you to examine a significant interaction between two categorical predictors (factors) using a categorical interaction plot.

We will use the agilis_morphometrics.csv dataset again. Note that I am using 'aov' instead of 'lm' because 'aov' is considered preferable when your predictors of interest are categorical.

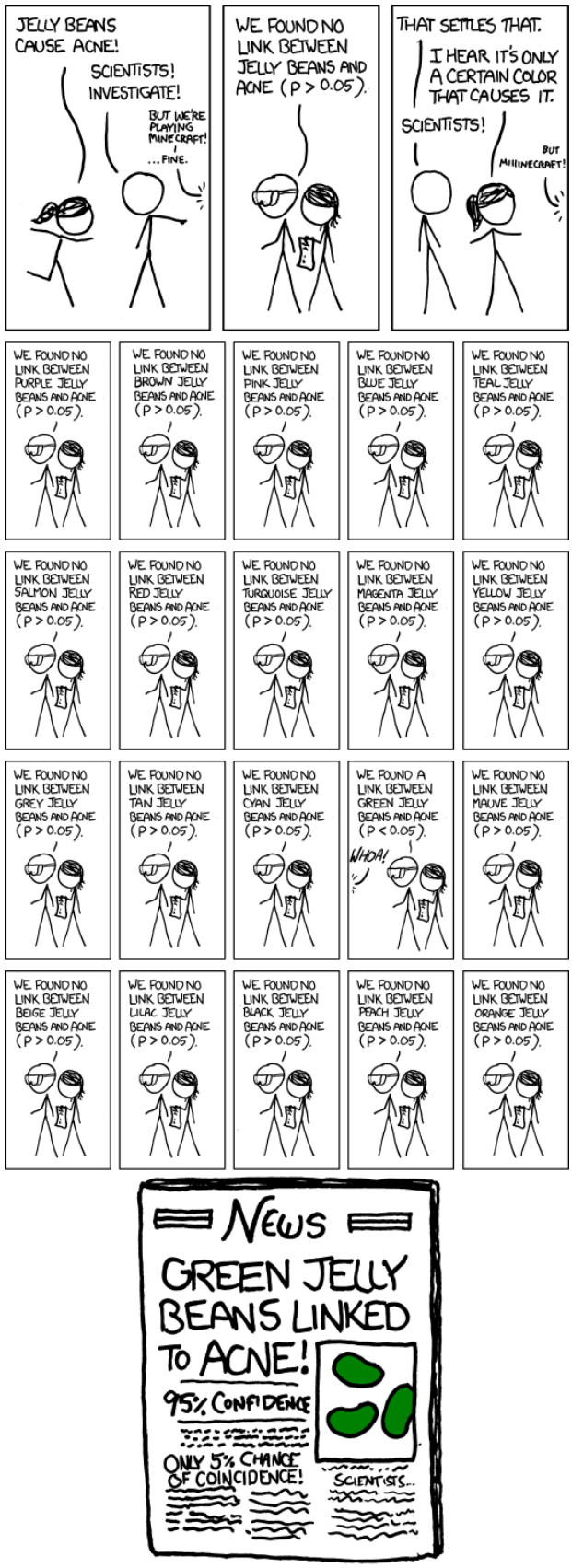
```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')

agilis.aov <- aov(MASS~MF * HABITAT, data= agilis)
summary(agilis.aov)

library(interactions)
cat_plot(agilis.aov, pred = "MF", modx = "HABITAT", data =
agilis)
```



Our interpretation would be that there is no difference in mass for female agile antechinus in the two environments (fragmented and continuous), but male agile antechinus are heavier in the fragmented habitat. We can draw this as a statistically valid conclusion because the bars indicate 95% confidence intervals (i.e. not standard errors), so if bars do not overlap, then the means are different at $P < 0.05$. However, do note that no 'pairwise' correction has been applied, and it might be sensible to try applying a Tukey's test to the interaction term as well. Which brings us to the next topic.

Post-hoc multiple comparisons

A *post-hoc* test is applied after the initial hypothesis test has been conducted. If the initial hypothesis test shows a significant effect of a factor, then *post-hoc* tests can help disentangle exactly what the effect may be. One important feature of *post-hoc* tests is that they adjust the *P*-value to take into account the number of comparisons being made. Because each comparison (using classically statistical methods) runs a 0.05% chance of returning a false positive, testing differences in groups by applying endless sequences of t-tests (for example) runs an increasingly high risk of returning a false significant result. The XKCD comic illustrates this quite clearly.



Tukey's Test

If you only have two levels in a factor (i.e. as when you are doing a *t*-test) there is no good reason to apply a *post-hoc* test. If there is a significant difference with just two levels you simply need to graph the data and check which of the two means is higher than the other. You already know the difference is significant because that was what your initial hypothesis test informed you.

Where there are **three or more levels of a significant factor**, then you need to undertake a post-hoc test to determine which groups have different sample means and which have the same sample means.

ASSUMPTIONS OF TUKEY TEST

(1) Equal variances
(2) Observations must be independent (collected randomly)
(3) You are testing the effect(s) of a factor on a response variable
(4) The factor has already been shown to be significant
(5) The factor has 3 or more levels

Import the agilis morphometrics dataset (if you haven't already):

```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')
```

Check the data:

```
head(agilis)
str(agilis)
```

Run the following:

1) Turn month into a factor (it is currently a number)

```
agilis$MONTH <- as.factor(agilis$MONTH)
```

2) Create an ANOVA model

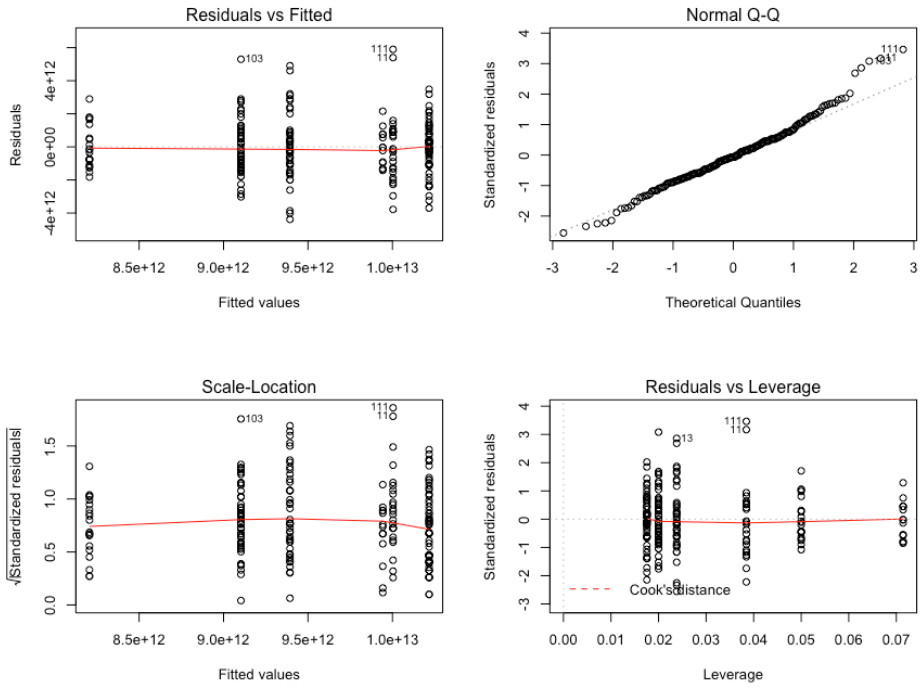
```
agilis.aov <- aov(RBC~MONTH,data=agilis)
```

3) Check the diagnostic plots. Do they look okay?

```
par(mfrow = c(2,2)) # set plotting window to 2x2 array
plot(agilis.aov)
```

4) Look at the results of the ANOVA.

```
summary(agilis.aov)
etaSquared(agilis.aov) # in library 'lsr'
```

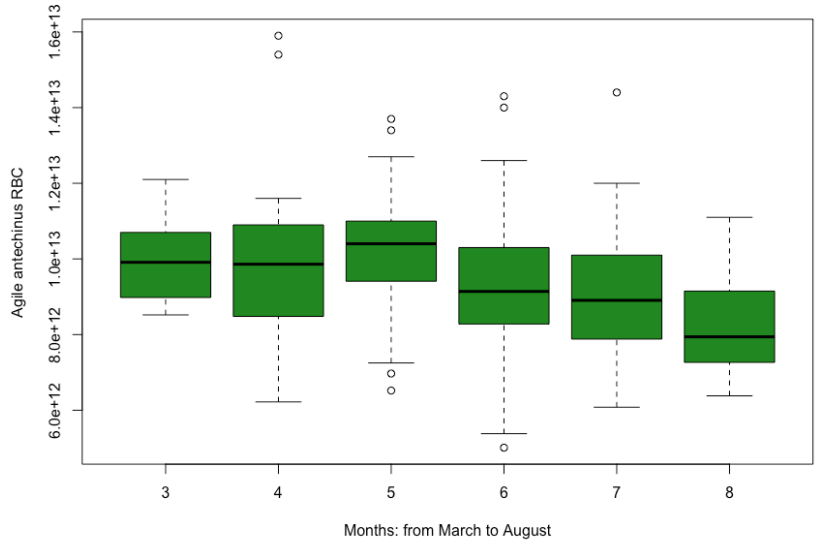


RESULT

```
> summary(agilis.aov)
             Df    Sum Sq   Mean Sq F value   Pr(>F)
MONTH         5 8.002e+25 1.600e+25   5.306 0.000132 ***
Residuals   203 6.122e+26 3.016e+24

> etaSquared(agilis.aov)
         eta.sq    eta.sq.part
MONTH 0.1155909      0.1155909
```

This result has informed us that there is some sort of difference by month for agile antechinus red blood cell counts, but we don't know which months are different to which other months. Because seasonal effects are often (although not always) non-linear because of the cyclical nature of the year, it is often better to view months as factors rather than numbers (at least at the more basic level of ANOVAs... more sophisticated tests for cyclical data do exist but we're not looking at them at this point). Let's have a look at the boxplots and a post-hoc comparison.

```
par(mfrow = c(1,1)) # set plotting window to 1x1 array
boxplot(RBC~MONTH,data=agilis,col=c("forestgreen"), ylab = "Agile
antechinus RBC", xlab = "Months: from March to August")
```




```
TukeyHSD(agilis.aov)
```

RESULT

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = RBC ~ MONTH, data = agilis)

$MONTH
              diff            lwr            upr     p adj
4-3  6.060440e+10 -1.595730e+12 1716938832527 0.9999982
5-3  2.741604e+11 -1.216218e+12 1764538487050 0.9949416
6-3 -5.497619e+11 -2.091725e+12  992201539901 0.9088421
7-3 -8.398571e+11 -2.350667e+12  670952313723 0.5999092
8-3 -1.736857e+12 -3.477979e+12    4264726072 0.0509691
5-4  2.135560e+11 -9.688964e+11 1396008418516 0.9953595
6-4 -6.103663e+11 -1.857210e+12  636476967489 0.7219131
7-4 -9.004615e+11 -2.108564e+12  307641140953 0.2688956
8-4 -1.797462e+12 -3.283554e+12 -311368596421 0.0079700
6-5 -8.239223e+11 -1.839994e+12  192149151876 0.1857561
7-5 -1.114018e+12 -2.082158e+12 -145877493844 0.0138576
8-5 -2.011018e+12 -3.309576e+12 -712458609957 0.0001986
7-6 -2.900952e+11 -1.335906e+12  755715252267 0.9675809
8-6 -1.187095e+12 -2.544549e+12  170358160783 0.1242693
8-7 -8.970000e+11 -2.218958e+12  424958274507 0.3736712
```

How do we interpret the output of the Tukey's test? The test is running all possible 'pairwise' comparisons for the mean of RBC by month, and adjusting to control for the 'familywise' error rate. The problem with running many tests is that the risk of a Type I error (rejecting the null when we shouldn't due to a chance event) increases. The Tukey's test penalises the P values by (in effect) multiplying P values by the number of comparisons, topping out at P > 0.999. Let's look at one line and interpret it:

```
             diff           lwr           upr     p adj
4-3  6.060440e+10 -1.595730e+12 1716938832527 0.9999982
```

`4-3` The mean of April minus the mean of March

`diff` Is a difference of...
`6.060440e+10` $6x10^{10}$ cells per L

The 95% lower confidence limit for this difference is `-1.595730e+12` RBC/L whereas the 95% upper confidence limit is `1716938832527` RBC/L. Note that the confidence interval crosses zero. This means that within a 95% level of confidence, the difference between the two means **could be zero**.

The adjusted P value for this difference is `0.9999982`, which suggests the difference is not significant (i.e. we have a 95% level of confidence that the difference in means may be zero: a P value is a CI 'flipped around' the other way).

How to report a Tukey's Test

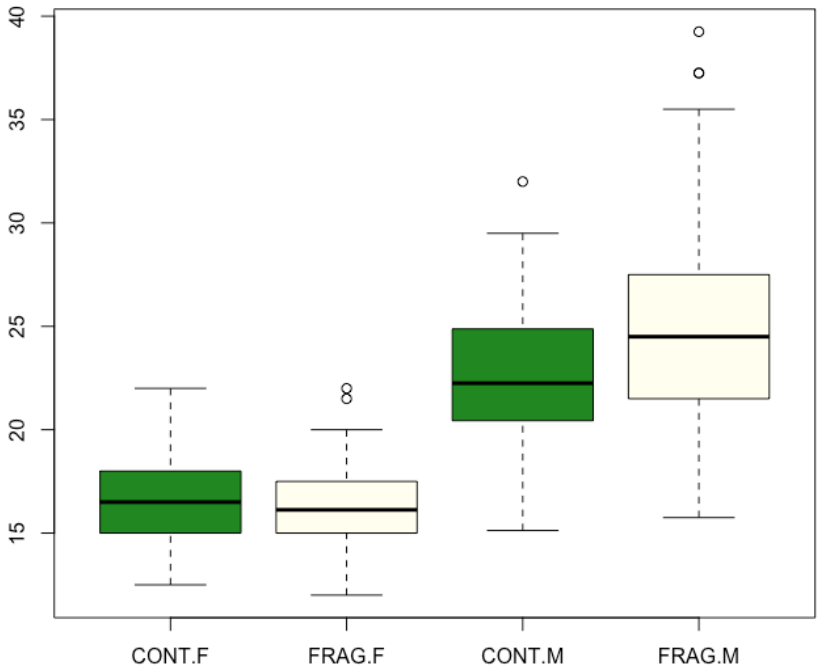
Typically, Tukey's contrasts are reported in a table. The **Difference** and the **Adjusted P-value** should always be reported. The confidence intervals should also be reported for preference, but space allowances in journals sometimes restrict this. You can also use equal signs like this (March = April = May = June: P > 0.05 for all contrasts), or does not equal signs like this (August ≠ April: P < 0.05), but you have to be very careful as you write out the equals and not-equals, as this can be confusing, and easy to mix up. Tukey's tests are also reported using **alphabet soup** on graphs, and we'll look at an example of this now.

Incidentally, you can plot a Tukey's test, but the plot only gives you the confidence intervals, which are **usually not reported in graphical form for a Tukey's test...**

```
agilis.tukey <- TukeyHSD(agilis.aov)
plot(agilis.tukey)
```

The following boxplot is a visual representation of a two-way ANOVA involving sex (**MF**) and fragmentation of habitat (**HABITAT**).

```
boxplot(MASS~HABITAT*MF,data=agilis,col=c("forestgreen","ivory"))
```




```
boxplot(MASS~HABITAT*MF,data=agilis
```

Run the following:

1) Make sure your factors are definitely factors:

```
agilis$HABITAT <- as.factor(agilis$HABITAT)
agilis$MF <- as.factor(agilis$MF)
```

2) Create an ANOVA model

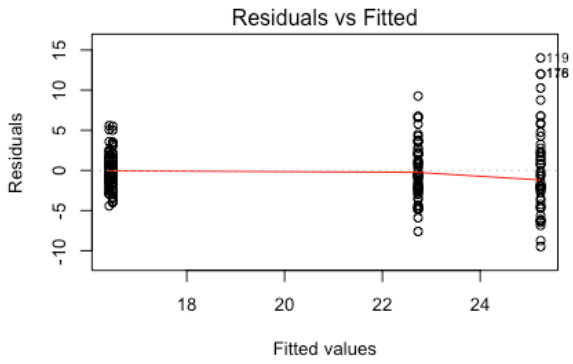
```
agilis.aov <- aov(MASS~HABITAT*MF,data=agilis)
```

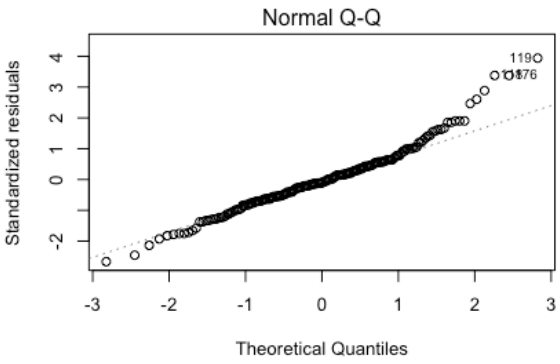
3) Check the diagnostic plots. Do they look okay?

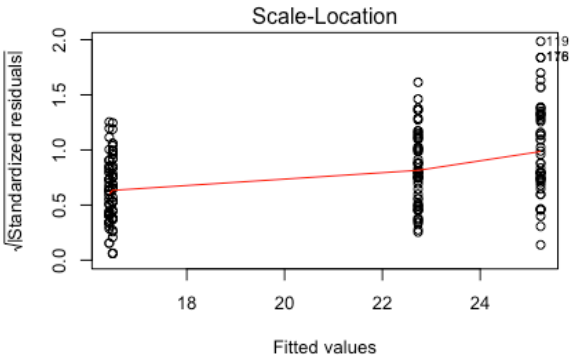
```
par(mfrow = c(2,2)) # set plotting window to 2x2 array
plot(agilis.aov)
```

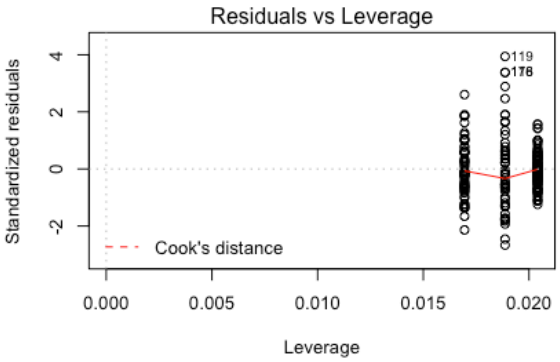
4) Look at the results of the ANOVA. Check the interactions. Is the interaction term `HABITAT:MF` significant?

```
summary(agilis.aov)
etaSquared(agilis.aov) # in library 'lsr'
```









RESULT

```
> summary(agilis.aov)
             Df Sum Sq Mean Sq F value Pr(>F)
HABITAT       1   63.6    63.6   4.935 0.0274 *
MF            1 2937.3  2937.3 227.863 <2e-16 ***
HABITAT:MF    1   86.6    86.6   6.722 0.0102 *
Residuals   206 2655.5    12.9


> etaSquared(agilis.aov) # in library 'lsr'
              eta.sq       eta.sq.part
HABITAT     0.01545856   0.03235094
MF          0.51145392   0.52519565
HABITAT:MF  0.01508764   0.03159923
```

Because the interaction term was significant, we shouldn't remove it from the model. As long as both terms are factors (remembering that Tukey's tests won't work for covariate predictors as there are no levels to contrast), we will get all contrasts including for the interaction levels:

```
TukeyHSD(agilis.aov)
```

RESULT

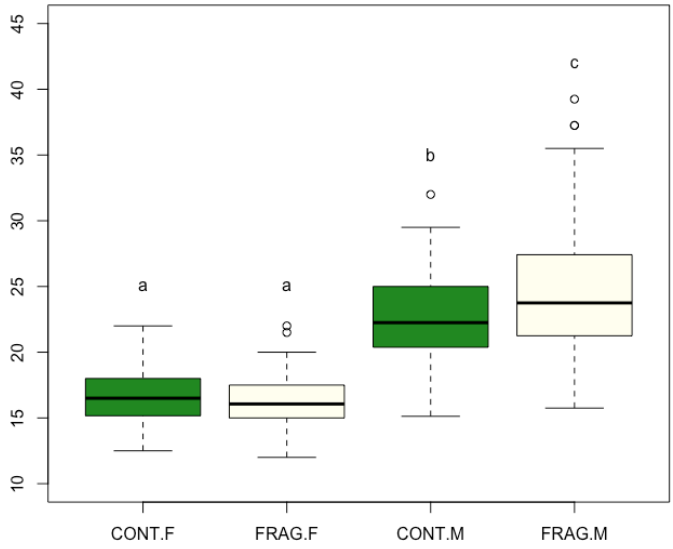
```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = MASS ~ HABITAT * MF, data = agilis)

$HABITAT
              diff       lwr      upr    p adj
FRAG-CONT 1.101247 0.1239175 2.078576 0.027404

$MF
        diff      lwr      upr p adj
M-F 7.493854 6.514746 8.472963     0

$`HABITAT:MF`
                       diff        lwr       upr     p adj
FRAG:F-CONT:F -0.07108844 -1.9498588  1.807682 0.9996619
CONT:M-CONT:F  6.24304931  4.4456499  8.040449 0.0000000
FRAG:M-CONT:F  8.74881273  6.9058317 10.591794 0.0000000
CONT:M-FRAG:F  6.31413775  4.5167384  8.111537 0.0000000
FRAG:M-FRAG:F  8.81990117  6.9769201 10.662882 0.0000000
FRAG:M-CONT:M  2.50576342  0.7458074  4.265719 0.0016372
```

So all contrasts are significant, except for female fragmented and female continuous forest populations. Using equal signs this would look like so: (FF = CF (P > 0.05) & CM ≠ FM ≠ FF (P < 0.05) & CM ≠ FM ≠ CF (P < 0.05)). If we were to allocate 'alphabet soup' to these contrasts we would need to add a letter to be shared by FF and CF (because they are not different), and add letters to distinguish all other groups. Something like this would do: FF_a, CF_a,, FM_b, CM_c. Using this notation method any groups that **share a letter** are **not significantly different**.

Alphabet Soup on a Boxplot

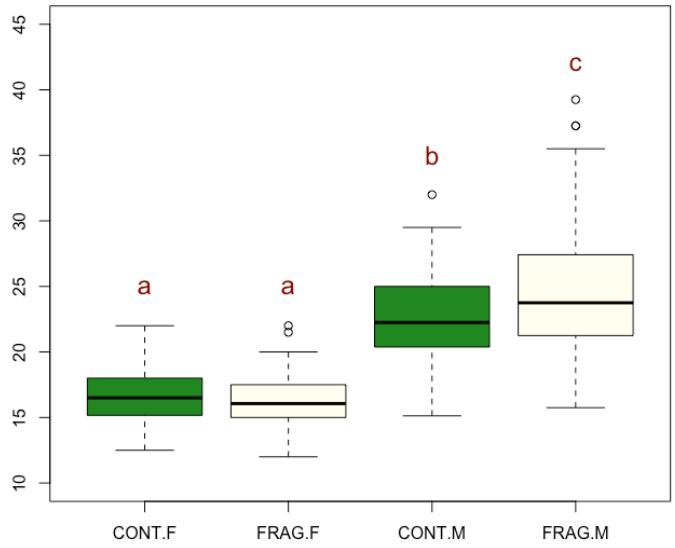
Let's use the boxplot we created above.

```
boxplot(MASS~HABITAT*MF,data=agilis,ylim=c(10,45),
col=c("forestgreen","ivory"))

text(c(1, 2, 3, 4), c(25, 25, 35, 42), labels=c('a', 'a', 'b', 'c'))
```



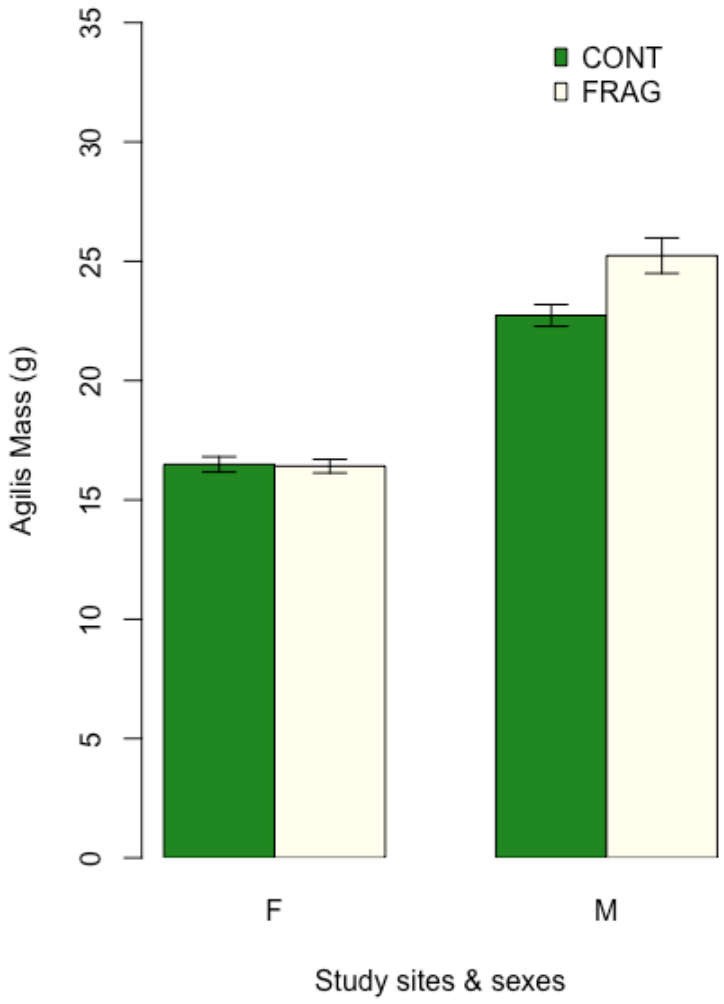
Use **cex** to increase font size and change the colour to dark red.

```
boxplot(MASS~HABITAT*MF,data=agilis,ylim=c(10,45),
col=c("forestgreen","ivory"))

text(c(1, 2, 3, 4), c(25, 25, 35, 42), labels=c('a', 'a', 'b', 'c'),
cex=1.5, col="darkred")
```



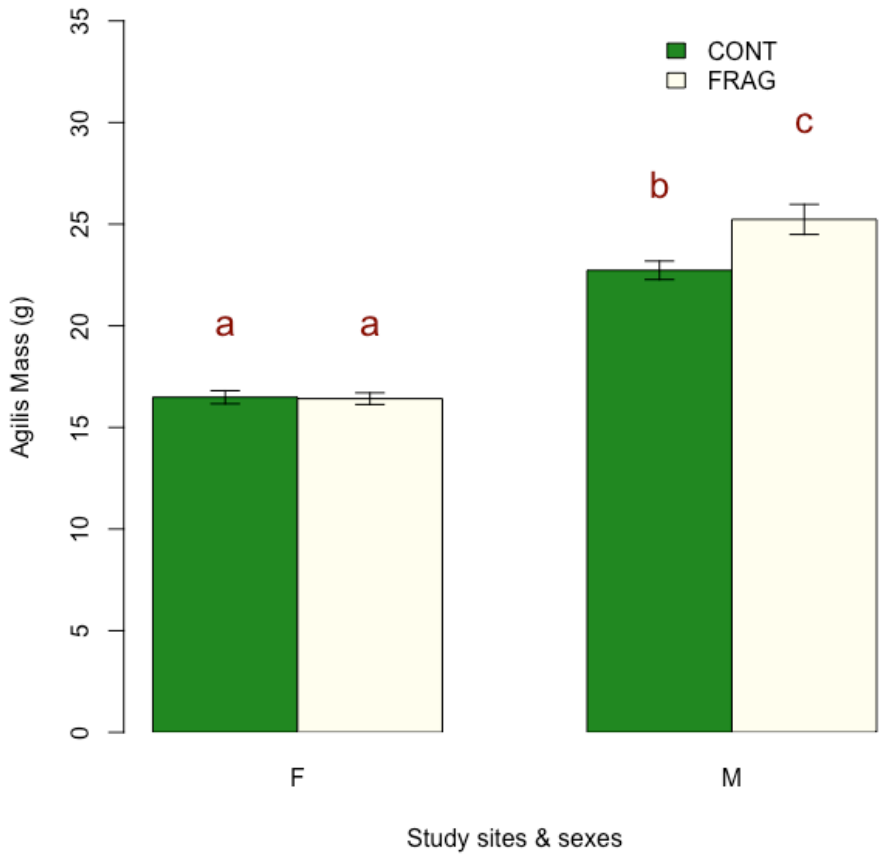
Because an ANOVA is a comparison of means (not medians), arguably, it is more suitable to present a bar graph with standard errors as confidence intervals. Some reviewers will insist on this. Others will be happy with boxplots. The library 'sciplot' allows you to plot a bar graph relatively easily if you need to.

```
library(sciplot)

attach(agilis)
bargraph.CI(
  MF, # categorical factor for the x-axis
  MASS, # numerical DV for the y-axis
  HABITAT, # grouping factor
  legend=T, # Use legend=F if you don't want a legend
  ylab="Agilis Mass (g)",
  xlab="Study sites & sexes",
  col=c("forestgreen","ivory"),
  ylim=c(0,35)) # set y axis to 0 to 35
```



Can we add alphabet soup to this graph? We can, as follows:

```
attach(agilis)
bargraph.CI(MF, MASS, HABITAT, legend=T, ylab="Agilis Mass
(g)", xlab="Study sites & sexes",
col=c("forestgreen","ivory"), ylim=c(0,35))

text(c(1.5, 2.5,4.5,5.5), c(20, 20, 27, 30),
labels=c('a', 'a', 'b', 'c'), cex=1.5, col="darkred")
```



Using 'glht' to apply Tukey's contrasts

The `TukeyHSD` command used above will only work on `aov` models. The test won't work on other types of linear models such as linear mixed effects (`lme` or `lme4`) or generalised linear mixed effects models (`glm`). Instead you can use the '`glht`' function in library '`multcomp` '. This same function also works on `aov` models, and we will use our dataset from above and an `aov` model above as an example:

```
install.packages("multcomp") # if not already installed
library(multcomp)
```

Make sure your 'factor' of interest is actually a factor in R!

```
your.data$your.factor <- as.factor(your.data$your.factor)
```

Make sure your dataset is attached!

```
attach(your.data)
```

```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')
```

Check the data:

```
head(agilis)
str(agilis)
```

1) Turn month into a factor (it is currently a number)

```
agilis$MONTH <- as.factor(agilis$MONTH)
```

2) Attach your dataset

```
attach(agilis)
```

3) Create an ANOVA model

```
agilis.aov <- aov(RBC~MONTH,data=agilis)
```

Apply a Tukey test to the model (example code):

```
fit.glht <- glht(your.model,linfct=mcp(YOUR.FACTOR="Tukey"))
```

The actual code for our agilis example:

```
agilis.glht <- glht(agilis.aov,linfct=mcp(MONTH="Tukey"))
```

```
plot(agilis.glht)
```

This is a plot of the confidence intervals of the contrasts. This plot is typically **not** included in scientific reports, but can be useful for you to look at.

```
summary(agilis.glht)
```

RESULT

```
      Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: aov(formula = RBC ~ MONTH, data = agilis)

Linear Hypotheses:
             Estimate Std. Error t value Pr(>|t|)
4 - 3 == 0  6.060e+10  5.757e+11   0.105  1.00000
5 - 3 == 0  2.742e+11  5.180e+11   0.529  0.99467
6 - 3 == 0 -5.498e+11  5.359e+11  -1.026  0.90504
7 - 3 == 0 -8.399e+11  5.251e+11  -1.599  0.58978
8 - 3 == 0 -1.737e+12  6.052e+11  -2.870  0.04861 *
5 - 4 == 0  2.136e+11  4.110e+11   0.520  0.99511
6 - 4 == 0 -6.104e+11  4.334e+11  -1.408  0.71329
7 - 4 == 0 -9.005e+11  4.199e+11  -2.144  0.26022
8 - 4 == 0 -1.797e+12  5.165e+11  -3.480  0.00754 **
6 - 5 == 0 -8.239e+11  3.532e+11  -2.333  0.17888
7 - 5 == 0 -1.114e+12  3.365e+11  -3.311  0.01306 *
8 - 5 == 0 -2.011e+12  4.513e+11  -4.456  < 0.001 ***
7 - 6 == 0 -2.901e+11  3.635e+11  -0.798  0.96604
8 - 6 == 0 -1.187e+12  4.718e+11  -2.516  0.11935
8 - 7 == 0 -8.970e+11  4.595e+11  -1.952  0.36360
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1
(Adjusted p values reported -- single-step method)
```

The results should broadly agree with the `TukeyHSD` example we used earlier (although may not be identical).

```
cld(agilis.glht)
```

This command automatically generates your alphabet soup letters for graphs and figures.

RESULT

```
      > cld(agilis.glht)
   3    4    5    6    7    8
"bc" "bc"  "c" "ac" "ab"  "a"
```

We would interpret this to mean that RBC in March is not significantly different to April, May or June. April is not significantly different to March, May and June. May is not different to March, April and June. June is not significantly different to any month. July is only significantly different to March. August is significantly different to March, April and May, but is not different to June and July.

glht: factorial interaction terms

If the model has an interaction term in it that involves your factor of interest and another factor, you may need to include an interaction average:

```
fit.glht <- glht(your.lme,linfct=mcp(YOUR.FACTOR="Tukey",
interaction_average=TRUE))

summary(fit.glht)
```

glht: covariate interaction terms

If the model has an interaction term that involves your factor of interest and a covariate in it, then you may need to include an interaction average:

```
fit.glht <- glht(your.lme,linfct=mcp(YOUR.FACTOR="Tukey",
covariate_average=TRUE))

summary(fit.glht)
```

Non-Parametric Equivalents of Variability Tests

Non-parametric tests do not assume that the data has any given distribution. These tests mean that data that is not normally distributed can be tested without resorting to transformations. Some researchers argue that we should always use non-parametric tests in the biological sciences because we don't actually know if increments of measured variables (such as mass) are isometric to outcomes of interest (such as survivorship and fitness) (i.e. you can't know that having twice the mass = exactly twice the fitness).

General non-parametric test for two groups

This non-parametric equivalent of a *t*-test is better known as a **Mann-Whitney *U***, but also called a **Wilcoxon rank sum test** or **Mann-Whitney-Wilcoxon test**. Non-parametric tests are suitable for data that is not normally distributed and data that is bounded, such as a percentage that is bounded by zero and one. Ratios can also be tested using non-parametric tests. In all cases, the non-parametric test avoids the need to transform the data ahead of time.

A Mann-Whitney *U* tests whether the **medians** are different. If you use a Mann-Whitney *U* test, you'll need to present data as box plots and medians with confidence intervals.

ASSUMPTIONS OF MANN-WHITNEY *U*

(1) The responses are ordinal (though not necessarily scaled)
(i.e. it is possible to say of two observations, which is larger)

(2) Observations must be independent (collected randomly)

We'll use the **same datasets** that we examined in the previous lab. This is so you can compare the results of the parametric and non-parametric equivalents. If you like, you can scroll back to the corresponding parametric test and check whether the *P*-values agree.

Import the data:

```
swallows <- read.table('swallows-adults.csv',header=T,sep=',')
head(swallows)
str(swallows)
```

Here's another way to check if BROODPATCH is a factor:

```
is.factor(swallows$BROODPATCH)
```

Change BROODPATCH to a factor:

```
swallows$BROODPATCH <- as.factor(swallows$BROODPATCH)
is.factor(swallows$BROODPATCH)
```

Apply the Mann-Whitney *U* test:

```
wilcox.test(MASS~BROODPATCH, data = swallows)
```

RESULT

```
     Wilcoxon rank sum test with continuity correction

data:  MASS by BROODPATCH
W = 819, p-value = 1.85e-05
alternative hypothesis: true location shift is not equal to 0
```

The implication is that there is a difference in the median mass (g) for sexes of swallows (as estimated by 'broodpatch').

General non-parametric paired test for two groups

A Wilcoxon signed-rank test can be used to test differences in paired observations. This is the non-parametric equivalent of a paired *t*-test

The Wilcoxon signed-rank test uses the same command as a Mann-Whitney *U* test in R. The data has to be set up differently in the csv file, but otherwise the commands are very similar to that given above.

ASSUMPTIONS OF WILCOXON SIGNED RANK

(1) The responses are on the same scale
(i.e. they are comparable, not enough just to be ordinal)
(2) Observations must be independent (collected randomly)
(3) Observations are paired (i.e. in blocks of two)

Import the paired mean seedling height data:

```
msh.p <- read.table('msh-paired.csv',header=T,sep=',')
str(msh.p)

wilcox.test(msh.p$TREATMENT, msh.p$CONTROL, paired=TRUE)
```

RESULT

```
        Wilcoxon signed rank test

data:  msh.p$TREATMENT and msh.p$CONTROL
V = 71, p-value = 0.009277
alternative hypothesis: true location shift is not equal to 0
```

The implication is that there is a difference in the median value of mean shrub heights (msh) in the paired plots that were studied.

General non-parametric test: one factor with multiple levels

We are now going to look at the non-parametric equivalent of a **one-way** ANOVA. We'll use the agilis data from the last lab because the RBC and Month model didn't perfectly fit the requirements for an ANOVA and it will be interesting to see if a non-parametric test gives a different answer to the test based on ANOVAs.

The Kruskal-Wallis test is a non-parametric equivalent of a one-way ANOVA.

ASSUMPTIONS OF KRUSKAL-WALLIS TEST

(1) Observations must be independent (collected randomly)

(2) One predictor factor (but can have many levels)

Import the dataset:

```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')
```

Check the data:

```
head(agilis)
str(agilis)
```

Turn month into a factor (it is currently a number)

```
agilis$MONTH <- as.factor(agilis$MONTH)
```

Apply a Kruskal-Wallis test:

```
kruskal.test(RBC~MONTH,data=agilis)
```

Note that we cannot add terms like `HABITAT+MONTH` or `HABITAT*MONTH` because a Kruskal-Wallis Test **only allows for one predictor** variable (i.e. it is a 'one-way' test).

RESULT

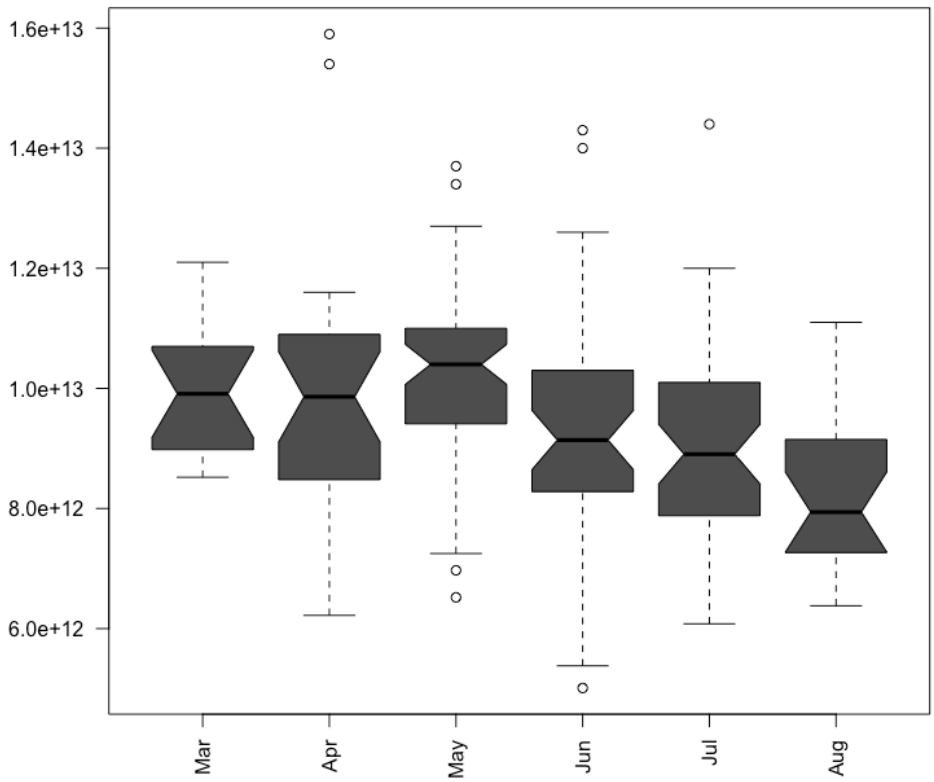
```
        Kruskal-Wallis rank sum test

data:  RBC by MONTH
Kruskal-Wallis chi-sqred = 29.89, df = 5, p-value = 1.55e-05
```

This suggests there is a significant difference in the RBC values by month for the agile antechinus studied. However, much like an ANOVA, we need to use a post-hoc test to identify which months are different to which other months. Technically, the Kruskal-Wallis test is testing for a difference in the overall distribution of values among groups, and we can't really say that significance indicates a difference in median or means. However, from a practical point of view, boxplots are usually presented with Kruskal-Wallis tests and we take significance to indicate a difference in the group 'central tendencies'.

If you wanted to check this against a visual test for differences in medians, boxplots can be set to have a 'notch'. This notch represents a (roughly) 95% CI for the median (actually the notch = `+/-1.58 IQR/sqrt(n)`). If two notches overlap, you can take this to suggest the medians are not different. If they do not overlap, then this can be taken to be 'strong evidence' that the medians are different.

Reorder `Month` so that it will appear in the right order on the boxplot (i.e. not alphabetically).

```
agilis$Month<-factor(agilis$Month, c("Mar", "Apr", "May",
"Jun", "Jul", "Aug"))

boxplot(RBC~Month, las = 2, data=agilis, notch=T,
col="grey30")
```



So, we could take the plot as evidence that Mar, Apr and May have equivalent RBC medians, but May is different to Jun, Jul and August. However, there is no adjustment for multiple comparisons here. It is possible to adjust the size of notches, but working out how to adjust for multiple comparisons is not straightforward (i.e. it wouldn't simply be a matter of applying a multiplier, and these notches are extremely rough anyway)

General non-parametric pairwise comparison

If you obtain a significant result for a Kruskal-Wallis test, and you have three or more levels in a given predictor factor, then you will need to perform a non-parametric equivilent of a Tukey's test. One good option is the Dunn's Test, a non-parametric pairwise comparison. The assumptions are the same as for a Kruskal-Wallis test.

For a Dunn's test:

```
install.package("dunn.test")
library(dunn.test)

attach(your.data)
dunn.test(RESPONSE, PREDICTOR.GROUP)
```

There are a number of different corrections you can apply. Bonferroni is the most standard, although Holm's correction is pretty widely used, and some people seem to prefer it...

```
attach(your.data)
dunn.test(RESPONSE, PREDICTOR.GROUP, method="bonferroni")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="sidak")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="holm")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="hs")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="hochberg")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="bh")
dunn.test(RESPONSE, PREDICTOR.GROUP, method="by")
```

We will use a Bonferroni correction for multiple comparison error:

```
attach(agilis)
dunn.test(RBC, MONTH, method="bonferroni")
```

The result output is in the form of a table with P values and differences next to each other. Each contrast has a difference between means (Col Mean - Row Mean) which is the top number, and a P value, which is the bottom number. P values 'top out' at 1.0000, although in a report you would probably write this as P > 0.999.

RESULT

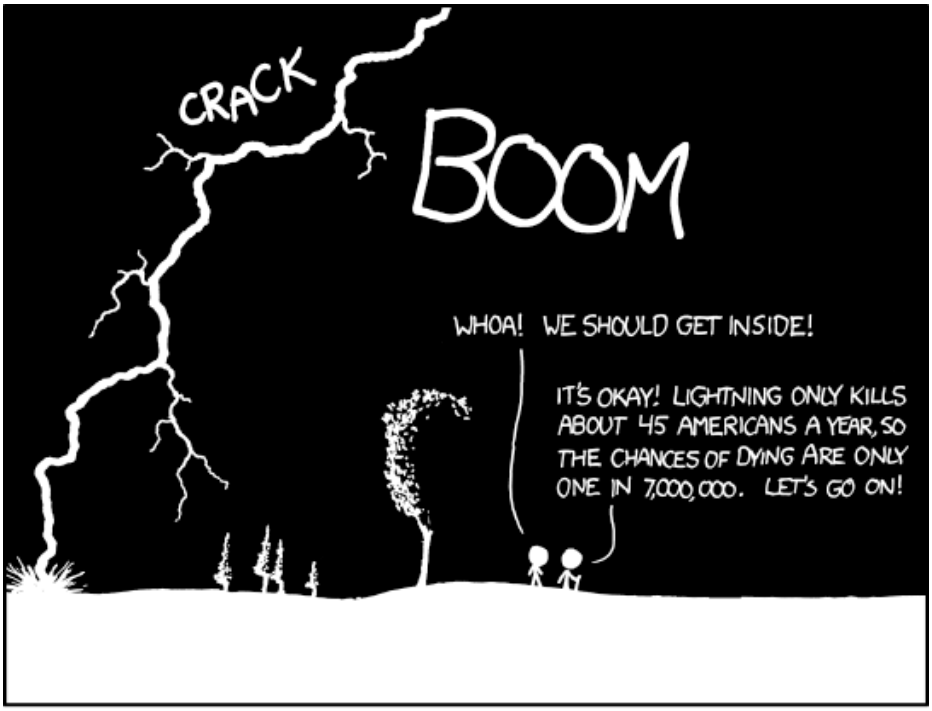
```
        Kruskal-Wallis rank sum test

data: RBC and MONTH
Kruskal-Wallis chi-squared = 29.8904, df = 5, p-value = 0


                          Comparison of RBC by MONTH
                                  (Bonferroni)
Col Mean-|
Row Mean |          3          4          5          6          7
---------+------------------------------------------------------------
       4 |   0.314788
         |     1.0000
         |
       5 |  -0.489010  -1.057298
         |     1.0000     1.0000
         |
       6 |   1.293663   1.181694   2.680513
         |     1.0000     1.0000     0.0551
         |
       7 |   1.874960   1.913177   3.678723   0.801222
         |     0.4560     0.4179    0.0018*     1.0000
         |
       8 |   3.114881   3.298577   4.737712   2.525763   1.959728
         |    0.0138*    0.0073*    0.0000*     0.0866     0.3752

alpha = 0.05
Reject Ho if p <= alpha/2
```

The March RBC was significantly higher than the August RBC (P = 0.014). The RBC in April was significantly higher than the RBC in August (P = 0.007). The RBC in May was significantly higher than the RBC in July (P = 0.002) and August (P < 0.001). If you are unsure which median value is higher, you can always just boxplot the data and check it that way. If you see agreement between the Dunn's Test and the notched boxplots, that would be quite strong evidence for a difference.

Appendices

Everything after this page is intended only to help you complete assignments in other units or during an honours year.

Nothing after this page is assessable for BIO3011.



THE ANNUAL DEATH RATE AMONG PEOPLE WHO KNOW THAT STATISTIC IS ONE IN SIX.

Generalised linear models

What if we want to examine count of binary response data in a more sophisticated way than chi squared tests allow? The mynas dataset we used earlier has various predictor variables, but because these are not all factors (i.e. cannot be used to generate a contingency table), chi-squared tests are not going to be much help.

The variables we have are:

```
'data.frame':     160 obs. of  7 variables:
$ REGION    : Factor w/ 4 levels "Peri-urban","Rural",..: 3 3 3 3 3 3 3 3 3 3 ...
$ FORAGE    : num  0.656 0.25 0.656 0.533 0.723 ...
$ VIGIL     : num  0.292 0.757 0.314 0.476 0.23 ...
$ PECK.rate: int  5 4 4 5 4 5 5 5 5 3 ...
$ HUMANS    : Factor w/ 2 levels "NO","YES": 2 1 1 1 2 2 2 1 2 1 ...
$ CONSPECS : Factor w/ 2 levels "NO","YES": 2 2 2 2 2 2 2 2 2 2 ...
$ VEHICLES : Factor w/ 2 levels "NO","YES": 2 1 1 2 2 1 1 2 2 2 ...
```

REGION	The type of landscape
FORAGE:	Percentage of time spent foraging
VIGIL	Percentage of time spent vigilant
PECK.rate	The response variable
HUMANS	Were humans (aside from the researcher) present? yes/no
CONSPECS	Were conspecifics present? yes/no
VEHICLES	Were vehicles present? yes/no

What we will do now is make use of **generalised linear models (glm)**. A generalised linear model is an advance over **general linear models (lm)** and **ANOVAs (aov)**, in that a generalised linear model can make use of some clever mathematics to model data using different distributions. The most commonly used are **Gaussian** (normal, the same as a **lm**), **Poisson** (when the response is count data) and **Binomial** (when the response is binary).

Generalised linear models use a "link" function to specify the distribution of residuals. For the most common two types of distribution used are:

Log	used for count data	$\eta = \log \mu$
Logistic (logit)	used for binary data	$\eta = \log (\mu / (1 - \mu))$

```
model.glm <- glm(y ~ x, family = binomial(link="logit"))
```

```
model.glm <- glm(y ~ x, family = poisson(link = "log"))
```

A generalised linear model is a test of **deviance** not **variance**. **Deviance** is a measure of fit, a bit like the other **goodness of fit** measures we've looked at today. It is a measure of how well your data fits a model

---

ASSUMPTIONS OF GENERALISED LINEAR MODEL

(1) Observations are independent (collected randomly)

(2) Correct link function is used

Residuals must fit the nominated error distribution

Overdispersion should not occur

These are the only two assumptions. If the residuals do not fit 'over-dispersion' will occur. Can check this by looking at the residual deviance:

- If the model does not fit, overdispersation occurs
- When this happens, the residual deviance will be larger than the degrees of freedom
- If this occurs, you may have to switch to a negativebionomial, quasibinomial (for binomial) or quasipoisson (for poisson) distribution and try again

---

PROBLEMS CAN OCCUR IF...

(1) Predictor variables are correlated

(2) Model is over-parameterised

(3) Covariates are bounded or not normally distributed

GLM: Poisson distribution

```
myna <- read.table('mynas_peck.csv',header=T,sep=',')
str(myna)
```

First things first. Let's check (roughly) for normality of our percentages and check they don't correlate. Anything with a correlation value less than -0.6 or greater than +0.6 may raise concerns.

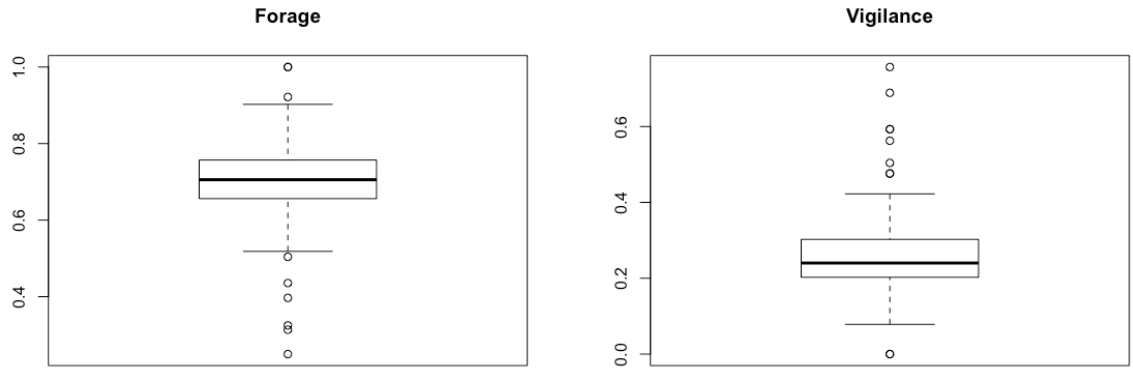
```
cor(myna$FORAGE, myna$VIGIL, method = "pearson")
```

RESULT

```
      [1] -0.9010008
```

There is definitely a potential problem here. -0.901 is strong negative correlation. In this case, why wouldn't we include both foraging or vigilance in the model? In a sense, these two correlated variables are measuring the same thing (probably an index of predator wariness). Including both variables as predictors will cause problems because they are attempting to explain the same variance (or because this is a glm, deviance) in the model. If both were included as predictors, we would describe the two predictors as 'confounded' because it wouldn't be possible to disentangle their respective effects.

```
boxplot(myna$FORAGE)
boxplot(myna$VIGIL)
```



These boxplots don't look too bad. If these were response variables we would transform them. A predictors they are probably okay. Pick one to use in the model (it doesn't really matter which one--they are effectively the same thing).

Now we can create a model. I'm going to use foraging, but either predictor is fine.

```
myna.glm <- glm(PECK.rate ~ FORAGE * HUMANS * CONSPECS * VEHICLES *
REGION, data = myna, family = poisson(link = "log"))
```

Check a summary of the full model:

```
summary(myna.glm)
```

Are any of the interactions significant? Do any of them have a P < 0.05?

The output is quite lengthy, and is pasted on the next page. Note that the way the output is generated lays out all possible contrasts of factorial predictors. NA appears when a particular pairing doesn't exist in the data. So that for example a situation where **Human = Yes**, **Conspecifics = Yes**, **Vehicles = Yes** was never recorded in an **Urban** environment (the very last line of output with NA next to it).

Also, it is worth being aware at this point that the effect sizes are uninterpretable unless you can reverse log numbers in your head. They are presented as the transformed values (logged, because of the link function we used).

RESULT

```
        Call:
glm(formula = PECK.rate ~ FORAGE * HUMANS * CONSPECS * VEHICLES *
    REGION, family = poisson(link = "log"), data = myna)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6655  -0.1941   0.0000   0.1729   0.9167

Coefficients: (8 not defined because of singularities)
                                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                                          1.70064    1.23823   1.373    0.170
FORAGE                                              -0.65030    1.73887  -0.374    0.708
HUMANSYES                                           -0.84695   31.92537  -0.027    0.979
CONSPECSYES                                          0.10434    3.40645   0.031    0.976
VEHICLESYES                                         -0.62352    1.85290  -0.337    0.736
REGIONRural                                          0.50278    1.66703   0.302    0.763
REGIONSuburban                                       0.22102    2.90307   0.076    0.939
REGIONUrban                                          0.58389    5.63319   0.104    0.917
FORAGE:HUMANSYES                                     0.99741   45.23772   0.022    0.982
FORAGE:CONSPECSYES                                  -0.10079    5.13540  -0.020    0.984
HUMANSYES:CONSPECSYES                                0.14058   36.47727   0.004    0.997
FORAGE:VEHICLESYES                                   0.73939    2.61155   0.283    0.777
HUMANSYES:VEHICLESYES                               -0.95617   31.42556  -0.030    0.976
CONSPECSYES:VEHICLESYES                             -1.01260    4.07884  -0.248    0.804
FORAGE:REGIONRural                                  -0.87373    2.43245  -0.359    0.719
FORAGE:REGIONSuburban                               -0.33694    4.24338  -0.079    0.937
FORAGE:REGIONUrban                                  -0.88110    7.70457  -0.114    0.909
HUMANSYES:REGIONRural                                1.08223   10.22576   0.106    0.916
HUMANSYES:REGIONSuburban                            -0.73287   11.87381  -0.062    0.951
HUMANSYES:REGIONUrban                               -0.05128    7.59494  -0.007    0.995
CONSPECSYES:REGIONRural                             -1.54173    4.26170  -0.362    0.718
CONSPECSYES:REGIONSuburban                          -0.64087    4.33917  -0.148    0.883
CONSPECSYES:REGIONUrban                             -1.86311    6.54198  -0.285    0.776
VEHICLESYES:REGIONRural                              0.62216    5.37331   0.116    0.908
VEHICLESYES:REGIONSuburban                           0.44497    6.59480   0.067    0.946
VEHICLESYES:REGIONUrban                              1.83943    7.25732   0.253    0.800
FORAGE:HUMANSYES:CONSPECSYES                        -0.24633   51.44416  -0.005    0.996
FORAGE:HUMANSYES:VEHICLESYES                         1.83710   44.52425   0.041    0.967
FORAGE:CONSPECSYES:VEHICLESYES                       1.44039    6.00074   0.240    0.810
HUMANSYES:CONSPECSYES:VEHICLESYES                    0.64441   30.28584   0.021    0.983
FORAGE:HUMANSYES:REGIONRural                        -1.35406   14.78706  -0.092    0.927
FORAGE:HUMANSYES:REGIONSuburban                      1.43465   16.26004   0.088    0.930
FORAGE:HUMANSYES:REGIONUrban                         0.53399   10.40299   0.051    0.959
FORAGE:CONSPECSYES:REGIONRural                       2.48408    6.36251   0.390    0.696
FORAGE:CONSPECSYES:REGIONSuburban                    0.74691    6.49397   0.115    0.908
FORAGE:CONSPECSYES:REGIONUrban                       2.65640    9.17892   0.289    0.772
HUMANSYES:CONSPECSYES:REGIONRural                    1.43287   20.67838   0.069    0.945
HUMANSYES:CONSPECSYES:REGIONSuburban                -1.36013   21.02278  -0.065    0.948
HUMANSYES:CONSPECSYES:REGIONUrban                    1.36053   19.25319   0.071    0.944
FORAGE:VEHICLESYES:REGIONRural                      -0.69581    7.67118  -0.091    0.928
FORAGE:VEHICLESYES:REGIONSuburban                   -0.13214    9.17660  -0.014    0.989
FORAGE:VEHICLESYES:REGIONUrban                      -2.29267    9.95876  -0.230    0.818
HUMANSYES:VEHICLESYES:REGIONRural                   -0.38255    2.51481  -0.152    0.879
HUMANSYES:VEHICLESYES:REGIONSuburban                 2.82335    8.34439   0.338    0.735
HUMANSYES:VEHICLESYES:REGIONUrban                   -0.56255    2.04104  -0.276    0.783
CONSPECSYES:VEHICLESYES:REGIONRural                  1.10659    6.98549   0.158    0.874
CONSPECSYES:VEHICLESYES:REGIONSuburban               1.94293    7.61644   0.255    0.799
CONSPECSYES:VEHICLESYES:REGIONUrban                  0.36954    9.62113   0.038    0.969
FORAGE:HUMANSYES:CONSPECSYES:VEHICLESYES            -0.51067   45.21351  -0.011    0.991
FORAGE:HUMANSYES:CONSPECSYES:REGIONRural            -2.37205   29.20126  -0.081    0.935
FORAGE:HUMANSYES:CONSPECSYES:REGIONSuburban          2.25463   29.02321   0.078    0.938
FORAGE:HUMANSYES:CONSPECSYES:REGIONUrban            -2.10929   26.66577  -0.079    0.937
FORAGE:HUMANSYES:VEHICLESYES:REGIONRural                  NA         NA      NA       NA
FORAGE:HUMANSYES:VEHICLESYES:REGIONSuburban         -4.99244   11.23960  -0.444    0.657
FORAGE:HUMANSYES:VEHICLESYES:REGIONUrban                  NA         NA      NA       NA
FORAGE:CONSPECSYES:VEHICLESYES:REGIONRural          -1.68433   10.10136  -0.167    0.868
FORAGE:CONSPECSYES:VEHICLESYES:REGIONSuburban       -2.91475   10.77265  -0.271    0.787
FORAGE:CONSPECSYES:VEHICLESYES:REGIONUrban          -0.91132   13.24266  -0.069    0.945
HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONRural             NA         NA      NA       NA
HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONSuburban          NA         NA      NA       NA
HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONUrban             NA         NA      NA       NA
FORAGE:HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONRural      NA         NA      NA       NA
FORAGE:HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONSuburban   NA         NA      NA       NA
FORAGE:HUMANSYES:CONSPECSYES:VEHICLESYES:REGIONUrban      NA         NA      NA       NA

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 21.259  on 159  degrees of freedom
Residual deviance: 11.747  on 104  degrees of freedom
AIC: 624.26

Number of Fisher Scoring iterations: 4
```

If none of the interactions are significant we should remove them from the model and re-apply the test. If we were going to publish this we might take some more care, removing the higher order (four and three way) interactions first and checking for lower order interactions. For the sake of speed though, let's just proceed with removing the interaction terms:

```
myna.glm <- glm(PECK.rate ~ FORAGE + HUMANS + CONSPECS + VEHICLES +
REGION, data = myna, family = poisson(link = "log"))
```

Check a summary of the full model:

```
summary(myna.glm)
```

The results in the output you should look at are:

RESULT

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.21513    0.29569   4.109 3.97e-05 ***
FORAGE       -0.02289    0.40026  -0.057    0.954
HUMANSYES     0.02128    0.10048   0.212    0.832
CONSPECSYES   0.04482    0.08777   0.511    0.610
VEHICLESYES  -0.01227    0.08543  -0.144    0.886
REGIONRural   0.04313    0.12123   0.356    0.722
REGIONSuburban  0.06952    0.12261   0.567    0.571
REGIONUrban   0.02047    0.12826   0.160    0.873
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 21.259  on 159  degrees of freedom
Residual deviance: 20.375  on 152  degrees of freedom
AIC: 536.89

Number of Fisher Scoring iterations: 4
```

Has overdispersal occurred? How can you tell?

Overdispersion occurs if the residual deviance is greater than the degrees of freedom. In this case, it is not (20.378 is less than 152), so the model is okay in that regard.

The "dispersion parameter" of 1 assumes the variance = mean (don't worry about this for now. We'll return to this a bit later).

The numbers in **red** are the estimated intercept and slopes on the **log scale**. These are effect sizes and the `Std. Error` is the error of the effect size. If you were to present this as a table in a set of results you would present it like so:

Table 1. An example of how to present these results in a tabular form for a report.

	Effect Size	SE	z	P	P < 0.05
Intercept	1.22	0.30	4.11	<0.001	*
FORAGE	-0.02	0.40	-0.06	0.954	
HUMANS (yes)	0.02	0.10	0.21	0.832	
CONSPECIFICS (yes)	0.04	0.09	0.51	0.610	
VEHICLES (yes)	-0.01	0.09	-0.14	0.886	
REGION (Rural)	0.04	0.12	0.36	0.722	
REGION (Suburban)	0.07	0.12	0.57	0.571	
REGION (Urban)	0.02	0.13	0.16	0.873	

The intercept wouldn't always be included in a results table because it isn't especially interesting (we expect it to depart from zero), so although I've included it, the intercept is often simply omitted.

Breaking down the output

DEVIANCE

Deviance is a measure of 'goodness of fit', or maybe more accurately, 'badness of fit' because higher numbers indicate a greater deviance away from a good fit. That is, if the Deviance is high the observed values are not matching the expected values given the predictors we have included in the model. If the Deviance is low, then the observed values are matching the expected values given the predictors. This is the reverse of an R^2 value, where a high R^2 indicates a good fit and a low R^2 indicates a poor fit.

The output calculates and provides Deviance in two ways: the null Deviance and the residual deviance. The null deviance tells you how well the data fit a model **that only includes the intercept** (i.e. in the absence of any of the predictors, how well does the data fit the model). The residual Deviance tells you **how much of the data is explained when the independent predictors are included in the model**. If the predictors are important for explaining the response, we would expect the **residual Deviance to be substantially lower than the null Deviance**.

In the above example, the null Deviance is 21.25 with 159 degrees of freedom, whereas the residual deviance is 20.38 with 152 degrees of freedom. We have lost some degrees of freedom because including the predictors eats away at degrees of freedom as per statistical analyses in general.

This is not an especially substantial improvement in Deviance, and even without looking at the P values, we could take a guess that the data is probably not being explained by these predictors in a very convincing way.

Interpreting the Coefficients

In the example above we produced a final table as per below (repeated from above):

Table 1. An example of how to present results in a tabular form in a report.

	Effect Size	SE	z	P	P < 0.05
Intercept	1.22	0.30	4.11	<0.001	*
FORAGE	-0.02	0.40	-0.06	0.954	
HUMANS (yes)	0.02	0.10	0.21	0.832	
CONSPECIFICS (yes)	0.04	0.09	0.51	0.610	
VEHICLES (yes)	-0.01	0.09	-0.14	0.886	
REGION (Rural)	0.04	0.12	0.36	0.722	
REGION (Suburban)	0.07	0.12	0.57	0.571	
REGION (Urban)	0.02	0.13	0.16	0.873	

However, one thing we are actually interested in is the real effect of a predictor, perhaps **`Foraging Time`**, on the response, in this case **`Peck.rate`**. However, coefficients (effect sizes) in glms are generated in complex ways, and interpreting them in terms of real world units is far from straightforward. If we want to know the effect of increasing **`Foraging Time`** by one unit, then we have to take into account the effects of the other predictors. The first point of importance is that interpretation of coefficients is different for numeric and categorical (factor) predictor variables.

Numeric predictors: The coefficient represents an exponent of a term that can be used multiplicatively to work out the effect *of increasing the predictor by 1 unit*.

Factorial predictors: The coefficient represents an exponent of a term that can be used multiplicatively to work out the effect *relative to the base (first) level for the factor*. The base level is the one that is missing for each factor in the output. For example, for Region, Periurban is missing, and Periurban is the base level.

Because the effects of the coefficients are dependent on other predictors, and because the coefficient is an exponential term to a base (e.g. exponential to base Euler's number (2.71828) for Poisson distributions), there are three important points to keep in mind.

1) The effect of a predictor depends on the level of the response
2) Additive changes in predictors has multiplicative effects on the response
3) It isn't possible to just interpret the coefficients unless you can mentally compute arbitrary exponentials in your head whilst humming Mozart and painting a Neo-Impressionist masterpiece.

The best, possibly only, way to really understand how to turn the effect sizes into comprehensible units is by use of examples. The first thing we need to do is look at our predictor variables:

- **Region** Factorial with four levels (periurban, rural, suburban, urban)
- **Forage** Numeric
- **Vigil** Numeric
- **Humans** Factorial with two levels (yes, no)
- **Conspecs** Factorial with two levels (yes, no)
- **Vehicles** Factorial with two levels (yes, no)

We used Region, Forage, Humans, Conspecifics and Vehicles in the model, and all of the predictors need to be taken into account when working out effect sizes in actual units.

- The Forage coefficient represents the effect for every 1 unit increase in Forage.
- The Humans coefficient represents the effect of Yes, relative to No (No = zero).
- The Conspecs coefficient represents the effect of Yes, relative to No (No = zero).
- The Vehicles coefficient represents the effect of Yes, relative to No (No = zero).
- The Rural coefficient represents the effect of Rural, relative to Periurban (= zero).
- The Suburbun coefficient represents the effect of Suburban, relative to Periurban.
- The Urban coefficient represents the effect of Urban, relative to Periurban.
- The Intercept coefficient is the baseline, and all other coeffecients are relative to it

To work out the effects of the various predictors we need to take into account the state of the whole system, so to speak. The following examples should help clarify this:

The estimated Peck Rate for a Forage of 1.0 (100% of time spent foraging), with no Humans, no conspecifics, no vehicles, in a Periurban environment:

```
exp(1.21513 + 1*-0.02289 + 0 + 0 + 0 + 0)
[1] 3.294453
```

The estimated Peck Rate for a Forage of 0.5 (50% of time spent foraging), with no Humans, no conspecifics, no vehicles, in a Periurban environment:

```
exp(1.21513 + 0.5*-0.02289 + 0 + 0 + 0 + 0)
[1] 3.332374
```

The estimated Peck Rate for a Forage of 0.5 (50% of time spent foraging), with Humans, no conspecifics, no vehicles, in a Periurban environment:

```
exp(1.21513 + 0.5*-0.02289 + 0.02128 + 0 + 0 + 0)
[1] 3.404047
```

The estimated Peck Rate for a Forage of 0.5 (50% of time spent foraging), with Humans, no conspecifics, no vehicles, in a Suburban environment:

```
exp(1.21513 + 0.5*-0.02289 + 0.02128 + 0 + 0 + 0.06952)
[1] 3.649116
```

Pretty soon, you can see how by playing around with the numbers you can work out the situation under which Peck Rate is estimated to be highest, and the situation under which Peck Rate is estimated to be lowest.

Note also, that we are using 0.5, 1.0 etc for Forage, because we have a variable that was measured as a percentage. If instead you had measured the number of minutes spent foraging *instead*, then `1*-0.02289` would represent 1 min spent foraging, `2*-0.02289` would represent two minutes spent foraging etc.

Using the Predict Function

Note that you can also use the **predict** function to produce the same estimates:

The estimated Peck Rate for a Forage of 0.5 (50% of time spent foraging), with no Humans, no conspecifics, no vehicles, in a Periurban environment (as above):

```
exp(1.21513 + 0.5*-0.02289 + 0 + 0 + 0 + 0)
[1] 3.332374
```

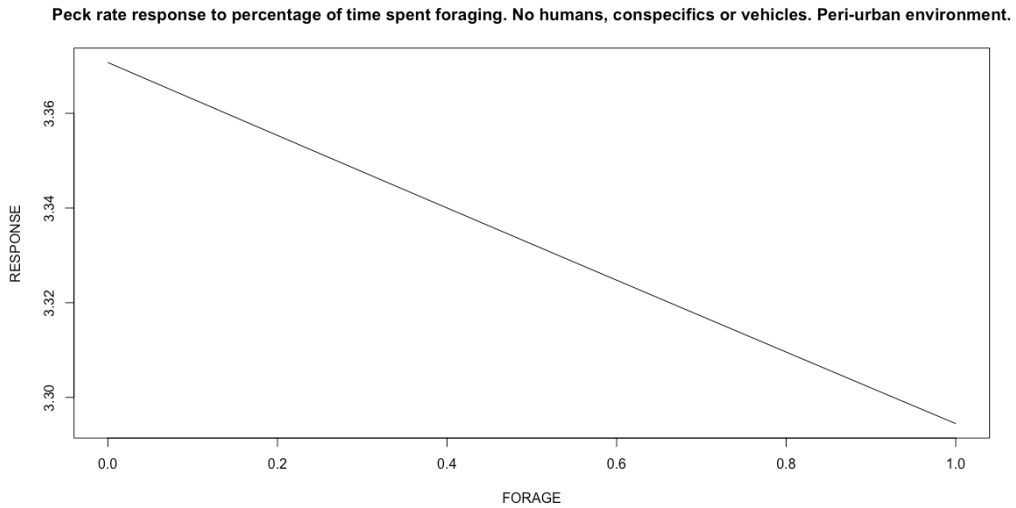
Using the `predict` function:

```
# create a dataframe holding the model states you want
# be careful to get spelling exactly right!

newdata <- data.frame(FORAGE = 0.5, HUMANS = "NO", CONSPECS =
"NO", VEHICLES = "NO", REGION = "Peri-urban")

# use the model you created earlier and predict function

predict(myna.glm, newdata, type="response")
```

Using the predict function: graphing output for a given range

You can also ask the **predict** function to return a range of responses for a given predictor. This is typically easiest to plot and interpret if you look at a range for one predictor while keeping the others set, which we will do here for forage.

```
myna.glm <- glm(PECK.rate ~ FORAGE + HUMANS + CONSPECS + VEHICLES + REGION, data = myna, family = poisson(link = "log"))

summary(myna$FORAGE) # to see ranges

forage.data<-seq(0, 1, by=0.01) # create a range of numbers from 0 to 1 at 0.01 increments. This could be any range of numbers but because Forage is a proportion from zero to one, this is a sensible range to look at.

forage.data # Look at the forage data

newdata <- data.frame(FORAGE = forage.data, HUMANS = "NO", CONSPECS = "NO", VEHICLES = "NO", REGION = "Peri-urban")

newdata # look at the dataframe

predict(myna.glm, newdata, type="response") # get responses across the specified range. This will just spit out numbers, which lets us check it is working.

RESPONSE <- predict(myna.glm, newdata, type="response") # drop into an object called RESPONSE

plot(RESPONSE~FORAGE, newdata, type="l") # Plot status across the ranges of FORAGE
```



Counterintuitively, peck rate is highest when time spent foraging is lowest. Perhaps birds compensate for less time foraging by pecking faster?

Relative Strength of Effect

But what if you simply want to be able to discuss the effects of the predictors in terms of their relative strengths? The following approach has just been made up by me right at this moment, so you probably want to run it past a real statistician before using it in public. That said, in principle it seems like this ought to work fine.

1) Turn the z values or t values into non-signed values (all positive)
2) Sum up and average any z or t values for factors with multiple levels
3) Work out the percentage of each z or t value relative to the intercept z or t value
4) Arbitrarily group predictors as follows
   - >50% Very Strong Relative Effect
   - 30-50% Strong Relative Effect
   - 15-30% Moderate Relative Effect
   - 10-15% Weak Relative Effect
   - 5-10% Very Weak Relative Effect
   - <5% Negligible Relative Effect

RELATIVE EFFECT EXAMPLES

Relative effects of predictors on the Peck Rate of mynas

```
Call:
glm(formula = PECK.rate ~ FORAGE + HUMANS + CONSPECS + VEHICLES +
    REGION, family = poisson(link = "log"), data = myna)

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     1.21513    0.29569   4.109 3.97e-05 ***
FORAGE         -0.02289    0.40026  -0.057    0.954
HUMANSYES       0.02128    0.10048   0.212    0.832
CONSPECSYES     0.04482    0.08777   0.511    0.610
VEHICLESYES    -0.01227    0.08543  -0.144    0.886
REGIONRural     0.04313    0.12123   0.356    0.722
REGIONSuburban  0.06952    0.12261   0.567    0.571
REGIONUrban     0.02047    0.12826   0.160    0.873
```

Forage = **0.057/4.109** = **0.0139** = 1.4% = **Negligible effect**
Humans = **0.212/4.109** = **0.0516** = 5.1% = **Very weak effect**
Conspecifics = **0.511/4.109** = **0.1243** = 12.4% = **Weak effect**
Vehicles = **0.144/4.109** = **0.0350** = 3.5% = **Negligible effect**
Region = **(0.356 + 0.567 + 0.160) / 3 = 0.361**
= **0.361/4.109** = **0.0879** = 0.9% = **Negligible effect**

Relative effects of predictors on the number of darters observed in two rivers

```
Call:
glm(formula = darters ~ river + pH + temp, family = poisson, data =
darterData)

Coefficients:
             Estimate Std.Error z value Pr(>|z|)
(Intercept)   3.144257  0.218646  14.381  < 2e-16 ***
riverWatauga -0.049016  0.051548  -0.951  0.34166
pH            0.086460  0.029821   2.899  0.00374 **
temp         -0.059667  0.009149  -6.522  6.95e-11 ***
```

River System = **0.951/14.381** = **0.0661** = 6.6% = **Very weak effect**
pH = **2.899/14.381**= **0.2015** = 20.2% = **Moderate effect**
Temperature = **6.522/14.381** = **0.4535** = 45.4% = **Strong effect**

Evaluating assumptions for a GLM

We really only have one assumption to check (is the link function correct?), but we do need to check this using a few different checks.

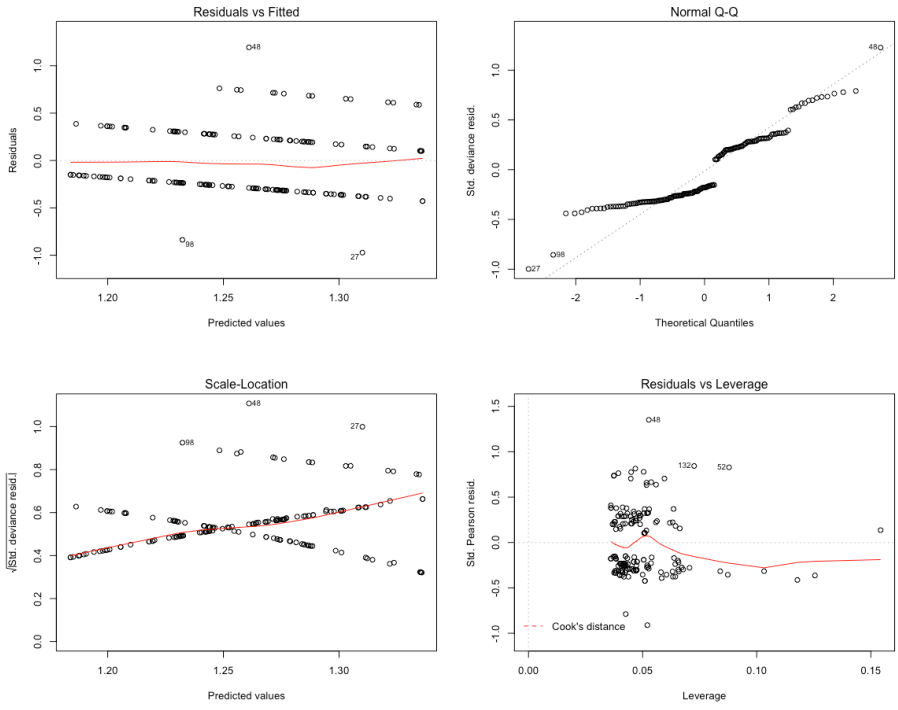
1) Check for overdispersion
2) Look at diagnostic plots
3) Check that mean:variance is approximately 1:1

```
myna.glm <- glm(PECK.rate ~ FORAGE + HUMANS + CONSPECS + VEHICLES + REGION, data = myna, family = poisson(link = "log"))
```

Our first step is to compare the Residual Deviance with the degrees of freedom. This is taken from the **summary(myna.glm)** output. We have **Residual deviance: 20.378 on 152 degrees of freedom**, which is fine. We want to see a *lower* residual deviance than degrees of freedom. If residual deviance is greater than the degrees of freedom, then 'overdispersion' has occurred, and the assumptions of the model are not met.

Next check diagnostic plots of the model. These are interpeted exactly the same way as diagnostic plots for other types of linear models. That is, you *don't* want to see a wedge in the residuals vs fitted (test of equal variance of residuals), but you do want to see the QQ dots following the line (test of normality of residuals).

```
par(mfrow=c(2,2))
plot(myna.glm)
```



These look mostly fine. The QQ plot is probably the worst of them, and it would be borderline for being acceptable.

We also have to check that the assumption **(Dispersion parameter for poisson family taken to be 1)** is satisfied. To do this we need to check the mean and variance for the response variable split up by the main predictor of interest. We want to see that they are approximate equal, i.e. 1:1.

```
attach(myna)
tapply(PECK.rate,REGION,mean)
tapply(PECK.rate,REGION,var)
```

```
tapply(PECK.rate,REGION,mean)
Peri-urban      Rural      Suburban     Urban
3.375           3.550      3.675        3.500

tapply(PECK.rate,REGION,var)
Peri-urban      Rural      Suburban     Urban
0.2916667       0.5615385  0.6352564    0.4615385
```

The variances are well below the means. We could accept a difference of maybe ±25% but this difference is too extreme. So that actually, we need to re-run the model using a quaispoisson distribution. But why didn't we just start with a quasipoisson distribution and save the hassle? Remember: **simpler models are always preferred**.

We start with the simplest approach and work towards complexity if it is needed. Reapply the test using a quasipoisson distribution. Notice how the assumed dispersion parameter of 1 has now been adjusted to a value of **0.1380571**. Remodel using the quasipoisson distribution:

```
myna.glm <- glm(PECK.rate ~ VIGIL + HUMANS + CONSPECS + VEHICLES +
REGION, data = myna, family = quasipoisson(link = "log"))
```

Check a summary of the full model:

```
summary(myna.glm)
```

If either dispersion > df **or** var/mean is not equal to 1 (roughly) then you need to use a quasipoisson distribution or a quasibinomial. You will find that the dispersion parameter changes (it is no longer taken to be 1). Overdispersion will probably still occur, but this is no longer a concern if you are using one of the quasi- distributions.

To check for overdisperstion **in a glmer model** (i.e. one with mixed effects) use dispersion check function in library blemco.

```
install.packages("blemco")
library(blemco)
dispersion_glmer(yourmodel.glmer)

# if over 1.4 or under 0.75 you may have dispersion problems
```

RESULT

```
> myna.glm <- glm(PECK.rate ~ VIGIL + HUMANS + CONSPECS + VEHICLES
+ REGION, data = myna, family = quasipoisson(link = "log"))
> summary(myna.glm)

Call:
glm(formula = PECK.rate ~ VIGIL + HUMANS + CONSPECS + VEHICLES +
    REGION, family = quasipoisson(link = "log"), data = myna)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9742  -0.2982  -0.1761   0.2742   1.1920

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.19413    0.05408  22.080   <2e-16 ***
VIGIL           0.02058    0.15043   0.137    0.891
HUMANSYES       0.02152    0.03723   0.578    0.564
CONSPECSYES     0.04459    0.03266   1.366    0.174
VEHICLESYES    -0.01247    0.03165  -0.394    0.694
REGIONRural     0.04314    0.04507   0.957    0.340
REGIONSuburban  0.06934    0.04558   1.521    0.130
REGIONUrban     0.02021    0.04756   0.425    0.671
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 0.1380548)

    Null deviance: 21.259  on 159  degrees of freedom
Residual deviance: 20.375  on 152  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

Notice how the dispersion parameter is no longer taken to be 1? You can check the diagnostic plots again, although you'll find that the qq plot isn't improved tremendously.

Adjusting GLMs for different sampling efforts

An offset variable can be used to adjust for differences in sampling effort in a glm. In the following example there is a sampling effort variable (this would be a column in your data sheet with trap nights or samples per site) included as YOUR.SAMPLING.EFFORT.

```
model.glm <- glm(YOUR.COUNT ~ YOUR.PREDICTORS +
offset(YOUR.SAMPLING.EFFORT), data = yourdata, family=poisson (link
= "log"))
```

Here's how a reduced model (with no interactions) might look:

```
model.glm <- glm(COUNT ~ DISTANCE.FROM.PATH + NEAR.WATER +
LOGS.COUNT + MICROHABITAT + CANOPY.COVER + offset(SAMPLING.EFFORT),
data = spider.counts, family=poisson(link = "log"))
```

In principal, the method for working through the glm is very similar to an ANOVA. You need to think about the order of predictors, look for significant interactions and remove them, and avoid over-fitting your model with too many predictors.

We've use an 'offset' variable to take into account differences in sampling effort, so that we don't have to do anything to the raw count (i.e. we don't need to generate a ratio of counts per unit of sample effort). You do need to check the assumptions of the model, and remember that glm assumptions are different to ANOVA assumptions.

Zero Inflated glms

Another problem that can occur is if there **are a lot of zeroes in the data**. In this situation we would want to check the glm against what is called a 'zero inflated' glm. A zero inflated glm is specifically designed to cope with datasets that have a lot of zeroes. Unfortunately, there is no core zero inflated glm model, so we need to make use of a library:

```
install.packages("pscl")
library(pscl)
```

The code is slightly different. The 'offset' variable is placed at the end, like so:

```
zeromodel.glm <- zeroinfl(COUNT ~ DISTANCE.FROM.PATH + NEAR.WATER +
LOGS.COUNT + MICROHABITAT + CANOPY.COVER, data = spider.counts,
offset = SAMPLING.EFFORT) # zero inflated glm
```

A Vuong Test can be used to compare the ordinary glm with the zero inflated glm. If there is a significant difference, then the zero inflated glm is better.

```
model.glm <- glm(COUNT ~ DISTANCE.FROM.PATH + NEAR.WATER +
LOGS.COUNT + MICROHABITAT + CANOPY.COVER + offset(SAMPLING.EFFORT),
data = spider.counts, family=poisson(link = "log")) # standard glm

vuong(zeromodel.glm, model.glm)

Vuong Non-Nested Hypothesis Test-Statistic: 4.824468
(test-statistic is asymptotically distributed N(0,1) under the
 null that the models are indistinguishible)
in this case:
model1 > model2, with p-value 7.018877e-07
```

In this case the *P* value is significant ($P < 0.05$), which indicates that model 1 (the zero inflated glm) is better than model 2 (the standard glm).

GLM: Binomial distribution

You should now be able to run a binomial GLM (using a binary response variable). The steps are the same as for a GLM using a poisson distribution, except that the family distribution is set to `binomial(link = "logit")` and if you need to resort to it, the quasi distribution family is `quasibinomial(link = "logit")`.

Using a binary response variable:

Import the data **yellowrobin.csv** data set.

```
robin <- read.table('yellowrobin.csv',header=T,sep=',')
```

Check the data

```
str(robin)
```

This data shows presence (= 1) and absence (= 0) for Eastern Yellow Robins in southeastern Victoria. The predicator variables are ordinal habitat scores (0 = none, 5 = heavy) and the size of the forest path in hectares (ha).

```
robin <- read.table('yellowrobin.csv',header=T,sep=',')
str(robin)

attach(robin) # A quick way to look at multiple correlations
round(cor(robin[,4:7]),2) # correlation grid: columns 4-7

robin.glm <- glm(Pr.Ab ~
AREA.ha*LEAF.LITTER*SHRUBS*CANOPY*WOODY.DEBRIS, family =
binomial(link="logit"), data = robin)
summary(robin.glm)
```

You will receive an error message at this point. The model is overparamtised and cannot be build (too many predictors, not enough observations). If we were going to publish this we would procede to check interactions in a careful, interaction-by-interaction approach, but for the sake of moving along we will just remove the interaction terms for now. This will drastically reduce the number of predictor terms in the model.

```
robin.glm <- glm(Pr.Ab ~
AREA.ha+LEAF.LITTER+SHRUBS+CANOPY+WOODY.DEBRIS, family =
binomial(link="logit"), data = robin)
```

Now check assumptions.

1) Has over-dispersion occurred?

```
summary(robin.glm) # just looking at the dispersion part of the output
```

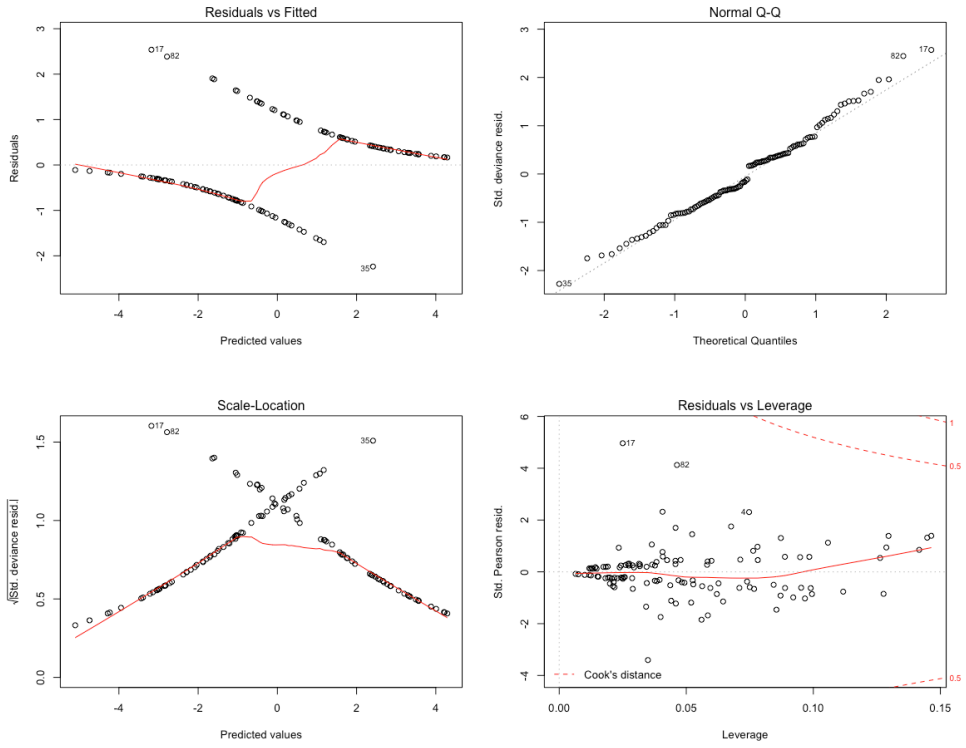
RESULT

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 166.222  on 119  degrees of freedom
Residual deviance:  95.806  on 114  degrees of freedom
```

95.806 is less than 114, so overdispersion has not occurred.

2) How do the diagnostic plots look?

```
par(mfrow=c(2,2))
plot(robin.glm)
```



These plots look okay, considering we are working with binary data. There is no obvious 'wedge' in the Residuals vs Fitted and the QQ looks fine (i.e. the dots are more or less on the line).

3) Is mean:var approximately 1:1?

```
attach(robin) # just looking at the man and variance by one
category. We could check all the predictors if unsure about this
one.
tapply(Pr.Ab,SHRUBS,mean)
tapply(Pr.Ab,SHRUBS,var)
```

RESULT

```
> tapply(Pr.Ab,SHRUBS,mean)
        0         1         2         3         4         5
0.7428571 0.5000000 0.3000000 0.4074074 0.2142857 0.3333333

> tapply(Pr.Ab,SHRUBS,var)
        0         1         2         3         4         5
0.1966387 0.2619048 0.2333333 0.2507123 0.1813187 0.2424242
```

These look more or less proportional. They are not exactly 1:1, but they are probably close enough to be okay.

```
robin.glm <- glm(Pr.Ab ~
AREA.ha+LEAF.LITTER+SHRUBS+CANOPY+WOODY.DEBRIS, family =
binomial(link="logit"), data = robin)

summary(robin.glm)
```

Interpretation

Which explanatory variables were significantly associated with Yellow Eastern Robin presence or absence?

RESULT

```
> summary(robin.glm)

Call:
glm(formula = Pr.Ab ~ AREA.ha + LEAF.LITTER + SHRUBS + CANOPY
+
    WOODY.DEBRIS, family = binomial(link = "logit"), data =
robin)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2363  -0.6343  -0.1487   0.5384   2.5379

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.920642   1.403663  -3.506 0.000456 ***
AREA.ha       0.014290   0.002854   5.007 5.54e-07 ***
LEAF.LITTER   0.099062   0.177167   0.559 0.576064
SHRUBS        0.528087   0.224560   2.352 0.018690 *
CANOPY       -0.275388   0.208429  -1.321 0.186416
WOODY.DEBRIS -0.096332   0.154887  -0.622 0.533973
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 166.222  on 119  degrees of freedom
Residual deviance:  95.806  on 114  degrees of freedom
AIC: 107.81

Number of Fisher Scoring iterations: 5
```

Area of forest fragment and Shrubs appear to have a significant association with probability of robin presence.

For the Eastern Yellowbreasted robins, what is the probability of robins being present in a forest block of 100 ha with Leaf Litter = 1, Shrubs = 2, Canopy = 1 and Woody Debris = 3?

You can use the same method we used above for a Poisson model, except note that because we ned to reverse a logistic regression, we need an inverse logit function, not an exponent.

```
??inv.logit
```

It trurns out that **inv.logit** is in the **boot** library.

```
library(boot)
inv.logit(Intercept + multiplier*X + multiplier*Y + multiplier*Z)
```

Now we'll have a go at working out the probability of robins being present in a forest block of 200 ha with Leaf Litter = 0, Shrubs = 5, Canopy = 4 and Woody Debris = 4. The values come from the estimates on the previous page.

```
inv.logit(-4.920642 + 200 * 0.014290 + 0 * 0.099062 + 5 * 0.528087 +
4 * -0.275388 + 4 * -0.096332)
```

RESULT

```
> inv.logit(-4.920642 + 200 * 0.014290 + 0 * 0.099062 + 5 *
0.528087 + 4 * -0.275388 + 4 * -0.096332)

[1] 0.2871867
```

Given the above conditions, our model is predicting a 28.7% chance of robin presence in a forest block.

Graphing

There are several options for graphing binary data against a continuous predictor. Three that you might find useful are the conditional density plot, the spline plot and a proportion plot. First both your response needs to be afactor:

```
dataset$your.response <- as.factor(dataset$your.response)
```

Conditional Density Plot

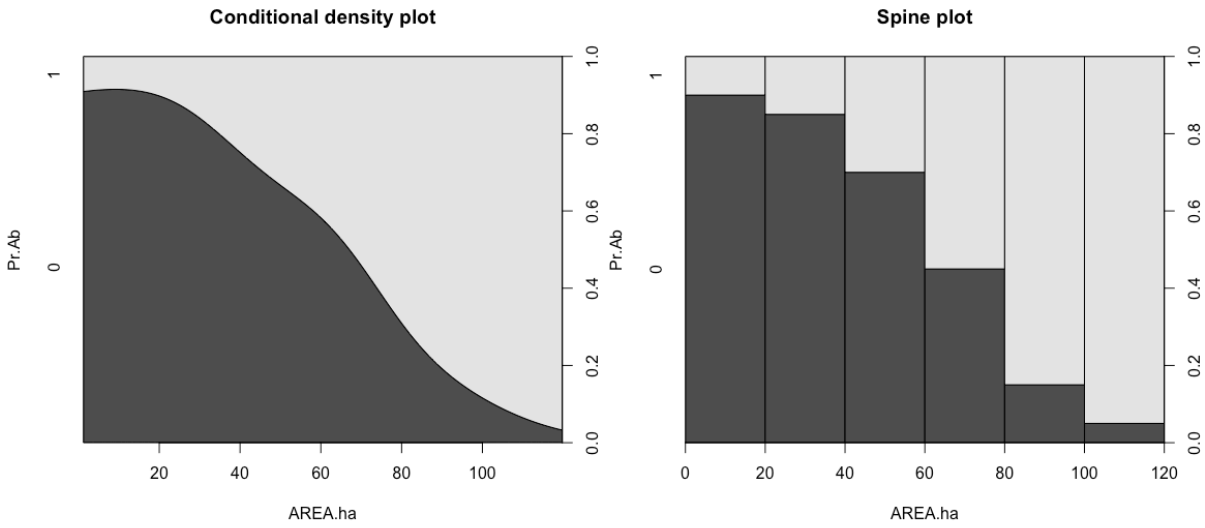
```
cdplot(your.response ~ your.predictor, data = your.data)
```

Spline Plot

```
spineplot(your.response ~ your.predictor, data = your.data)
```

Usually we would only plot the **significant** relationships. We will plot the relationship with area of forest fragment in ha.

```
robin$Pr.Ab <- as.factor(robin$Pr.Ab)

# Conditional density plot
cdplot(Pr.Ab ~ AREA.ha, data = robin)

# Spine plot
spineplot(Pr.Ab ~ AREA.ha, data = robin)
```



You wouldn't typically report both plots, rather you'd tend to pick the one that you think shows the relationship most clearly. In both cases the dark section represents proportion of forest fragments where robins where absent. This decreases as the area of the forest gets larger.

Mixed Effect Models

Mixed effect models have become very popular in biology. We're only going to look at them briefly because they can become quite complicated and we don't want to get bogged down in too many details at this point.

A mixed effect model contains two types of explanatory variables. These are **fixed effects** and **random effects**. The difference between these variables is that:

- **Fixed effects:** Influence only the **mean** of the response variable **(y)**
- **Random effects:** Influence only the variance of the response variable **(y)**

Biologists can get into elaborate debates about whether a variable should be a fixed or random effect when entered into a model. As a very rough rule of thumb *usually* (but not always) the **fixed effects are the explanatory variables of interest**, whereas the **random effects are the factors that you want to control for to avoid pseudoreplication** (random effects are always factors, whereas fixed effects can be continuous variables, counts, binary data or factors). If you dig into this deeper, you will actually find many conflicting definitions. Here are a few collated by Andrew Gelman (*Analysis of variance—why it is more important than ever* (2005) Annals of Statistics):

(1) Fixed effects are constant across individuals, and random effects vary. For example, in a growth study, a model with random intercepts a_i and fixed slope b corresponds to parallel lines for different individuals i, or the model y_it = a_i + b t. Kreft and De Leeuw (1998) thus distinguish between fixed and random coefficients.

(2) Effects are fixed if they are interesting in themselves or random if there is interest in the underlying population. Searle, Casella, and McCulloch (1992, Section 1.4) explore this distinction in depth.

(3) "When a sample exhausts the population, the corresponding variable is fixed; when the sample is a small (i.e., negligible) part of the population the corresponding variable is random." (Green and Tukey, 1960)

(4) "If an effect is assumed to be a realized value of a random variable, it is called a random effect." (LaMotte, 1983)

(5) Fixed effects are estimated using least squares (or, more generally, maximum likelihood) and random effects are estimated with shrinkage ("linear unbiased prediction" in the terminology of Robinson, 1991). This definition is standard in the multilevel modeling literature (see, for example, Snijders and Bosker, 1999, Section 4.2) and in econometrics.

The problem is that these various definitions conflict, sometimes wildly. However, from a pragmatic point of view, the important thing is that a random effect is (1) a factor (i.e. not a covariate) that is taken into account in a model (2) before the fixed effects are examined. This means that if most of the variance in a model can be explained by a fixed effect (animal ID, plot, site, year), then that variance is removed before the fixed effects get a chance to explain anything. This effectively controls for pseudoreplication. In effect, we are trying to control for 'groups' with shared properties. For our purposes, a group is any set of individuals that share some sort of common property.

- Multiple observations of an animal
- Multiple animals from a site
- Multiple sites within a landscape
- Multiple landscapes within a climactic range

Any categorical (factorial) variable can potentially be a group.

To understand how we use mixed effects in models, the best thing is probably to work through an example. To do this open the **agilis-morphometrics.csv** data set. This is a cut-down version of a PhD dataset (the actual data set is much larger). Individual agile antechinus have been captured at study sites that are either forest fragments or undisturbed continuous forest sites (this is a comparative control). The problem however, is that we have multiple individuals from each site, and there were two trapping grids per site, and it looks like we might want to control for the sampling year as well.

We could control for these things (site, trapping grid and year) by averaging all of the values by these factors, but a lot of information is going to be lost if we do that.

It would be better to include that information in a model. We're going to have a go at a simple linear mixed effect model and then we'll attempt a generalised mixed effect model.

Linear mixed effects models

The assumptions for a linear mixed effect model are the same as for a linear model or ANOVA:

ASSUMPTIONS OF LME MODEL

(1) Residuals are normally distributed

(2) Residuals must be independent (collected randomly)

(3) Residuals have equal variances

PROBLEMS MAY OCCUR IF...

(1) Predictor variables correlate

(two variables explain the same thing)

(2) The model is over-parameterised

(too many predictors given your data set size)

(3) Covariates are bounded or not normally distributed

In linear mixed effect models order is important!

The assumptions for a generalised linear mixed effect model are the same as for a generalised linear model:

ASSUMPTIONS OF GLMME MODEL

(1) Observations are independent (collected randomly)

(2) Correct link function is used

Residuals must fit the nominated error distribution

Overdispersion should not occur

PROBLEMS CAN OCCUR IF...

(1) Predictor variables are correlated

(2) Model is over-parameterised

(3) Covariates are bounded or not normally distributed

Notice how the 'problems' are the same for both lme and glmme. The problems (which are not absolute assumptions, but which can cause models to be less accurate, or fail to even work) are typical of a lot of models. Often, it is a good idea to check your predictor variables for correlation at the outset regardless of what approach you are going to take. If nothing else, you'll want to know whether it might be present in the data because sometimes correlation is itself interesting to examine.

Linear mixed effect model example

There is a strong preference in the published literature for using the package **lme4** for linear mixed effects models over **nlme**. I'll discuss key differences below. The **nlme** package is easier to learn from the start with, though, and we will use it as a starting point.

Load the necessary library:

```
library(nlme) # remember to install if not already installed
```

Import the data set, check it and remove missing values:

```
agilis <- read.table('agilis-morphometrics.csv',header=T,sep=',')
agilis <- na.omit(agilis)
str(agilis)
```

```
agilis.lme <- lme(MASS ~ MONTH + SEX * HABITAT, random = ~1|
YEAR/SITE/TRAP.GRID, method="REML", data = agilis)
```

MASS	The response variable
MONTH + SEX * HABITAT	The fixed effects. **MONTH** is a covariate. **SEX** and **HABITAT** are factors
YEAR/SITE/TRAP.GRID	The random effects **TRAP.GRID** is nested inside **SITE** which is nested inside **YEAR**
method="REML"	Instruction to use **REML** not **ML** **REML** is better for accuracy (especially of P values) **ML** is necessary for model selection
data = agilis	The data

```
plot(agilis.lme) # look at the residual plot (equal variances)
library(car)
qqPlot(resid(agilis.lme)) # look at the QQ plot (normality)
```

```
summary(agilis.lme)
anova(agilis.lme)
```

Both of these options provide P values. The **anova** option is simpler and often easier for you to interpret. Here are some extra things you can pull out of the model:

```
(cor(fitted(agilis.lme),getResponse(agilis.lme))^2) # R^2 for the
model
```

```
AIC(agilis.lme) # An AIC for the model
```

```
VarCorr(agilis.lme) # The percentage of variance explained for the
random effects
```

Presenting the LME results

You wouldn't want to present the whole output of the model in a research paper. The most important parts are shown in red.

```
> summary(agilis.lme)
Linear mixed-effects model fit by REML
 Data: agilis
       AIC      BIC    logLik
  1037.378 1066.881 -509.689

Random effects:
 Formula: ~1 | YEAR
        (Intercept)
StdDev:   0.1149765

 Formula: ~1 | SITE %in% YEAR
        (Intercept)
StdDev:    2.833205

 Formula: ~1 | TRAP.GRID %in% SITE %in% YEAR
         (Intercept) Residual
StdDev: 0.0006117597 2.409616

Fixed effects: MASS ~ MONTH + SEX * HABITAT
                    Value Std.Error DF   t-value p-value
(Intercept)      14.238141 1.7873446 81  7.966086  0.0000
MONTH             0.423502 0.2924507 56  1.448113  0.1532
SEXM              6.049804 0.4890037 81 12.371694  0.0000
HABITATFRAG       0.128548 0.9001976 56  0.142800  0.8870
SEXM:HABITATFRAG  2.388402 0.7072864 81  3.376853  0.0011
 Correlation:
                  (Intr) MONTH  SEXM   HABITA
MONTH             -0.934
SEXM              -0.161  0.006
HABITATFRAG       -0.255  0.005  0.308
SEXM:HABITATFRAG   0.110 -0.003 -0.691 -0.431

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.05428158 -0.58097493 -0.04721096  0.51659196  2.63548661

Number of Observations: 201
Number of Groups:
                         YEAR                SITE %in% YEAR TRAP.GRID %in% SITE
%in% YEAR
                            2                            60                 118
```

Both of these options provide slightly different reporting choices. You don't *always* need to report the significance for the intercept because it isn't surprising that the line is not passing through zero. For the **`summary(model.lme)`** option the **`Value`** is an effect size and the **`Std.Error`** is the Standard Error of the effect size. The direction of effect is indicated in the left-hand column.

So, for example, SEXM means that there was a positive 6.05 g trend towards males being heavier than females. The standard error of this was ±0.48. **Because the standard error does not overlap with zero the effect is significant**. That is, the error also tells us whether the effect is significantly different from zero.

The **anova(model.lme)** option is much simpler and doesn't have effect sizes built in. report everything in red: The numDF is the numerator degrees of freedom and the denDF is the denominator degrees of freedom, as per an ANOVA. Also, because out predictors are factors (not numeric), the **anova** option might be generating more sensible results .

```
> anova(agilis.lme)
             numDF denDF   F-value p-value
(Intercept)      1    81 2490.2066  <.0001
MONTH            1    56    1.8089  0.1841
SEX              1    81  413.3465  <.0001
HABITAT          1    56    3.1368  0.0820
SEX:HABITAT      1    81   11.4031  0.0011
```

The other important bit of the model is how much variance was explained by the **random effect**s and how much was explained by the **residuals** (the fixed effects + unexplained variance that is left over). The best way to present this is as a percentage of total variance explained:

```
> VarCorr(agilis.lme)
            Variance     StdDev
YEAR =      pdLogChol(1)
(Intercept) 1.321959e-02 0.1149764591
SITE =      pdLogChol(1)
(Intercept) 8.027053e+00 2.8332053471
TRAP.GRID = pdLogChol(1)
(Intercept) 3.742499e-07 0.0006117597
Residual    5.806251e+00 2.4096164438
```

YEAR	0.013127	0.013127 / 13.8	=	0.10%
SITE	8.027108	8.027108 / 13.8	=	58.17%
TRAP.GRID	0.000000	0.000000 / 13.8	=	0.00%
Residual	5.806250	5.806250 / 13.8	=	42.07%
SUM	13.8			

The Random Effects are taken into account first and then after these have been used to explain variation in the response variable, the leftover variance (called the 'residual variance') is used for the Fixed Effects. This is how the Random Effects control for the Fixed Effects. If it turns out that actually Year, Site and Trap Grid explain all the variation in the data then there will be no variation left over for the Fixed Effects to explain.

In this case, the Random Effects are explaining in total about 60% of the variation in the response variable (mass of agile antechinus). This leaves about 40% for the Fixed Effects.

There is some code on the next page that does the same thing in R.

```
agilis.var <- VarCorr(agilis.lme)
agilis.var # look at the variances

YEAR <- as.numeric(agilis.var[2]) # grab the variance for year
SITE <-  as.numeric(agilis.var[4]) # grab the variance for site
TRAP.GRID <-  as.numeric(agilis.var[6]) # for trapping grid
RESIDUALS <-  as.numeric(agilis.var[7]) # grab the unexplained
variance that will be passed to the fixed effects

YEAR;SITE;TRAP.GRID;RESIDUALS # check the numbers
TOTAL.VARIANCE <- YEAR+SITE+TRAP.GRID+RESIDUALS # sum them

YEAR/TOTAL.VARIANCE # calculate proportion
SITE/TOTAL.VARIANCE # calculate proportion
TRAP.GRID/TOTAL.VARIANCE # calculate proportion
RESIDUALS/TOTAL.VARIANCE # calculate proportion
```

RESULT

```
> agilis.var <- (VarCorr(agilis.lme))
> agilis.var

            Variance     StdDev
YEAR =      pdLogChol(1)
(Intercept) 1.321958e-02 0.1149764269
SITE =      pdLogChol(1)
(Intercept) 8.027053e+00 2.8332053467
TRAP.GRID = pdLogChol(1)
(Intercept) 3.744421e-07 0.0006119168
Residual    5.806251e+00 2.4096164440

> YEAR <- as.numeric(agilis.var[2])
> SITE <-  as.numeric(agilis.var[4])
> TRAP.GRID <-  as.numeric(agilis.var[6])
> RESIDUALS <-  as.numeric(agilis.var[7])

> YEAR;SITE;TRAP.GRID;RESIDUALS
[1] 0.01321958
[1] 8.027053
[1] 3.744421e-07
[1] 5.806251

> TOTAL.VARIANCE <- YEAR+SITE+TRAP.GRID+RESIDUALS

> YEAR/TOTAL.VARIANCE
[1] 0.0009547219
> SITE/TOTAL.VARIANCE
[1] 0.5797161
> TRAP.GRID/TOTAL.VARIANCE
[1] 2.704232e-08
> RESIDUALS/TOTAL.VARIANCE
[1] 0.4193291
```

Table 1. Example of how to present linear mixed effect model results. Note that because there is a significant interaction of SEX x HABITAT, the next step would be to split the data by SEX and re-run the analyses to check what the effect of habitat is on the two sexes. The significant interaction term (as with regression analyses, ANOVAs) renders the main effects of SEX and HABITAT uninterpretable as they stand. Some unravelling of the complex relationship is required.

Fixed Effects	**DF (num, dem)**	**F**	**P**
Intercept	1,81	2490.2	< 0.001
MONTH	1,56	1.81	0.184
SEX	1,81	413.3	< 0.001
HABITAT	1,56	3.1	0.082
SEX x HABITAT	1,81	11.4	0.001
Random effects	**Percentage of variation explained**		
YEAR	0.1%		
SITE	58.0 %		
TRAP GRID	< 0.1%		
Residual	42.0%		

But what about AICs? What are they used for? An AIC is an information criterion used to pick the best model. This is a different approach to using *P*-values and you should probably not use AICs and P values in the same analysis.

AICs use a statistic called Deviance that is something like an R^2 to work out which model best explains the data. The AIC then penalises models with too many predictors. If you have 20 predictors and 20 data points you will have a perfect model, but it is also not very useful because each data point is explained by one predictor. Instead what you want is the most **parsimonious** model: the model that explains the most variation in the data whilst using the fewest predictors.

Instead of trying to find which predictors are significant, AICs are used to decide which model best explains the variation in the response (i.e. which predictors to include and which to leave out) given the number of predictors used. Try the following and see what happens. **Note that we have switch REML to ML because we want to compare models now instead of obtain P values…**

```
agilis.lme <- lme(MASS ~ MONTH + SEX * HABITAT, random = ~1|
YEAR/SITE/TRAP.GRID, method="ML", data = agilis)

install.packages("MuMIn")
library(MuMIn)

dredge(agilis.lme)
```

We won't get into this in detail here. If you are interested in AICs have a look at the Appendix at the model selection chapter.

nlme and lme4

If you hunt around on the various R groups you'll find that most of the stats gurus advise in favour of using **lme4** instead of **nlme** for linear mixed effect models. The package **lme4** does have more accurate algorithms, but the difference is marginal and for our purposes the **nlme** package is easier to use. However, to run the same model using lme4 you would use this code:

```
install.packages("lme4")
library(lme4)

agilis <- read.table('agilis-morphometrics.csv',header=T,sep=',')
agilis <- na.omit(agilis) # remove missing observations
str(agilis)

agilis.lmer <- lmer(MASS ~ MONTH + SEX * HABITAT +
(1| YEAR/SITE/TRAP.GRID), REML = TRUE, data = agilis)

summary(agilis.lmer) # Note: there are no P-values!
anova(agilis.lmer) # Note: there are no P-values!
```

The author who manages **lme4** feels that there isn't an accurate way to calculate P values for linear mixed effects models. This may well be true, but unless your P values are marginal (in which case you should be cautious with your interpretation anyway), and as long as you are looking at effect sizes as well, and thinking about biological meaningfulness, slightly inaccurate P values are not going to make or break your work. We will use a method out of another package, **lmerTest**.

```
install.packages("lmerTest")
library(lmerTest)

agilis.lmer <- lmer(MASS ~ MONTH + SEX * HABITAT +
(1| YEAR/SITE/TRAP.GRID), REML = TRUE, data = agilis) # Rebuild
model after having loaded the new library

summary(agilis.lmer)
anova(agilis.lmer)
```

Results are on the next page. If you compare the output, the **lme4** model has produced largely similar results except that Habitat based on **nlme** was marginally non-significant (P = 0.08), whereas for lme4 it is not significant (P = 0.005). Given that the interaction term **SEX:HABITAT** is also significant, the main effects for **SEX** and **HABITAT** are not interpretable out of this model anyway, so the difference is a bit moot.

RESULT

```
summary(agilis.lmer)

Linear mixed model fit by REML. t-tests use Satterthwaite's method
['lmerModLmerTest']
Formula: MASS ~ MONTH + SEX * HABITAT + (1 | YEAR/SITE/TRAP.GRID)
   Data: agilis

REML criterion at convergence: 1019.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.0543 -0.5810 -0.0472  0.5166  2.6355

Random effects:
 Groups                 Name        Variance Std.Dev.
 TRAP.GRID:(SITE:YEAR) (Intercept) 0.00000  0.0000
 SITE:YEAR             (Intercept) 8.02711  2.8332
 YEAR                  (Intercept) 0.01313  0.1146
 Residual                          5.80625  2.4096
Number of obs: 201, groups:  TRAP.GRID:(SITE:YEAR), 118; SITE:YEAR,
60; YEAR, 2

Fixed effects:
                Estimate Std. Error       df t value Pr(>|t|)
(Intercept)      20.2880     1.7756  52.6347  11.426 7.17e-16 ***
MONTH             0.4235     0.2925  52.3926   1.448 0.153546
SEX              -6.0498     0.4890 140.7775 -12.372  < 2e-16 ***
HABITATFRAG       2.5170     0.8728  68.6208   2.884 0.005245 **
SEX:HABITATFRAG  -2.3884     0.7073 142.4323  -3.377 0.000945 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) MONTH  SEX    HABITA
MONTH       -0.938
SEX         -0.114 -0.006
HABITATFRAG -0.241  0.002  0.243
SEX:HABITAT  0.079  0.003 -0.691 -0.366

anova(agilis.lmer)
Type III Analysis of Variance Table with Satterthwaite's method
             Sum Sq Mean Sq NumDF   DenDF  F value    Pr(>F)
MONTH         12.18   12.18     1  52.393   2.0970 0.1535457
SEX         2436.29 2436.29     1 142.533 419.5976 < 2.2e-16 ***
HABITAT       48.28   48.28     1  68.621   8.3159 0.0052451 **
SEX:HABITAT   66.21   66.21     1 142.432  11.4032 0.0009454 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Generalised linear mixed effect model

A generalised linear mixed effects model is simply an extension of the GLM we looked at earlier, but allowing for one or more random effects. Here's the code for a generalised linear mixed effect model (GLMM) using the lmer package. We'll examine whether there are differences in exoparasite counts (**EXO**) for agile antechinus using the same factors and covariates as above.

```
agilis <- read.table('agilis-morphometrics.csv',header=T,sep=',')
agilis <- na.omit(agilis) # remove missing observations
str(agilis)
```

Is the response variable a count of whole numbers?

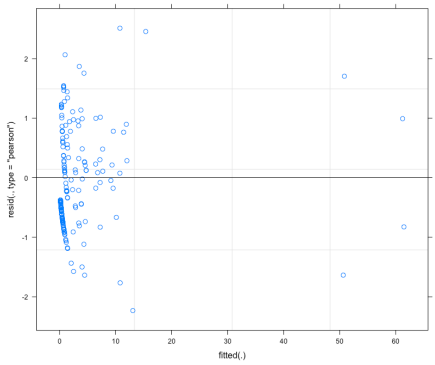
```
is.integer(agilis$EXO)
```

Load the necessary library:

```
library(lme4)
```

```
agilis.glmer <- glmer(EXO ~ SEX * HABITAT + MONTH + (1 |
YEAR/SITE/TRAP.GRID), family = poisson(link = "log"), REML =
TRUE, agilis)
```

Look at a plot of the model to check assumptions. Only one plot is generated, the residuals vs fitted plot. We want to see a roughly evenly distributed cloud of data with no wedge in it. The following looks all right.




```
summary(agilis.glmer) # preferable if most of your predictors
are numeric
anova(agilis.glmer) # preferable if most of your predictors
are factorial
```

Previously we had difficulty extracted P values from a **linear mixed effect model (lmer)** (which is a mixed effect variant of an ANOVA-like model (see above), and is a test that relies on variance). However generalised linear models are **not** tests of variance, they are **tests of deviance**.

The author of the **lme4** package thinks that obtaining P values from mixed effects GLMS is valid, and they have provided P-values for these GLMM models. This means we don't have to do the extra step of extracting P-values using another library.

RESULT

```
summary(agilis.glmer)
Generalized linear mixed model fit by maximum likelihood (Laplace
Approximation) ['glmerMod']
 Family: poisson  ( log )
Formula: EXO ~ SEX * HABITAT + MONTH + (1 | YEAR/SITE/TRAP.GRID)
   Data: agilis

     AIC      BIC   logLik deviance df.resid
   739.9    766.3   -361.9    723.9      193

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.2317 -0.6562 -0.3891  0.3737  2.5118

Random effects:
 Groups                 Name        Variance Std.Dev.
 TRAP.GRID:(SITE:YEAR) (Intercept) 0.2459   0.4959
 SITE:YEAR             (Intercept) 1.7666   1.3291
 YEAR                  (Intercept) 0.0000   0.0000
Number of obs: 201, groups:  TRAP.GRID:(SITE:YEAR), 118; SITE:YEAR,
60; YEAR, 2

Fixed effects:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.52102    0.88331  -0.590   0.5553
SEX              0.16237    0.17603   0.922   0.3563
HABITATFRAG      1.25766    0.41083   3.061   0.0022 **
MONTH           -0.04061    0.14416  -0.282   0.7782
SEX:HABITATFRAG -0.15848    0.19944  -0.795   0.4268
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
            (Intr) SEX    HABITA MONTH
SEX         -0.088
HABITATFRAG -0.278  0.199
MONTH       -0.936 -0.003  0.020
SEX:HABITAT  0.077 -0.882 -0.227  0.004

anova(agilis.glmer)
Analysis of Variance Table
            Df Sum Sq Mean Sq F value
SEX          1 0.2194  0.2194  0.2194
HABITAT      1 8.9309  8.9309  8.9309
MONTH        1 0.0783  0.0783  0.0783
SEX:HABITAT  1 0.6237  0.6237  0.6237
```

What happens if you try to get *P*-values from this formula?

```
agilis.glmer <- glmer(MASS ~ SEX * HABITAT + MONTH + (1 |
YEAR/SITE/TRAP.GRID), family = gaussian(link = "identity"),
REML = TRUE, agilis)

summary(agilis.glmer)
```

RESULT

```
summary(agilis.glmer)
Fixed effects:
                 Estimate Std. Error t value
(Intercept)       20.2880     1.7756  11.426
SEX               -6.0498     0.4890 -12.372
HABITATFRAG        2.5170     0.8728   2.884
MONTH              0.4235     0.2925   1.448
SEX:HABITATFRAG   -2.3884     0.7073  -3.377


anova(agilis.glmer)
Analysis of Variance Table
            Df  Sum Sq Mean Sq  F value
SEX          1 2398.24 2398.24 413.0450
HABITAT      1   18.11   18.11   3.1184
MONTH        1   12.36   12.36   2.1289
SEX:HABITAT  1   66.21   66.21  11.4032
```

So, what we've done here is used a GLMM to test for a relationship expecting normally distributed residuals of a model, which is defined using the code `family = gaussian(link = "identity")`.

This is the same situation as previously, where the `lme4` author does not think it is straightforward to calculate P values for linear models with mixed effects, so that what might appear to be a sneaky backdoor way of getting P values (running a GLMM with a normal distribution instead) has also had the P values intentionally omitted.

Applying a Tukey's test to a mixed effect model

Best to use the **glht** function in library **multcomp**:

```
library(multcomp)
```

Make sure your dataset is attached!

```
attach(your.data)
```

Make sure your 'factor' of interest is actually a factor in R

```
your.data$your.factor <- as.factor(your.data$your.factor)
```

Apply a Tukey test to the model

```
fit.glht <- glht(your.lme,linfct=mcp(YOUR.FACTOR="Tukey"))

plot(fit.glht) # plot of confidence intervals of differences.
These plots are usually not reported in a paper.
summary(fit.glht) # Tukey's contrasts
cld(fit.glht) # alphabet soup
```

Factorial interactions

If the lme or glmr has an interaction in it you need to include an interaction average:

```
fit.glht <- glht(your.lme,linfct=mcp(YOUR.FACTOR="Tukey",
interaction_average=TRUE))

plot(fit.glht)
summary(fit.glht)
```

Covariate interactions

If the lme or glmr has an interaction in it involving a covariate you need to include an interaction average:

```
fit.glht <- glht(your.lme,linfct=mcp(YOUR.FACTOR="Tukey",
covariate_average=TRUE))

plot(fit.glht)
summary(fit.glht)
```

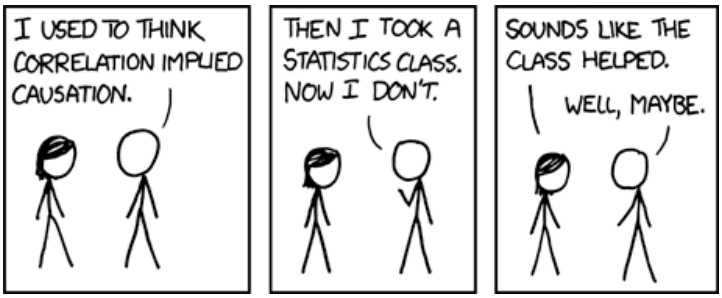
Try applying a Tukey's test to the *agilis* linear model for mass we created above. Here is the code for the model again to save you scrolling up:

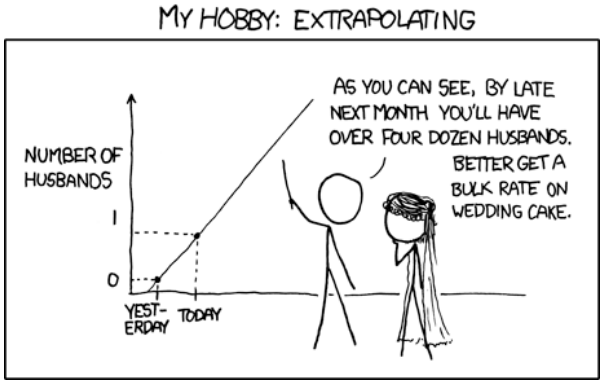
```
agilis.lme <- lmer(MASS ~ MONTH + SEX * HABITAT + (1 |
YEAR/SITE/TRAP.GRID), REML = TRUE, data = agilis)
```

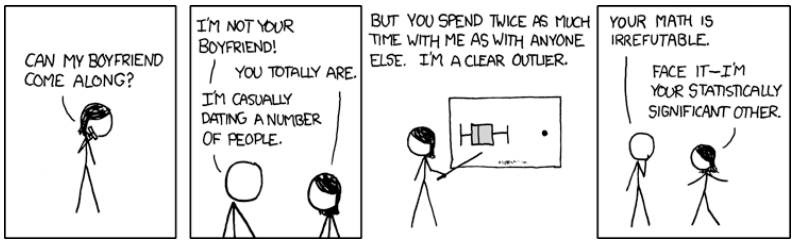
Make sure that Month is a factor (use `str(agilis)` if unsure) and attach your dataset. Results you should see are on the next page:

RESULT

```
library(multcomp)
agilis$MONTH <- as.factor(agilis$MONTH)
agilis.lme <- lmer(MASS ~ MONTH + SEX * HABITAT + (1 |
YEAR/SITE/TRAP.GRID), REML = TRUE, data = agilis)

attach(agilis)
agilis.glht <- glht(agilis.lme,linfct=mcp(MONTH="Tukey"))

summary(agilis.glht)

       Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lmer(formula = MASS ~ MONTH + SEX * HABITAT + (1 |
YEAR/SITE/TRAP.GRID),
    data = agilis, REML = TRUE)

Linear Hypotheses:
           Estimate Std. Error z value Pr(>|z|)
4 - 3 == 0  -2.7063     1.9472  -1.390    0.725
5 - 3 == 0  -1.7484     1.7719  -0.987    0.919
6 - 3 == 0  -1.1944     1.8373  -0.650    0.986
7 - 3 == 0  -0.7579     1.7964  -0.422    0.998
8 - 3 == 0   0.8455     2.0638   0.410    0.998
5 - 4 == 0   0.9578     1.3747   0.697    0.982
6 - 4 == 0   1.5118     1.4573   1.037    0.902
7 - 4 == 0   1.9484     1.4050   1.387    0.727
8 - 4 == 0   3.5518     1.7348   2.047    0.306
6 - 5 == 0   0.5540     1.2136   0.456    0.997
7 - 5 == 0   0.9905     1.1507   0.861    0.954
8 - 5 == 0   2.5939     1.5349   1.690    0.528
7 - 6 == 0   0.4366     1.2489   0.350    0.999
8 - 6 == 0   2.0399     1.6101   1.267    0.796
8 - 7 == 0   1.6034     1.5636   1.025    0.906
(Adjusted p values reported -- single-step method)

cld(agilis.glht) # alphabet soup
  3   4   5   6   7   8
"a" "a" "a" "a" "a" "a"
```


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.


MY HOBBY: EXTRAPOLATING
NUMBER OF HUSBANDS
1
0
YEST-ERDAY
TODAY
AS YOU CAN SEE, BY LATE NEXT MONTH YOU'LL HAVE OVER FOUR DOZEN HUSBANDS.
BETTER GET A BULK RATE ON WEDDING CAKE.


CAN MY BOYFRIEND COME ALONG?
I'M NOT YOUR BOYFRIEND!
YOU TOTALLY ARE.
I'M CASUALLY DATING A NUMBER OF PEOPLE.
BUT YOU SPEND TWICE AS MUCH TIME WITH ME AS WITH ANYONE ELSE. I'M A CLEAR OUTLIER.
YOUR MATH IS IRREFUTABLE.
FACE IT—I'M YOUR STATISTICALLY SIGNIFICANT OTHER.

Advanced Graphics

Graphically speaking, there's a lot of things you can play around with in R, and once you know what you are doing R will produce some really beautiful graphics for you. We'll get into a bit of a taster of some of these features. A lot of this chapter is adapted from *R in Action* by Robert I Kabacoff, which is an excellent source for information on how to produce attractive graphs and figures in R.

par

The 'par' command is used to reset the global settings for figures. We've been using it to reset plot spaces to 2x2 instead of the usual 1x1 setting. It can be used to reset other things too, but before you change global settings you should save the default. Otherwise to restore the default you have to close R and re-open it again.

View the current settings

```
par()
```

Save the current settings (can be an important step!)

```
save.par <- par()
```

Restore the saved settings

```
par(save.par)
```

To change a figure using the par settings you need to reset par before running the figure. We'll have a look at this on the next page.

Import today's data:

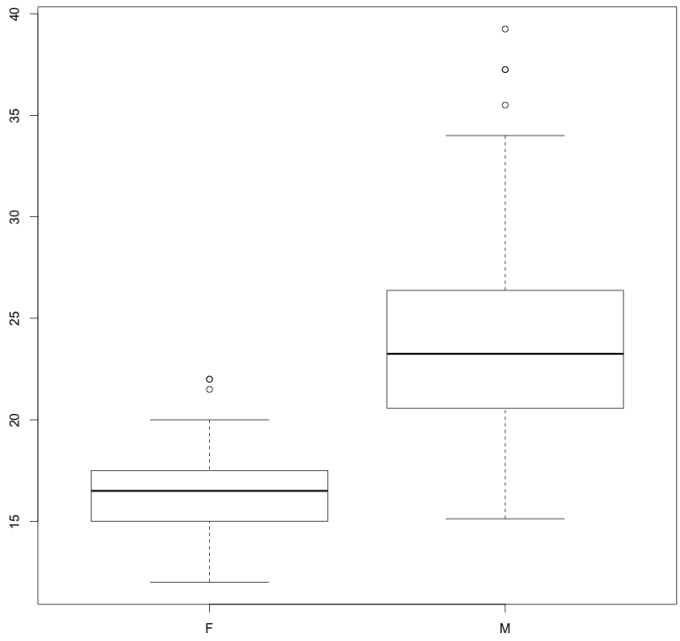
```
agilis <- read.table('agilis-
morphometrics.csv',header=T,sep=',')

agilis <- na.omit(agilis)
str(agilis)
```

Aggressively restoring graphing defaults

You can use the following code to aggressively restore plotting defaults, but, keep in mind this will erase all your previous plots as well.

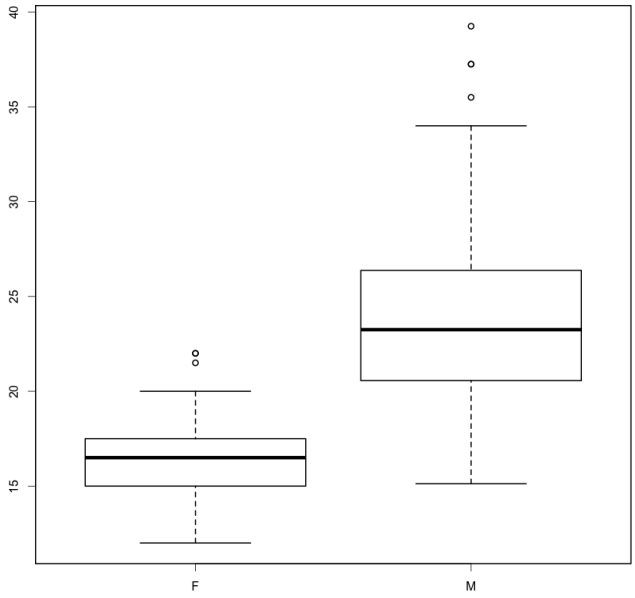
```
dev.off()
```

```
boxplot(MASS~SEX, data = agilis)
```



Set the default line width to 2

```
par(lwd = 2)

boxplot(MASS~SEX, data = agilis)
```



Because the par is now set to line width 2, all plots you create will have line width 2. If you want to reset this, restore the old par, like this:

```
par(save.par)
```

What can be changed with par?

Type **`?par`** to see the list of parameters that can be altered. Here are some of the most common things you may want to change:

Text size & font

`cex =` Text size . 1=default. 1.5 = 50% larger. 0.5 = 50% smaller, etc.

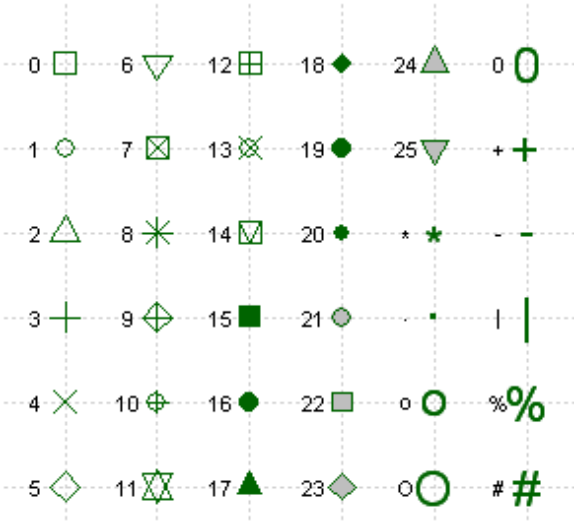
`family =` Font family to use. **`"serif"`**, **`"sans"`**, **`"mono"`** are most common.

`font =` 1=plain. 2=bold. 3=italic. 4=bold italic.

`las =` Style of axis labels. 0 = parallel to the axis [default]. 1 = always horizontal. 2 = always perpendicular to the axis. 3 = always vertical.

Point symbols

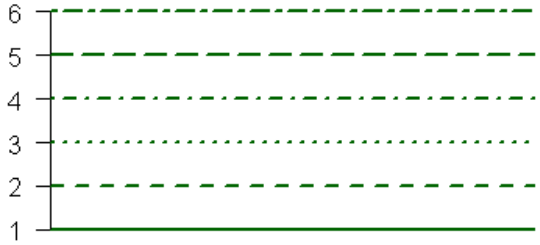
`pch =` Use the following chart to see what points are available. For example, **`par(pch = 1)`** will provide hollow circles.



Lines

`lwd =` Line width. 1 = default. 2 = 2x thickness. 3 = 3x thickness.

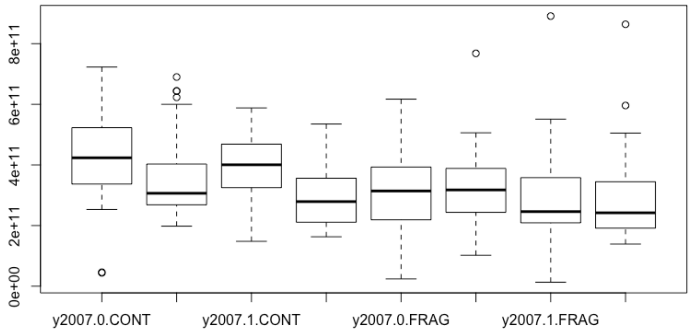
`lty =` Line type. 1 = solid. 2 through 6 are dashed.



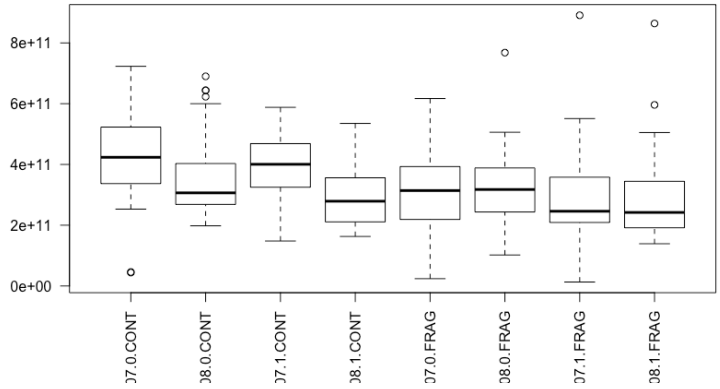
Margins

If labels or legends are cut-off, you can adjust margins of plotting areas.

```
boxplot(WBC~YEAR+SEX+HABITAT,data=agilis)
```

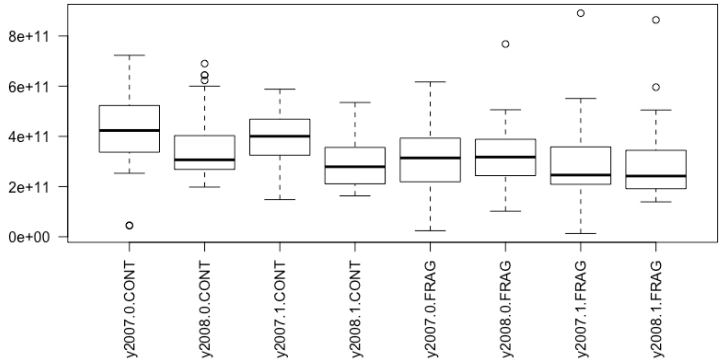



```
boxplot(WBC~YEAR+SEX+HABITAT,data=agilis, las=2)
```



Change the bottom margin to 6 character spaces (making it bigger, allowing for more room). The numbers are bottom, left, top and right.

```
par(mar=c(10,4,2,2))
boxplot(WBC~YEAR+SEX+HABITAT,data=agilis, las=2)
```



Colours

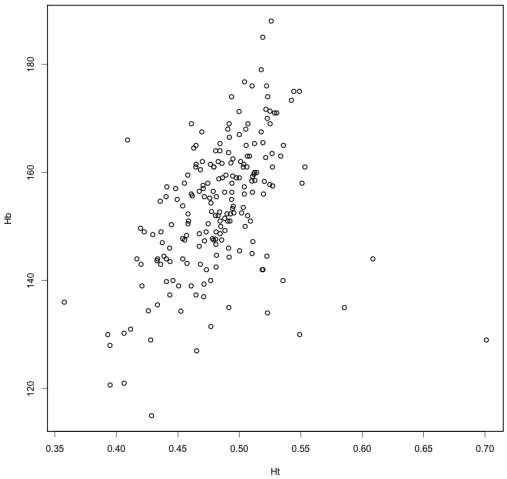
`col =` Default plotting colour. Some functions, such as lines, accept lists of colours that are 'recycled'

`col = c("grey", "white")` Will start with grey, and cycle to white, grey, white, grey etc.

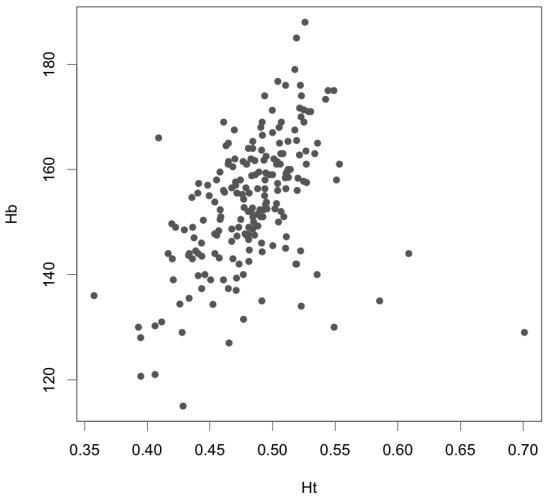
`col = c("darkgrey", "grey", "white")` Will start with dark grey, then produce grey, and cycle to white, dark grey, grey, white, dark grey etc.

Colours can be denoted using numbers or names or colour codes:

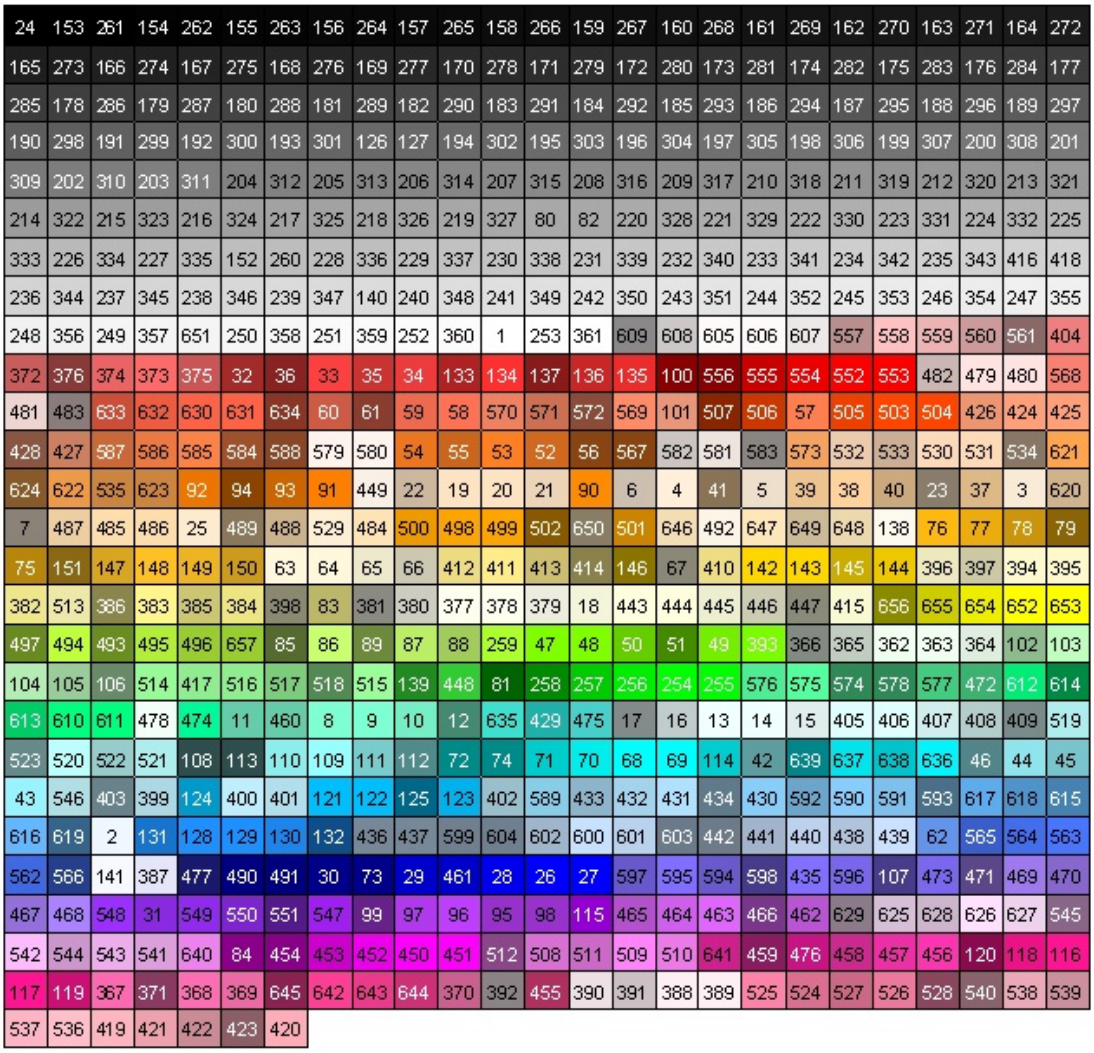
```
par(save.par) ### some defaults won't reset. This is ok ###
plot(Hb~Ht, data = agilis)
```




```
par(pch = 16)
par(col = "dimgrey")
par(cex = 1.5)
par(lwd = 2)
plot(Hb~Ht, data = agilis)
```



1	white	#FFFFFF	255	255	255
2	aliceblue	#F0F8FF	240	248	255
3	antiquewhite	#FAEBD7	250	235	215
4	antiquewhite1	#FFEFDB	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8378	139	131	120
8	aquamarine	#7FFFD4	127	255	212
9	aquamarine1	#7FFFD4	127	255	212
10	aquamarine2	#76EEC6	118	238	198
11	aquamarine3	#66CDAA	102	205	170
12	aquamarine4	#458B74	69	139	116
13	azure	#F0FFFF	240	255	255
14	azure1	#F0FFFF	240	255	255
15	azure2	#E0EEEE	224	238	239
16	azure3	#C1CDCD	193	205	205
17	azure4	#838B8B	131	139	139
18	beige	#F5F5DC	245	245	220
19	bisque	#FFE4C4	255	228	196
20	bisque1	#FFE4C4	255	228	196
21	bisque2	#EED5B7	238	213	183
22	bisque3	#CDB79E	205	183	158
23	bisque4	#8B7D6B	139	125	107
24	black	#000000	0	0	0
25	blanchedalmond	#FFEBCD	255	235	205
26	blue	#0000FF	0	0	255
27	blue1	#0000FF	0	0	255
28	blue2	#0000EE	0	0	238
29	blue3	#0000CD	0	0	205
30	blue4	#00008B	0	0	139
31	blueviolet	#8A2BE2	138	43	226
32	brown	#A52A2A	165	42	42
33	brown1	#FF4040	255	64	64
34	brown2	#EE3B3B	238	59	59
35	brown3	#CD3333	205	51	51
36	brown4	#8B2323	139	35	35
37	burlywood	#DEB887	222	184	135
38	burlywood1	#FFD39B	255	211	155
39	burlywood2	#EEC591	238	197	145
40	burlywood3	#CDAA7D	205	170	125
41	burlywood4	#8B7355	139	115	85
42	cadetblue	#5F9EA0	95	158	160
43	cadetblue1	#98F5FF	152	245	255
44	cadetblue2	#8EE5EE	142	229	238
45	cadetblue3	#7AC5CD	122	197	205
46	cadetblue4	#53868B	83	134	139
47	chartreuse	#7FFF00	127	255	0
48	chartreuse1	#7FFF00	127	255	0
49	chartreuse2	#76EE00	118	238	0
50	chartreuse3	#66CD00	102	205	0
51	chartreuse4	#458B00	69	139	0
52	chocolate	#D2691E	210	105	30
53	chocolate1	#FF7F24	255	127	36
54	chocolate2	#EE7621	238	118	33
55	chocolate3	#CD661D	205	102	29
56	chocolate4	#8B4513	139	69	19
57	coral	#FF7F50	255	127	80
58	coral1	#FF7256	255	114	86
59	coral2	#EE6A50	238	106	80
60	coral3	#CD5B45	205	91	69
61	coral4	#8B3E2F	139	62	47
62	cornflowerblue	#6495ED	100	149	237
63	cornsilk	#FFF8DC	255	248	220
64	cornsilk1	#FFF8DC	255	248	220
65	cornsilk2	#EEE8CD	238	232	205
66	cornsilk3	#CDC8B1	205	200	177
67	cornsilk4	#8B8878	139	136	120
68	cyan	#00FFFF	0	255	255
69	cyan1	#00FFFF	0	255	255
70	cyan2	#00EEEE	0	238	238
71	cyan3	#00CDCD	0	205	205
72	cyan4	#008B8B	0	139	139
73	darkblue	#00008B	0	0	139
74	darkcyan	#008B8B	0	139	139
75	darkgoldenrod	#B8860B	184	134	11
76	darkgoldenrod1	#FFB90F	255	185	15
77	darkgoldenrod2	#EEAD0E	238	173	14
78	darkgoldenrod3	#CD950C	205	149	12
79	darkgoldenrod4	#8B6508	139	101	8
80	darkgray	#A9A9A9	169	169	169
81	darkgreen	#006400	0	100	0
82	darkgrey	#A9A9A9	169	169	169
83	darkkhaki	#BDB76B	189	183	107
84	darkmagenta	#8B008B	139	0	139
85	darkolivegreen	#556B2F	85	107	47
86	darkolivegreen1	#CAFF70	202	255	112
87	darkolivegreen2	#BCEE68	188	238	104
88	darkolivegreen3	#A2CD5A	162	205	90
89	darkolivegreen4	#6E8B3D	110	139	61
90	darkorange	#FF8C00	255	140	0
91	darkorange1	#FF7F00	255	127	0
92	darkorange2	#EE7600	238	118	0
93	darkorange3	#CD6600	205	102	0
94	darkorange4	#8B4500	139	69	0
95	darkorchid	#9932CC	153	50	204
96	darkorchid1	#BF3EFF	191	62	255
97	darkorchid2	#B23AEE	178	58	238
98	darkorchid3	#9A32CD	154	50	205
99	darkorchid4	#68228B	104	34	139
100	darkred	#8B0000	139	0	0


24 153 261 154 262 155 263 156 264 157 265 158 266 159 267 160 268 161 269 162 270 163 271 164 272
165 273 166 274 167 275 168 276 169 277 170 278 171 279 172 280 173 281 174 282 175 283 176 284 177
285 178 286 179 287 180 288 181 289 182 290 183 291 184 292 185 293 186 294 187 295 188 296 189 297
190 298 191 299 192 300 193 301 126 127 194 302 195 303 196 304 197 305 198 306 199 307 200 308 201
309 202 310 203 311 204 312 205 313 206 314 207 315 208 316 209 317 210 318 211 319 212 320 213 321
214 322 215 323 216 324 217 325 218 326 219 327 80 82 220 328 221 329 222 330 223 331 224 332 225
333 226 334 227 335 152 260 228 336 229 337 230 338 231 339 232 340 233 341 234 342 235 343 416 418
236 344 237 345 238 346 239 347 140 240 348 241 349 242 350 243 351 244 352 245 353 246 354 247 355
248 356 249 357 651 250 358 251 359 252 360 1 253 361 609 608 605 606 607 557 558 559 560 561 404
372 376 374 373 375 32 36 33 35 34 133 134 137 136 135 100 556 555 554 552 553 482 479 480 568
481 483 633 632 630 631 634 60 61 59 58 570 571 572 569 101 507 506 57 505 503 504 426 424 425
428 427 587 586 585 584 588 579 580 54 55 53 52 56 567 582 581 583 573 532 533 530 531 534 621
624 622 535 623 92 94 93 91 449 22 19 20 21 90 6 4 41 5 39 38 40 23 37 3 620
7 487 485 486 25 489 488 529 484 500 498 499 502 650 501 646 492 647 649 648 138 76 77 78 79
75 151 147 148 149 150 63 64 65 66 412 411 413 414 146 67 410 142 143 145 144 396 397 394 395
382 513 386 383 385 384 398 83 381 380 377 378 379 18 443 444 445 446 447 415 656 655 654 652 653
497 494 493 495 496 657 85 86 89 87 88 259 47 48 50 51 49 393 366 365 362 363 364 102 103
104 105 106 514 417 516 517 518 515 139 448 81 258 257 256 254 255 576 575 574 578 577 472 612 614
613 610 611 478 474 11 460 8 9 10 12 635 429 475 17 16 13 14 15 405 406 407 408 409 519
523 520 522 521 108 113 110 109 111 112 72 74 71 70 68 69 114 42 639 637 638 636 46 44 45
43 546 403 399 124 400 401 121 122 125 123 402 589 433 432 431 434 430 592 590 591 593 617 618 615
616 619 2 131 128 129 130 132 436 437 599 604 602 600 601 603 442 441 440 438 439 62 565 564 563
562 566 141 387 477 490 491 30 73 29 461 28 26 27 597 595 594 598 435 596 107 473 471 469 470
467 468 548 31 549 550 551 547 99 97 96 95 98 115 465 464 463 466 462 629 625 628 626 627 545
542 544 543 541 640 84 454 453 452 450 451 512 508 511 509 510 641 459 476 458 457 456 120 118 116
117 119 367 371 368 369 645 642 643 644 370 392 455 390 391 388 389 525 524 527 526 528 540 538 539
537 536 419 421 422 423 420


cornsilk3 dodgerblue4 gray45 grey3 grey69 lemonchiffon2 mediumorchid paleviolletred4 slateblue
cornsilk2 dodgerblue3 gray44 grey2 grey68 lemonchiffon1 mediumblue paleviolletred3 skyblue4
cornsilk1 dodgerblue2 gray43 grey1 grey67 lemonchiffon mediumaquamarine paleviolletred2 skyblue3
cornsilk dodgerblue1 gray42 grey0 grey66 lawngreen maroon4 paleviolletred1 skyblue2 yellowgreen
cornflowerblue dodgerblue gray41 grey grey65 lavenderblush4 maroon3 paleviolletred skyblue1 yellow4
coral4 dimgrey gray40 greenyellow grey64 lavenderblush3 maroon2 paleturquoise4 skyblue yellow3
coral3 dimgray gray39 green4 grey63 lavenderblush2 maroon1 paleturquoise3 sienna4 yellow2
coral2 deepskyblue4 gray38 green3 grey62 lavenderblush1 maroon paleturquoise2 sienna3 yellow1
coral1 deepskyblue3 gray37 green2 grey61 lavenderblush magenta4 paleturquoise1 sienna2 yellow
coral deepskyblue2 gray36 green1 grey60 lavender magenta3 paleturquoise sienna1 whitesmoke
chocolate4 deepskyblue1 gray35 green grey59 khaki4 magenta2 palegreen4 sienna wheat4
chocolate3 deepskyblue gray34 gray100 grey58 khaki3 magenta1 palegreen3 seashell4 wheat3
chocolate2 deeppink4 gray33 gray99 grey57 khaki2 magenta palegreen2 seashell3 wheat2
chocolate1 deeppink3 gray32 gray98 grey56 khaki1 linen palegreen1 seashell2 wheat1
chocolate deeppink2 gray31 gray97 grey55 khaki limegreen palegreen seashell1 wheat
chartreuse4 deeppink1 gray30 gray96 grey54 ivory4 lightyellow4 palegoldenrod seashell violetred4
chartreuse3 deeppink gray29 gray95 grey53 ivory3 lightyellow3 orchid4 seagreen4 violetred3
chartreuse2 darkviolet gray28 gray94 grey52 ivory2 lightyellow2 orchid3 seagreen3 violetred2
chartreuse1 darkturquoise gray27 gray93 grey51 ivory1 lightyellow1 orchid2 seagreen2 violetred1
chartreuse darkslategrey gray26 gray92 grey50 ivory lightyellow orchid1 seagreen1 violetred
cadetblue4 darkslategray4 gray25 gray91 grey49 indianred4 lightsteelblue4 orchid seagreen violet
cadetblue3 darkslategray3 gray24 gray90 grey48 indianred3 lightsteelblue3 orangered4 sandybrown turquoise4
cadetblue2 darkslategray2 gray23 gray89 grey47 indianred2 lightsteelblue2 orangered3 salmon4 turquoise3
cadetblue1 darkslategray1 gray22 gray88 grey46 indianred1 lightsteelblue1 orangered2 salmon3 turquoise2
cadetblue darkslategray gray21 gray87 grey45 indianred lightsteelblue orangered1 salmon2 turquoise1
burlywood4 darkslateblue gray20 gray86 grey44 hotpink4 lightslategrey orangered salmon1 turquoise
burlywood3 darkseagreen4 gray19 gray85 grey43 hotpink3 lightslategray orange4 salmon tomato4
burlywood2 darkseagreen3 gray18 gray84 grey42 hotpink2 lightslateblue orange3 saddlebrown tomato3
burlywood1 darkseagreen2 gray17 gray83 grey41 hotpink1 lightskyblue4 orange2 royalblue4 tomato2
burlywood darkseagreen1 gray16 gray82 grey40 hotpink lightskyblue3 orange1 royalblue3 tomato1
brown4 darkseagreen gray15 gray81 grey39 honeydew4 lightskyblue2 orange royalblue2 tomato
brown3 darksalmon gray14 gray80 grey38 honeydew3 lightskyblue1 olivedrab4 royalblue1 thistle4
brown2 darkred gray13 gray79 grey37 honeydew2 lightskyblue olivedrab3 royalblue thistle3
brown1 darkorchid4 gray12 gray78 grey36 honeydew1 lightseagreen olivedrab2 rosybrown4 thistle2
brown darkorchid3 gray11 gray77 grey35 honeydew lightsalmon4 olivedrab1 rosybrown3 thistle1
blueviolet darkorchid2 gray10 gray76 grey34 grey100 lightsalmon3 olivedrab rosybrown2 thistle
blue4 darkorchid1 gray9 gray75 grey33 grey99 lightsalmon2 oldlace rosybrown1 tan4
blue3 darkorchid gray8 gray74 grey32 grey98 lightsalmon1 navyblue rosybrown tan3
blue2 darkorange4 gray7 gray73 grey31 grey97 lightsalmon navy red4 tan2
blue1 darkorange3 gray6 gray72 grey30 grey96 lightpink4 navajowhite4 red3 tan1
blue darkorange2 gray5 gray71 grey29 grey95 lightpink3 navajowhite3 red2 tan
blanchedalmond darkorange1 gray4 gray70 grey28 grey94 lightpink2 navajowhite2 red1 steelblue4
black darkorange gray3 gray69 grey27 grey93 lightpink1 navajowhite1 red steelblue3
bisque4 darkolivegreen4 gray2 gray68 grey26 grey92 lightpink navajowhite purple4 steelblue2
bisque3 darkolivegreen3 gray1 gray67 grey25 grey91 lightgrey moccasin purple3 steelblue1
bisque2 darkolivegreen2 gray0 gray66 grey24 grey90 lightgreen mistyrose4 purple2 steelblue
bisque1 darkolivegreen1 gray gray65 grey23 grey89 lightgray mistyrose3 purple1 springgreen4
bisque darkolivegreen goldenrod4 gray64 grey22 grey88 lightgoldenrodyellow mistyrose2 purple springgreen3
beige darkmagenta goldenrod3 gray63 grey21 grey87 lightgoldenrod4 mistyrose1 powderblue springgreen2
azure4 darkkhaki goldenrod2 gray62 grey20 grey86 lightgoldenrod3 mistyrose plum4 springgreen1
azure3 darkgrey goldenrod1 gray61 grey19 grey85 lightgoldenrod2 mintcream plum3 springgreen
azure2 darkgreen goldenrod gray60 grey18 grey84 lightgoldenrod1 midnightblue plum2 snow4
azure1 darkgray gold4 gray59 grey17 grey83 lightgoldenrod mediumvioletred plum1 snow3
azure darkgoldenrod4 gold3 gray58 grey16 grey82 lightcyan4 mediumturquoise plum snow2
aquamarine4 darkgoldenrod3 gold2 gray57 grey15 grey81 lightcyan3 mediumspringgreen pink4 snow1
aquamarine3 darkgoldenrod2 gold1 gray56 grey14 grey80 lightcyan2 mediumslateblue pink3 snow
aquamarine2 darkgoldenrod1 gold gray55 grey13 grey79 lightcyan1 mediumseagreen pink2 slategrey
aquamarine1 darkgoldenrod ghostwhite gray54 grey12 grey78 lightcyan mediumpurple4 pink1 slategray4
aquamarine darkcyan gainsboro gray53 grey11 grey77 lightcoral mediumpurple3 pink slategray3
antiquewhite4 darkblue forestgreen gray52 grey10 grey76 lightblue4 mediumpurple2 peru slategray2
antiquewhite3 cyan4 floralwhite gray51 grey9 grey75 lightblue3 mediumpurple1 peachpuff4 slategray1
antiquewhite2 cyan3 firebrick4 gray50 grey8 grey74 lightblue2 mediumpurple peachpuff3 slategray
antiquewhite1 cyan2 firebrick3 gray49 grey7 grey73 lightblue1 mediumorchid4 peachpuff2 slateblue4
antiquewhite cyan1 firebrick2 gray48 grey6 grey72 lightblue mediumorchid3 peachpuff1 slateblue3
aliceblue cyan firebrick1 gray47 grey5 grey71 lemonchiffon4 mediumorchid2 peachpuff slateblue2
white cornsilk4 firebrick gray46 grey4 grey70 lemonchiffon3 mediumorchid1 papayawhip slateblue1

When is it okay to use colour in a plot?

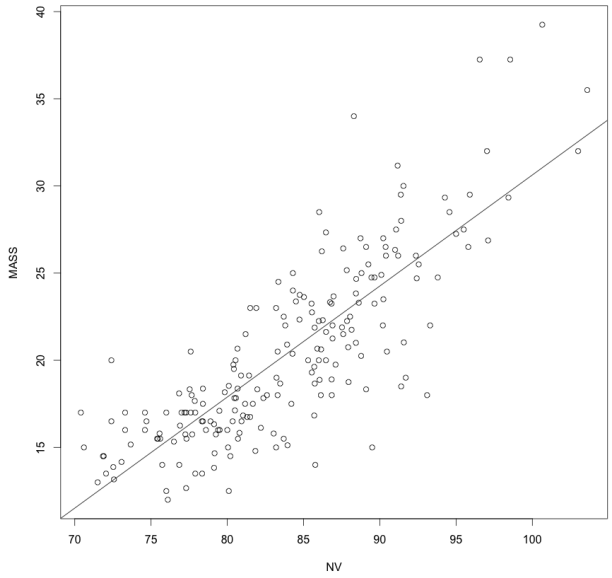
There is still a tendency in science to avoid using colour in written reports. However, it is acceptable and sometimes desirable to use colour in a figure for a scientific poster or a seminar presentation, and colour in reports is becoming more acceptable. The most sensible thing is to check with your unit co-ordinator or lecturer, and if in doubt, just stick to greyscale figures for written papers.

Adding directly to plots without using 'par'

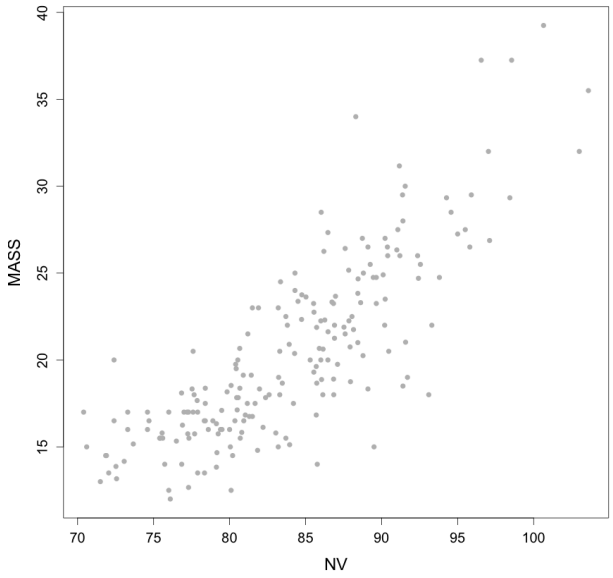
Most plots will accept **par** instructions added onto the end of their code. Each instruction is separated with a comma and the **par** command itself isn't required. This allows you to change features of a graph without changing the defaults.

Restore defaults

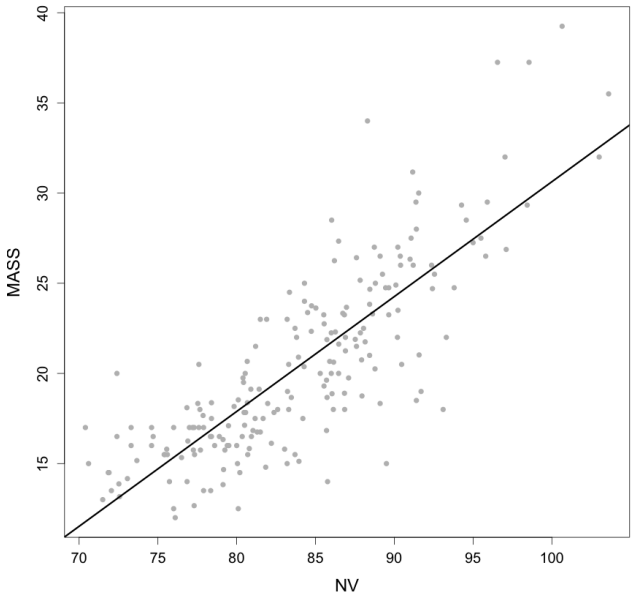
```
par(save.par)

plot(MASS~NV, data = agilis)

abline(lm(MASS~NV, data = agilis))
```



```
plot(MASS~NV, data = agilis, pch = 16, col = "grey", cex.axis =
1.25, cex.lab = 1.5)
```




```
abline(lm(MASS~NV, data = agilis), lwd = 3)
```

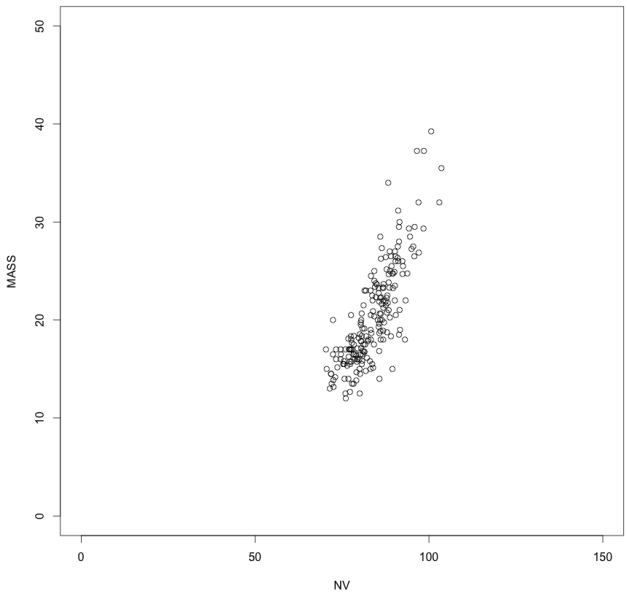


Setting axes ranges

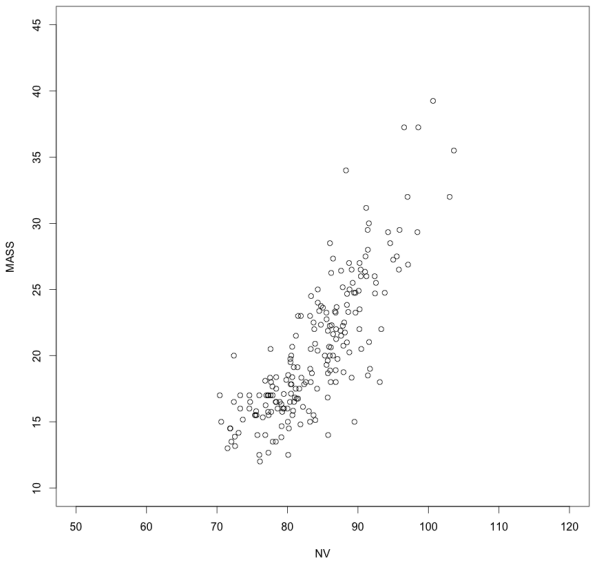
You can use **`xlim`** and **`ylim`** to adjust axis lengths.

```
xlim=c(xmin, xmax)
ylim=c(ymin, ymax)
```

```
plot(MASS~NV, data = agilis, xlim=c(0, 150), ylim=c(0, 50))
```



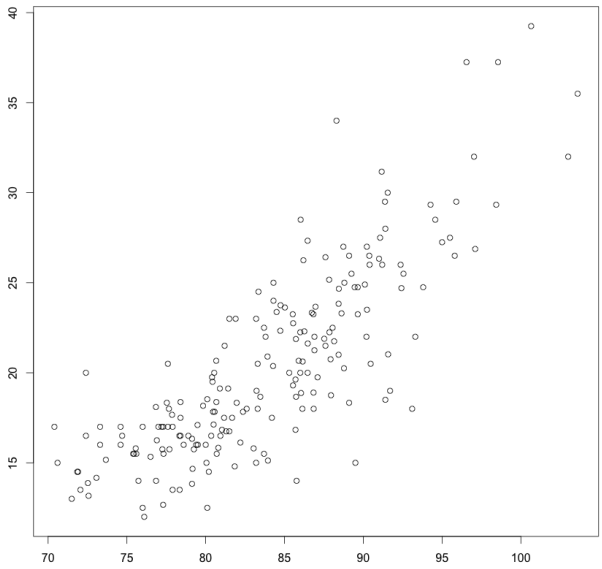

```
plot(MASS~NV, data = agilis, xlim=c(50, 120), ylim=c(10, 45))
```



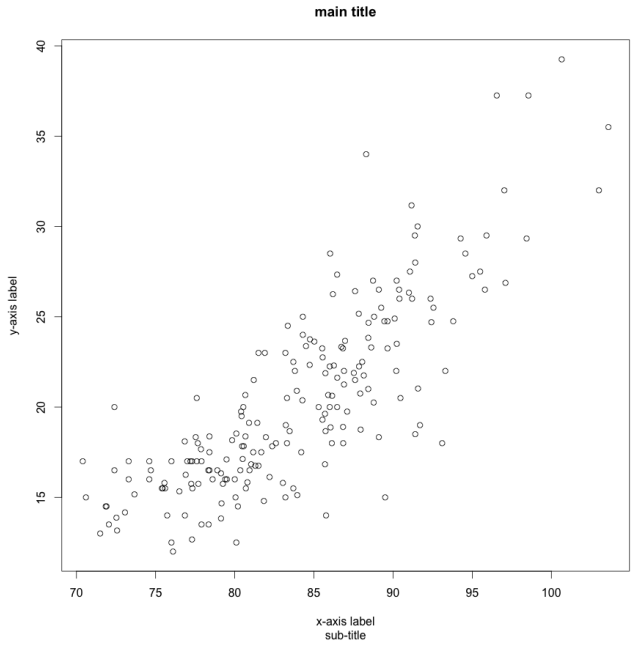
Axis labels and titles

You can usually include instructions like **ylab** and **xlab** to apply labels and titles to a graph by adding them as code to the end of the plotting code. You can also apply title and axis labels after already making a graph, but if you do you need to set the graph labels to "" when you plot it or else you will end up with letters layered on top of one-another.

```
plot(MASS~NV, data = agilis, xlab = "", ylab = "")
```



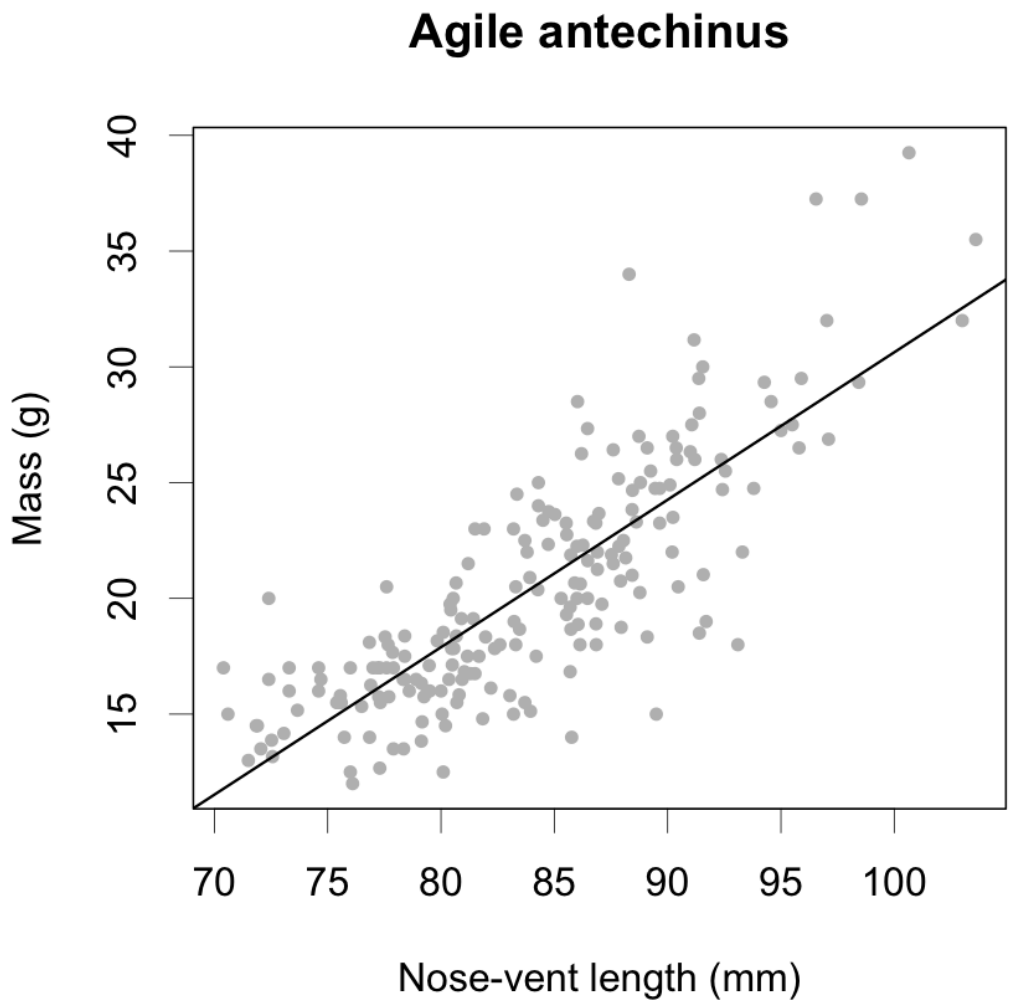

```
title(main="main title", sub="sub-title",  xlab="x-axis label",
ylab="y-axis label")
```



Graphics: putting it all together

Let's have a go at merging these all into a single graph.

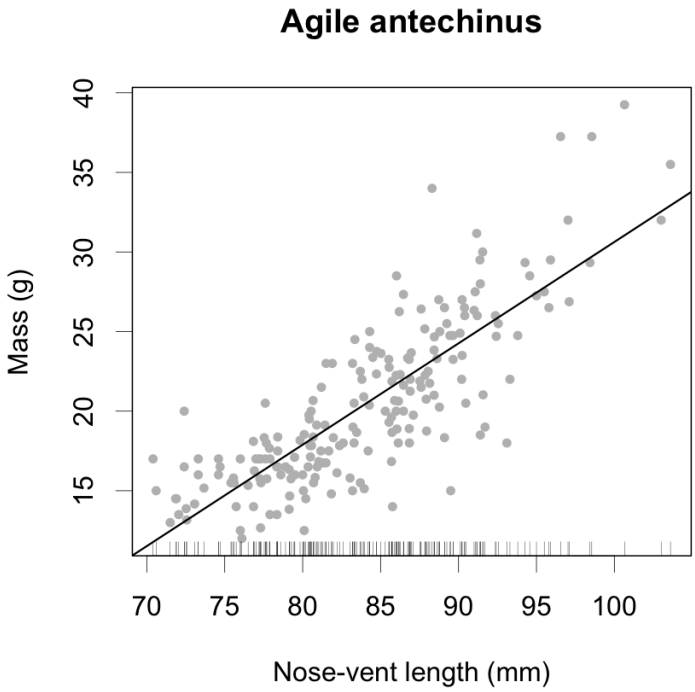
```
par(lwd=2)
par(cex=2)

plot(MASS~NV, data = agilis, pch = 16, col = "grey", xlab =
"", ylab = "", cex=0.75)

abline(lm(MASS~NV, data = agilis), lwd = 3)

title(main="Agile antechinus", xlab="Nose-vent length (mm)",
ylab="Mass (g)")
```



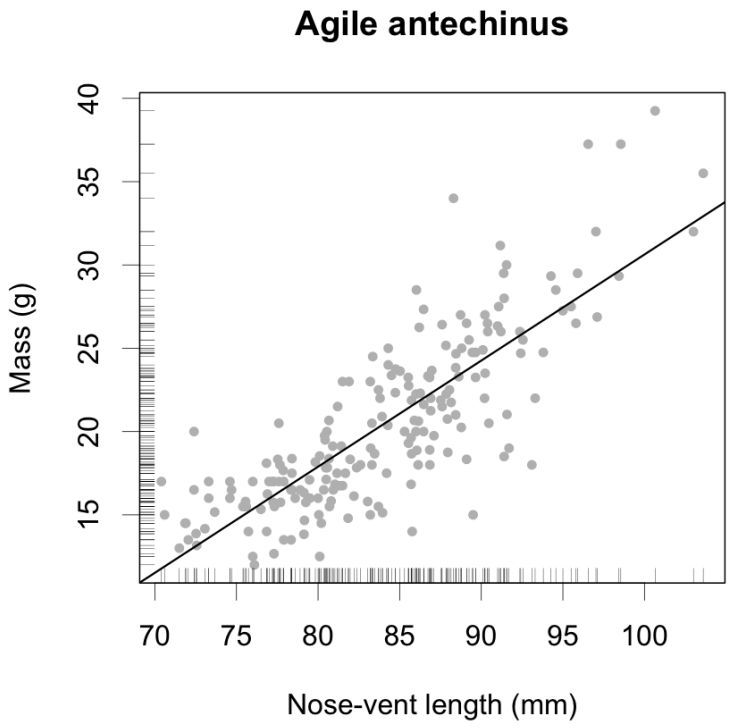
Adding axis information

It is possible to add other elements to a graph after building it. A rugplot can be added to the bottom of a graph.

```
rug(agilis$NV, col="black", side = 1)
```




```
rug(agilis$MASS, col="black", side = 2)
```



Play around with the plotting window to add boxplots to the axes of a scatterplot.

Restore defaults you saved earlier

```
par(save.par)
```

Tell R to set the plot window to one large and two small plots

```
par(fig=c(0,0.8,0,0.8), new=TRUE)
```

Scatterplot

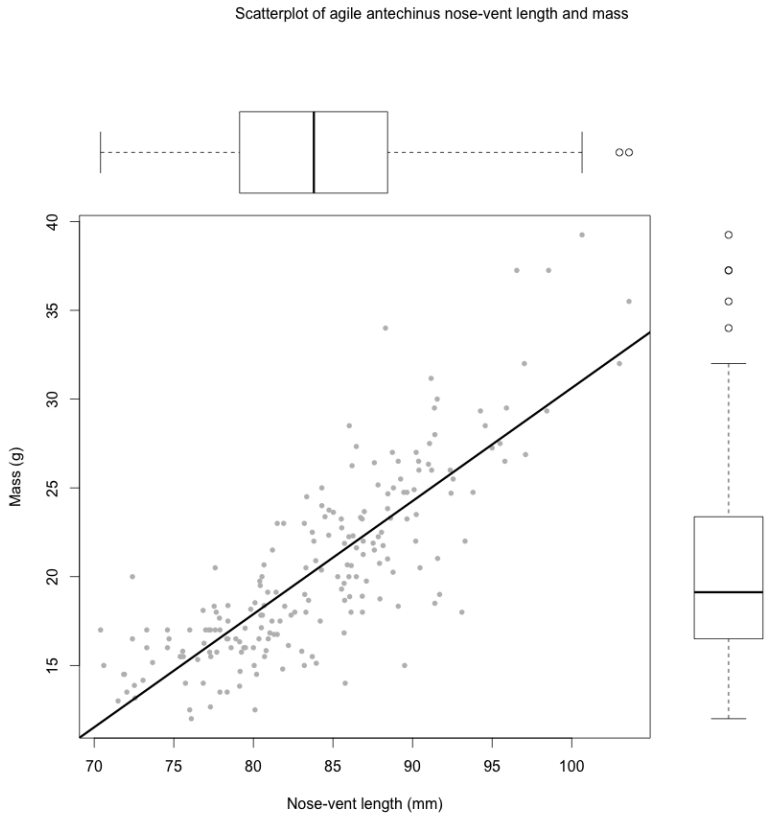
```
plot(MASS~NV, data = agilis, pch = 16, col = "grey", xlab =
"", ylab = "", cex=0.75)

abline(lm(MASS~NV, data = agilis), lwd = 3)

title(xlab="Nose-vent length (mm)", ylab="Mass (g)")
```

Boxplot on y-axis

```
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(agilis$NV, horizontal=TRUE, axes=FALSE)
```

Boxplot on x-axis

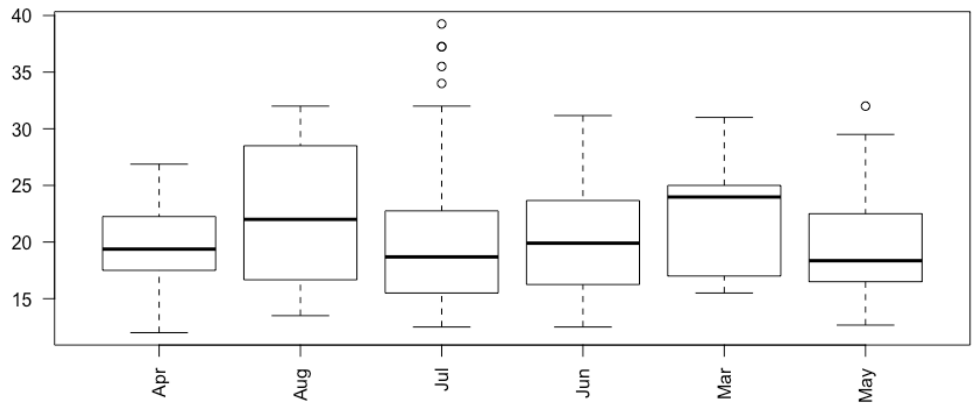
```
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(agilis$MASS, axes=FALSE)

mtext("Scatterplot of agile antechinus nose-vent length and
mass", side=3, outer=TRUE, line=-3)
```



Reordering categories on the axes for boxplots

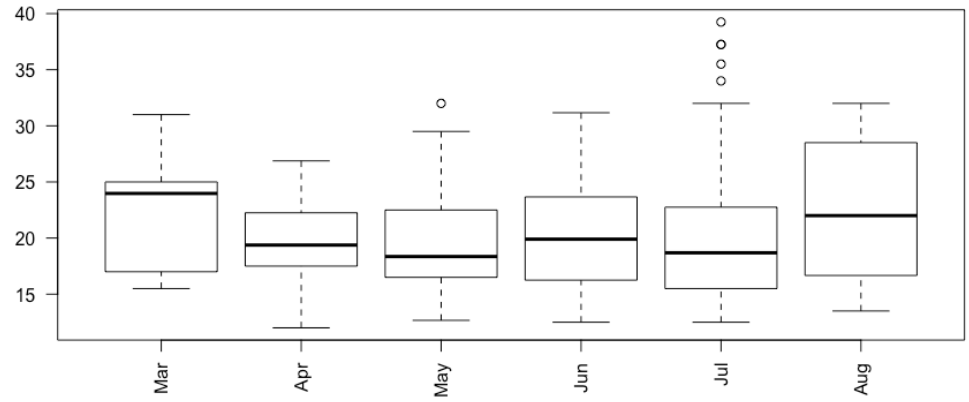
Sometimes we need to reorder the labels on a plot because the alphabetical order isn't sensible. You actually run a bit of code to reorder the factor itself, then re-run the graph. You can also re-arrange boxplot horizontal axis categories. Try generating a boxplot of MASS by month:

```
boxplot(MASS~Month, las = 2, data=agilis)
```



R defaults to arranging the categories alphabetically, which is not especially useful for months of the year. To get around this we'll need to tell R what order we want the months to fall into. Enter the variables in the order you want them to plot:

```
agilis$Month<-factor(agilis$Month, c("Mar", "Apr", "May",
"Jun", "Jul", "Aug"))

boxplot(MASS~Month, las = 2, data=agilis)
```



Setting up plots in grids and adding letters

Sometimes we want to present several individual plots as a single plot using letters a, b, c, d etc. We can set the rows and columns for the plotting window by using **par**.

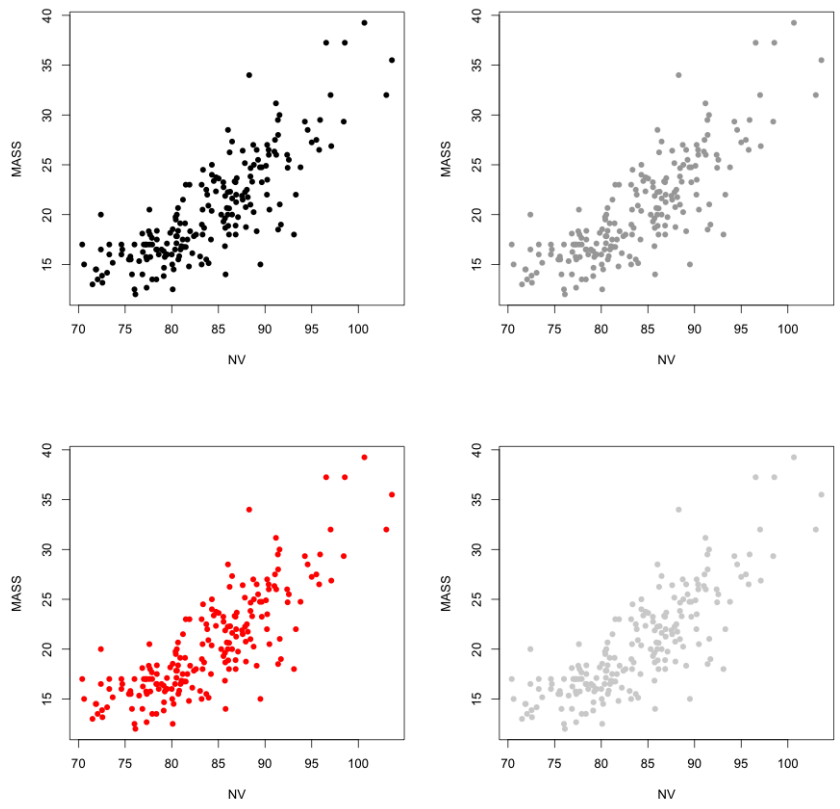
You may have to use the 'clear plots' button in your plot window to get some of these plots to work. Otherwise you might end up plotting on top of old plots.

A normal plot space with one graph

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

plot(MASS~NV, data = agilis, pch = 16, col = "grey")
```

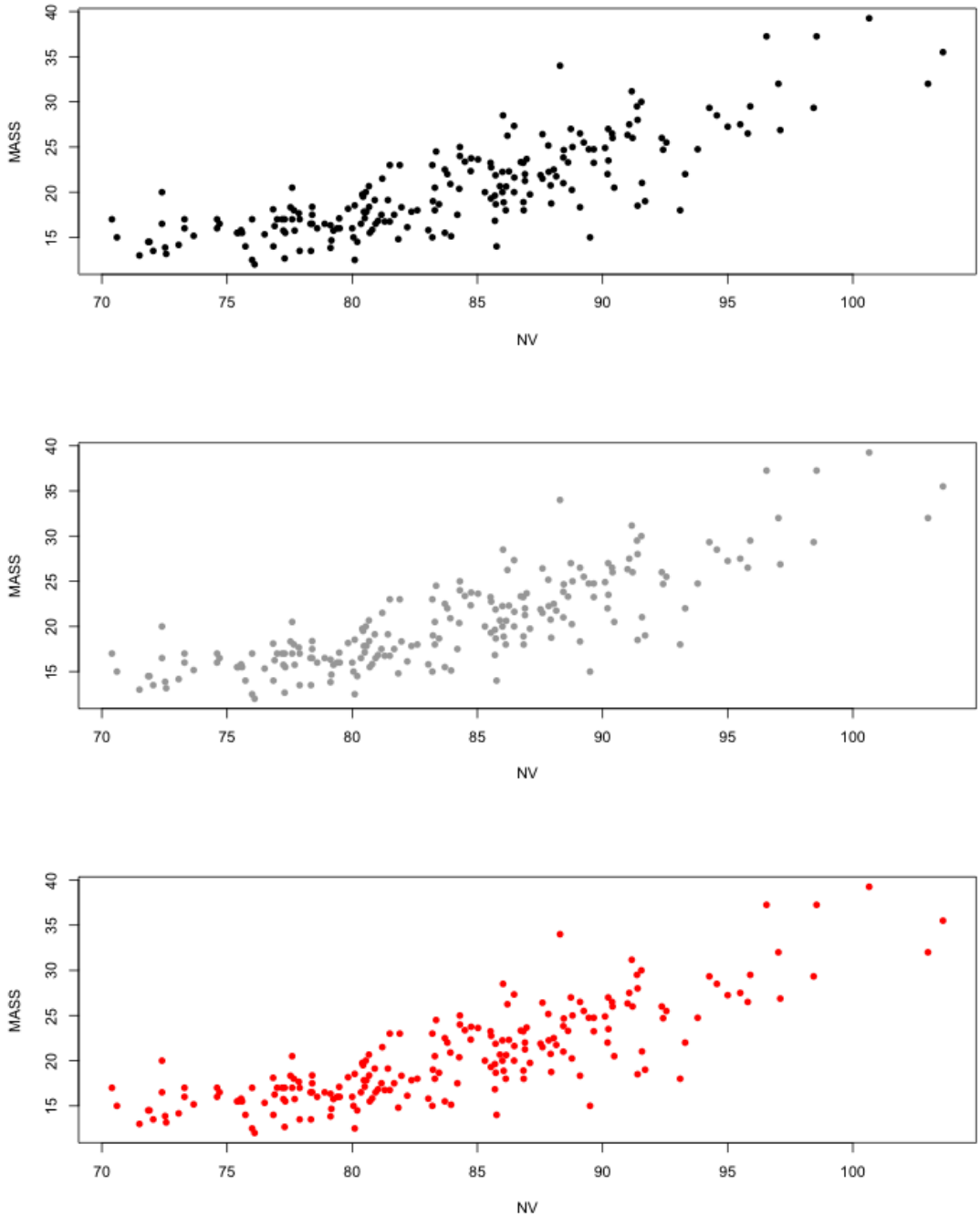
A plot space that holds 4 graphs

```
par(mfrow = c(2, 2))

plot(MASS~NV, data = agilis, pch = 16, col = "black")
plot(MASS~NV, data = agilis, pch = 16, col = "darkgrey")
plot(MASS~NV, data = agilis, pch = 16, col = "red")
plot(MASS~NV, data = agilis, pch = 16, col = "lightgrey")
```



A plot space that holds three graphs in a column

```
par(mfrow = c(3, 1))

plot(MASS~NV, data = agilis, pch = 16, col = "black")
plot(MASS~NV, data = agilis, pch = 16, col = "darkgrey")
plot(MASS~NV, data = agilis, pch = 16, col = "red")
```



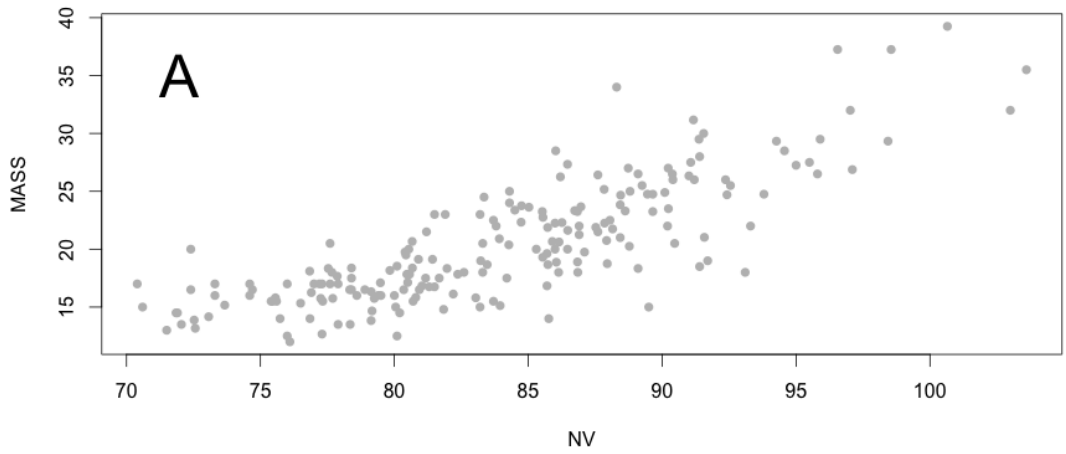
Add letters to a plot

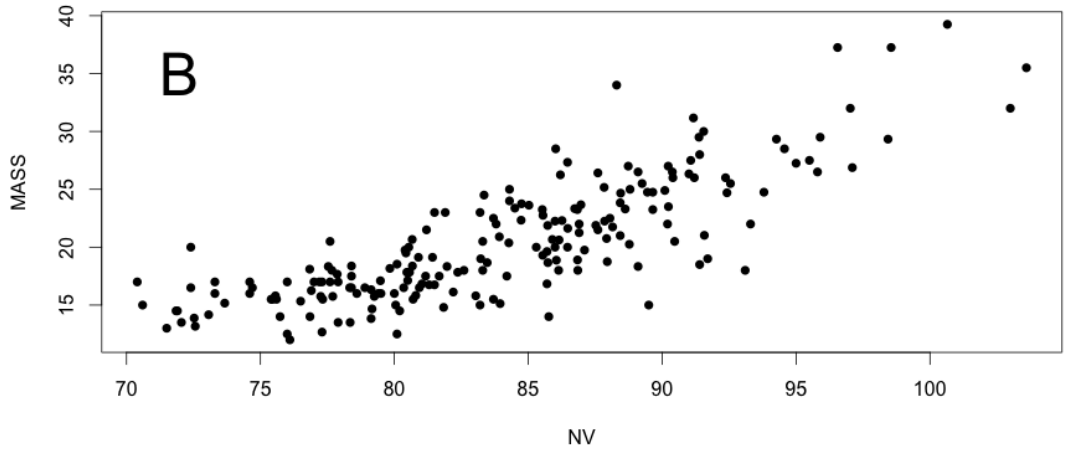
Text can be added inside a plot using this code.

```
text(X, Y, "what you want to write", cex = 1)
```

The X and Y co-ordinates are in the units of the axes.

```
par(mfrow = c(2, 1))

plot(MASS~NV, data = agilis, pch = 16, col = "grey")
text(72, 35, "A", cex = 3)

plot(MASS~NV, data = agilis, pch = 16, col = "black")
text(72, 35, "B", cex = 3)
```



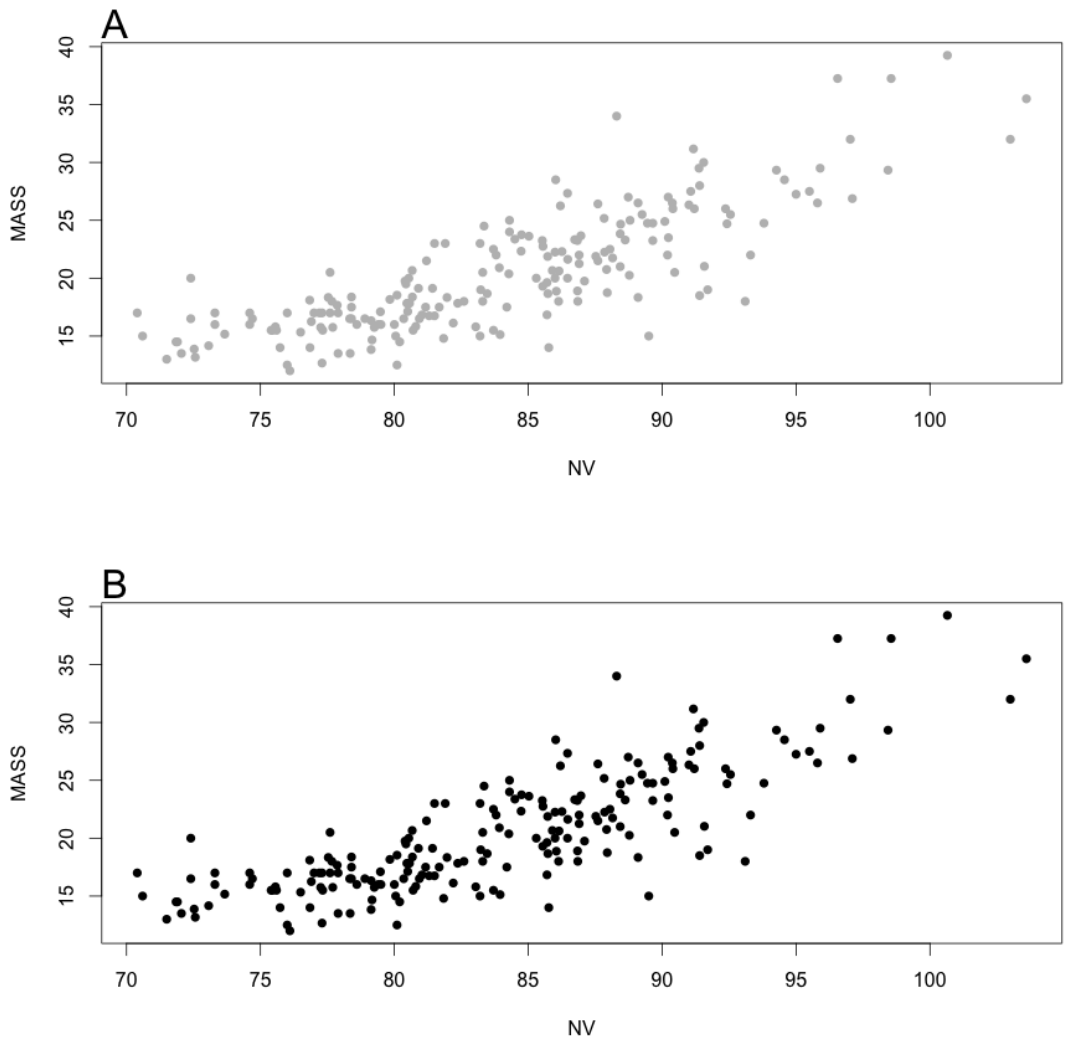


Text can be added above a plot using this code.

```
mtext("text", side = 3, adj = 0)
```

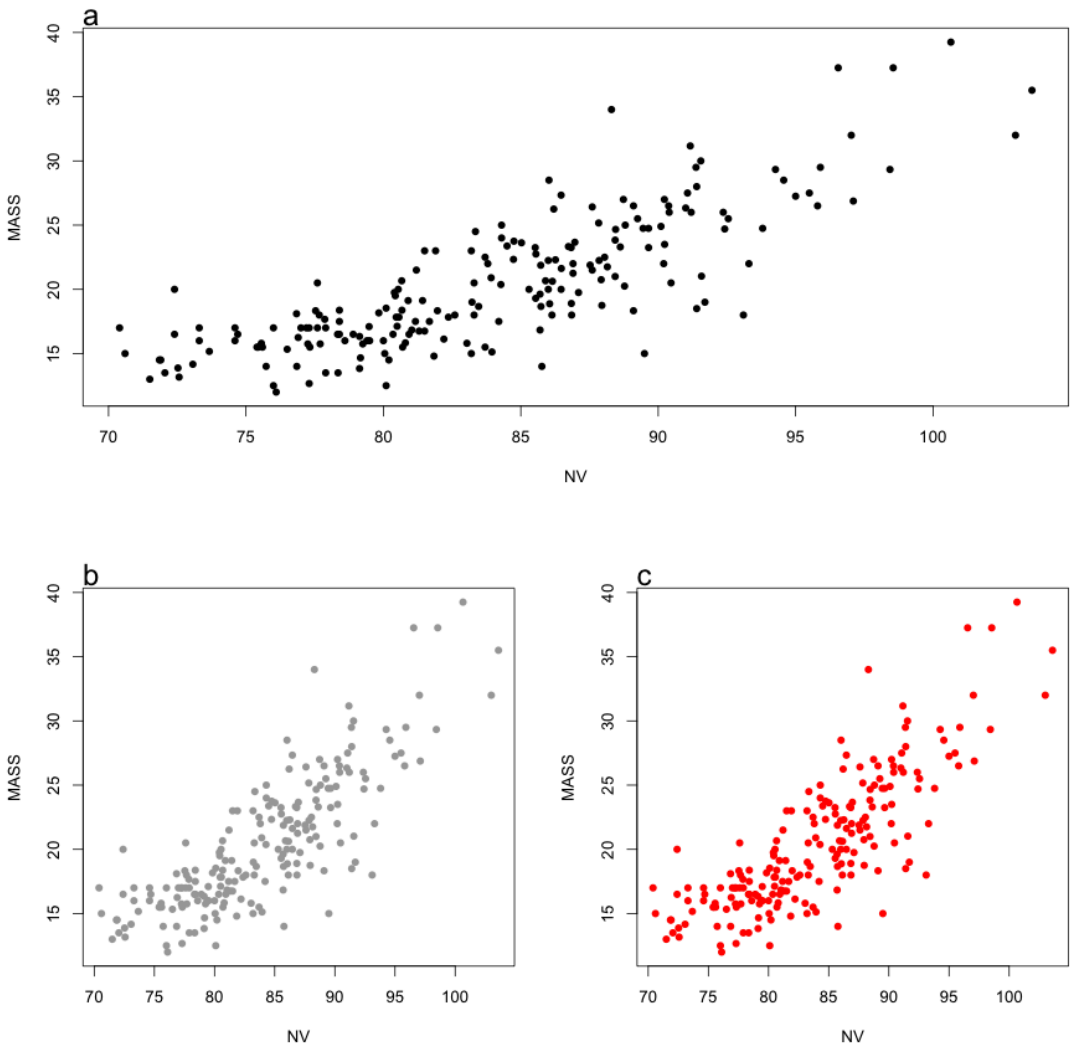
The X and Y co-ordinates are in the units of the axes.

```
par(mfrow = c(2, 1))

plot(MASS~NV, data = agilis, pch = 16, col = "grey")
mtext("A", side = 3, adj = 0, cex = 2, col = "black")

plot(MASS~NV, data = agilis, pch = 16, col = "black")
mtext("B", side = 3, adj = 0, cex = 2, col = "black")
```



The layout function can be used to generate a more unusual layout

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))

plot(MASS~NV, data = agilis, pch = 16, col = "black")
mtext("a", side = 3, adj = 0, cex = 1.5, col = "black")

plot(MASS~NV, data = agilis, pch = 16, col = "darkgrey")
mtext("b", side = 3, adj = 0, cex = 1.5, col = "black")

plot(MASS~NV, data = agilis, pch = 16, col = "red")
mtext("c", side = 3, adj = 0, cex = 1.5, col = "black")
```



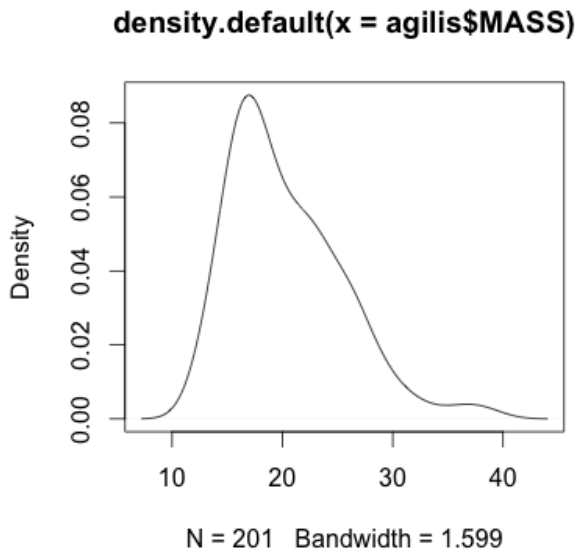
Density plots

Kernel density plots are an under-used form of plot. Kernal density estimation is a non-parametric way to estimate the probability density function of a random variable. The mathematics is beyond the scope of this course, but density plots can be very effective ways to view a continuous variable. Frist reset your par. We'll creat ea histogram first for comparison.

```
par(save.par)
```

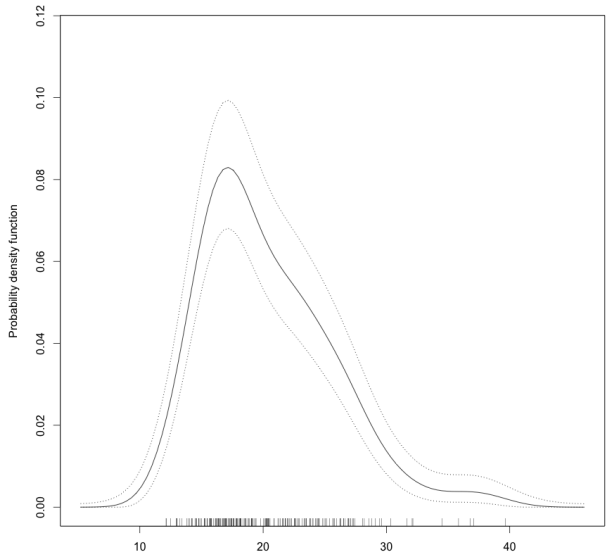
```
hist(agilis$MASS)
```


```
plot(density(agilis$MASS))
```

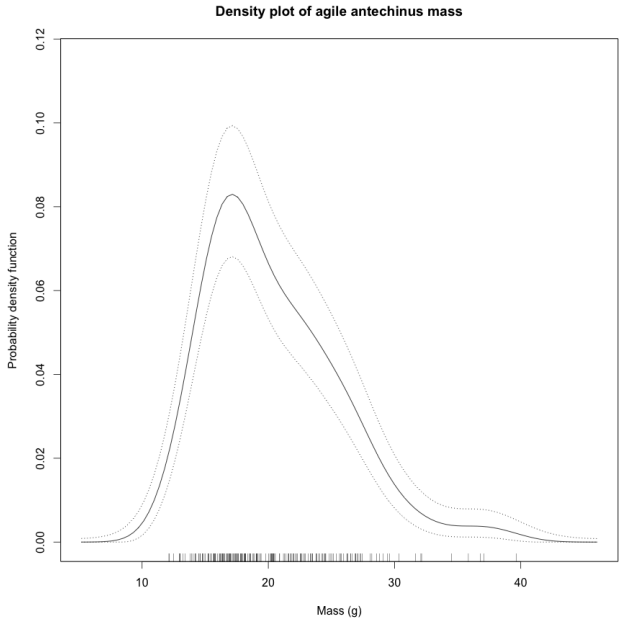


The library 'sm' has some nice additions to the density plot function.

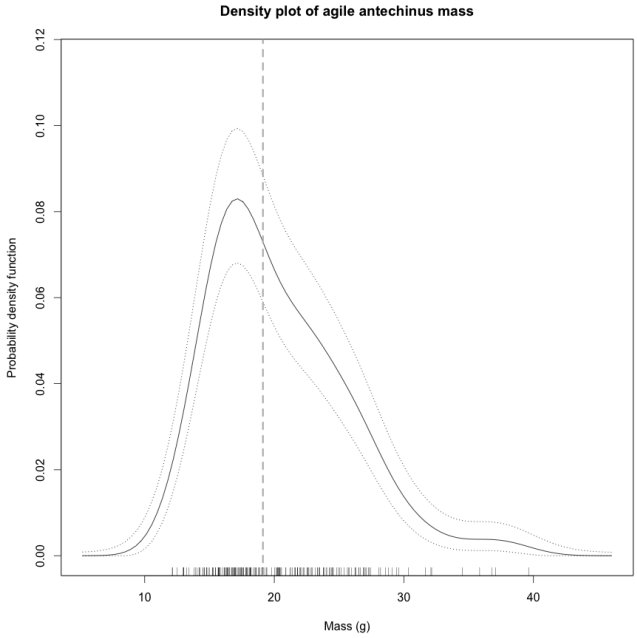
```
install.packages("sm")
library(sm)

sm.density(agilis$MASS, display="se", xlab="")
```




```
title("Density plot of agile antechinus mass", xlab="Mass (g)")
```



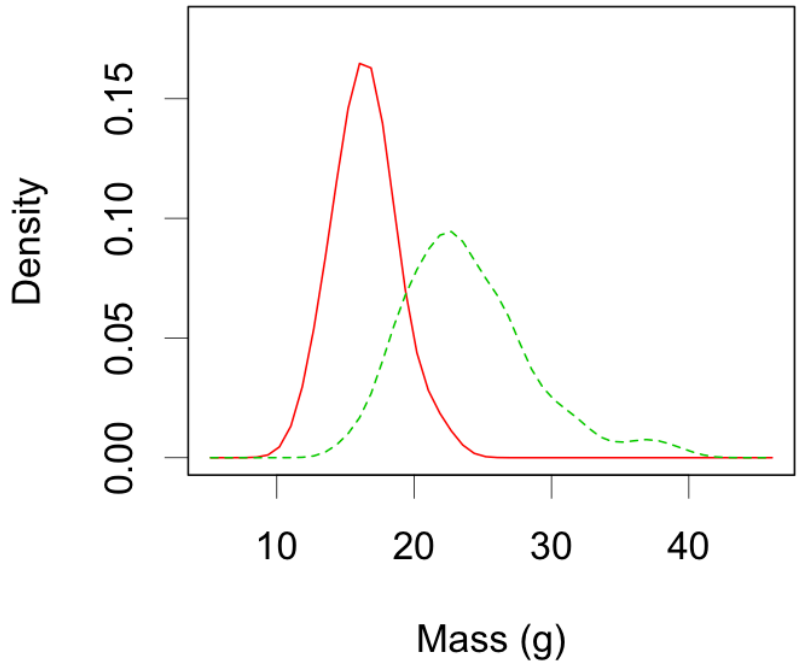

```
abline(v=median(agilis$MASS), lty=2, lwd = 3, col = "grey")
```



The above plot has a standard error and rugplot added to it. The rugplot is added by default.

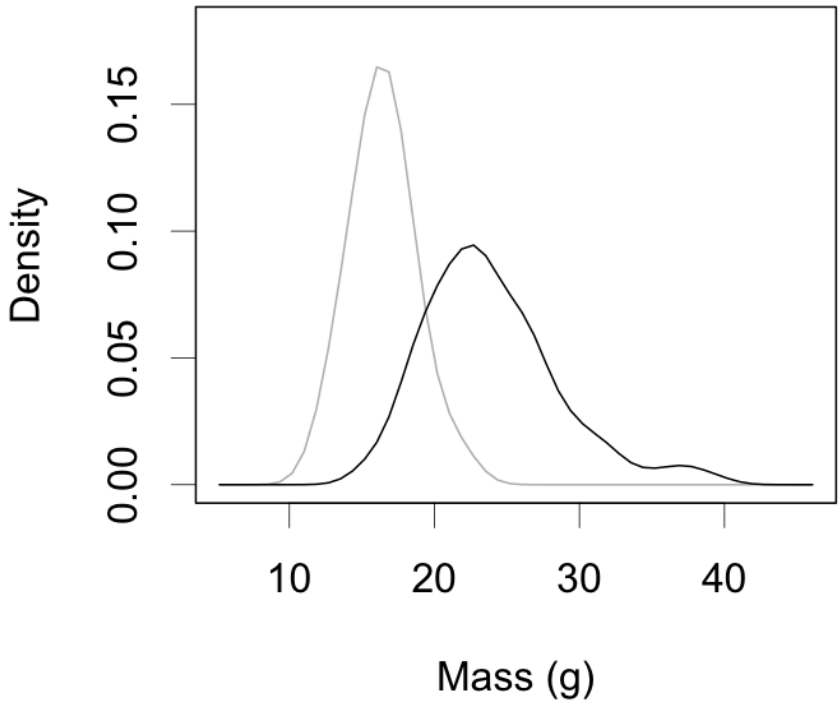
The **sm** library doesn't have a lot of plotting options built into it, so we need to change defaults before plotting if we want to alter aspects of the plot.

```
par(lwd=2)
par(cex=2)

sm.density.compare(agilis$MASS, agilis$SEX, xlab = "Mass (g)")
```



We might not like the colours or lines, so we can change those too.

```
sm.density.compare(agilis$MASS, agilis$SEX, xlab = "Mass
(g)",col=c("grey","black"),lty=c(1,1))
```



In the above example we would need to state in the figure caption that grey = females and black = males.

Using ggplot2

The **ggplot2** library works slightly differently to other plotting syntaxes in R. Instead of adding commands with a bracket, **ggplot2** adds commands using a plus (+) and including the new command with a bracket for any specific things you decide you want to set. Most of the following plots have been saved with a width of 400 pixels and a height of 400 pixels.

```
install.packages("ggplot2")
library(ggplot2)
```

The basic plots you are likely to need have a similar structure for data, but the defining difference is the geometric 'add-on'. ggplot2 works by attaching these 'add-on' components to a piece of basic code.

ggplot2 scatterplot

```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_point()
```

ggplot2 boxplot

```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_boxplot()
```

ggplot2 geometric dotplot

```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_dotplot()
```

ggplot2 violin plot

```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_violin()
```

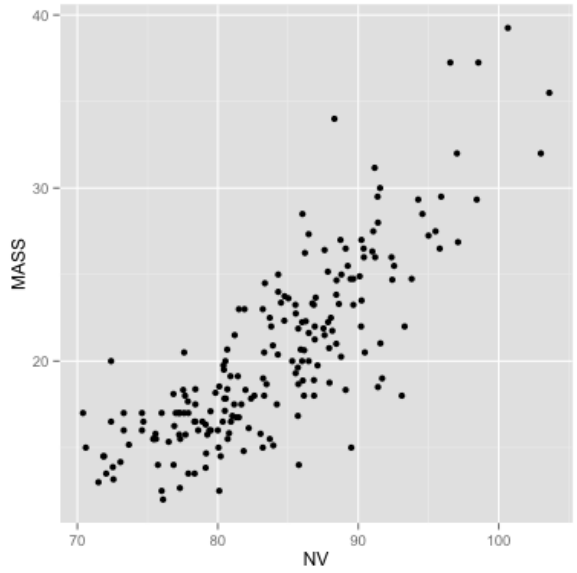
ggplot2 density plot

```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_density()
```

Basic scatterplot in ggplot2

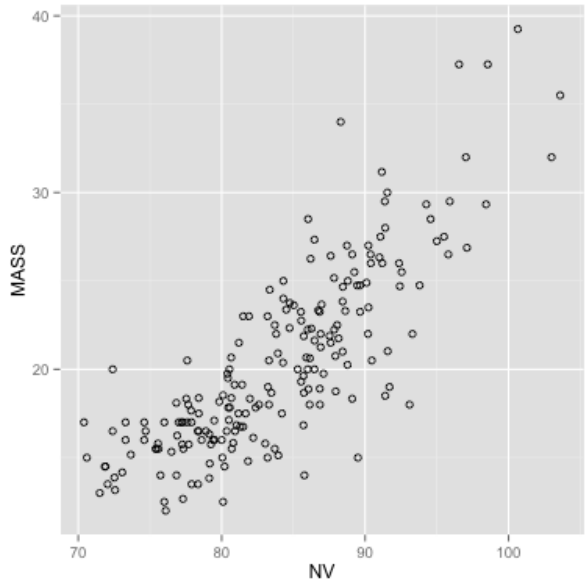
```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_point()
```

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point()
```



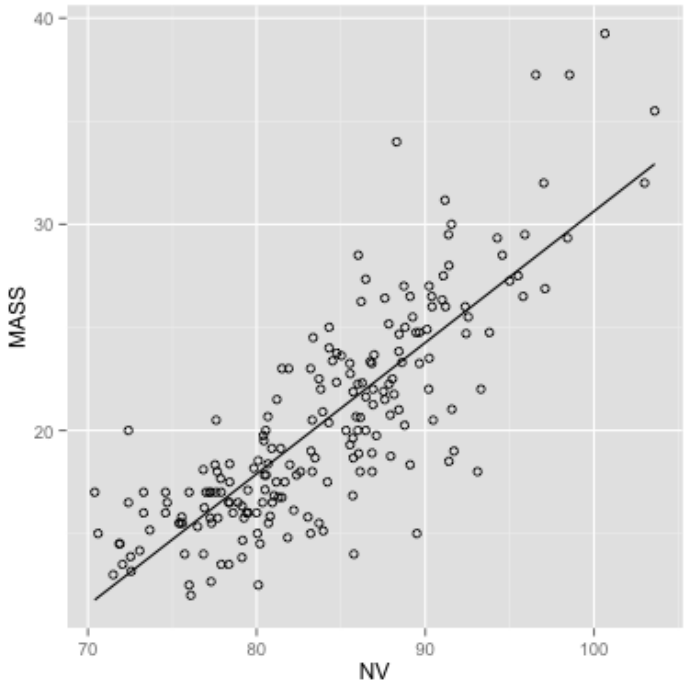
Use hollow circles

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1)
```



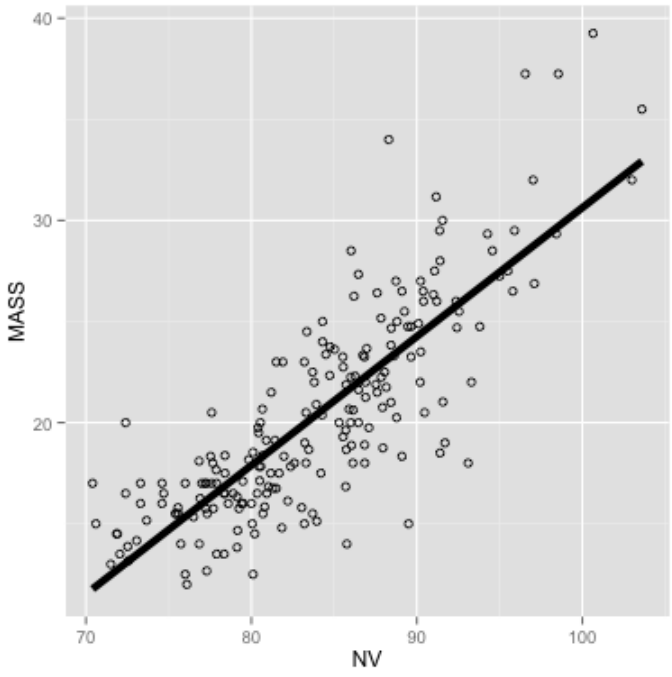
Add line of best fit

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1)
+ geom_smooth(method=lm, se=FALSE, col = "black")
```



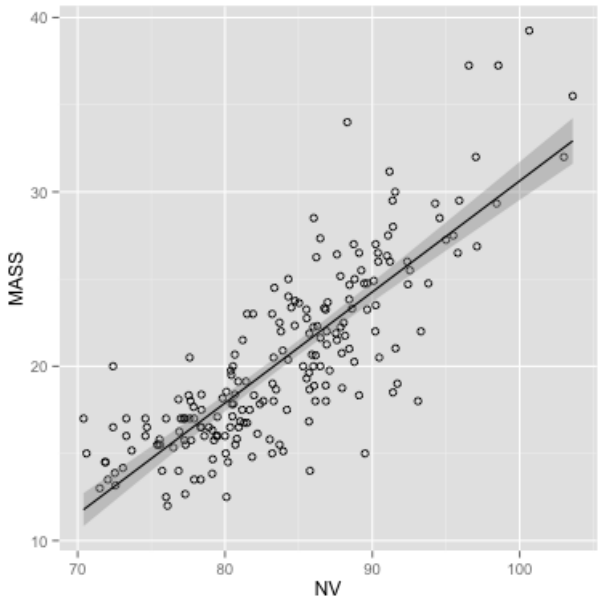
Add thick line of best fit

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1)
+ geom_smooth(method=lm, se=FALSE, col = "black", lwd = 2)
```



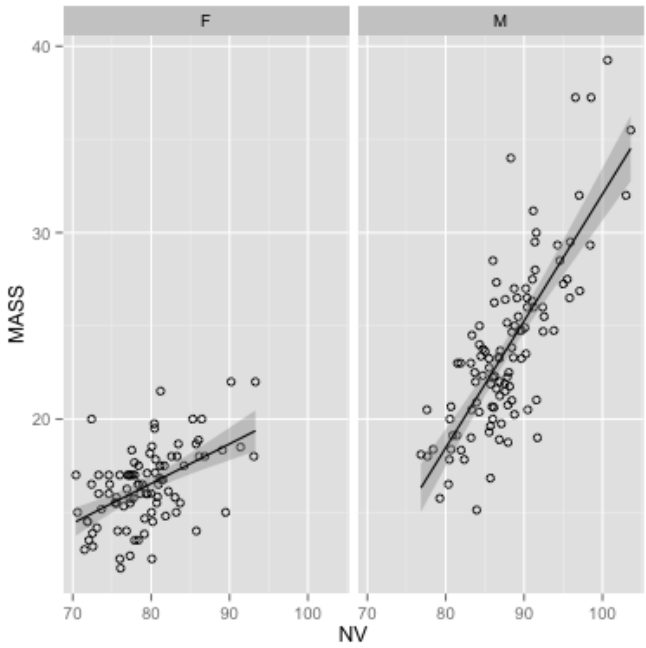
Add standard error (68.2%) confidence intervals

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1)
+ geom_smooth(method=lm, col = "black")
```



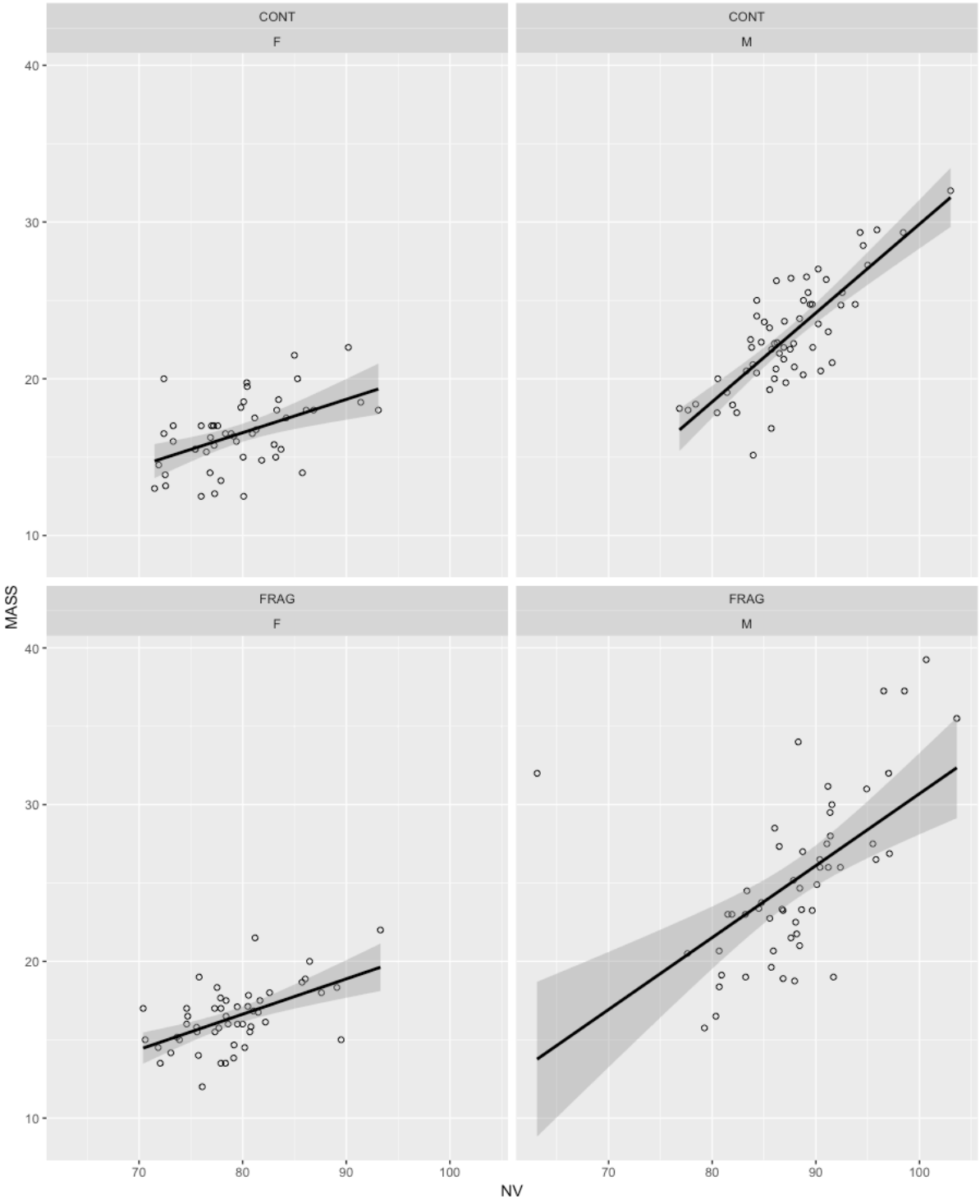
Wrap the scatterplot by SEX (MF)

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1) +
geom_smooth(method=lm, col = "black") + facet_wrap(~MF)
```



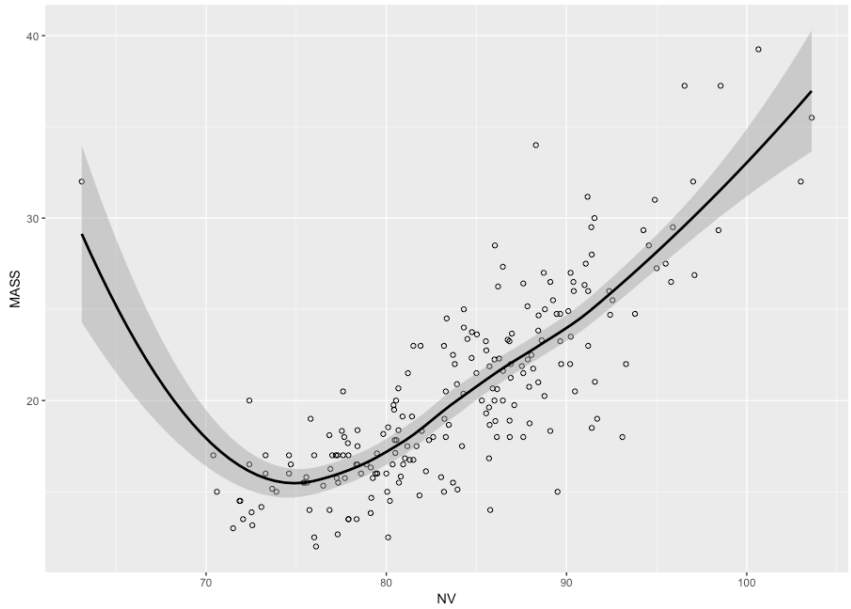
Wrap the scatterplot by SEX and HABITAT

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1) +
geom_smooth(method=lm, col = "black")
+ facet_wrap(HABITAT~MF)
```



Use a loess smoothed line instead of a straight line (appropriate for non-parametric correlations)

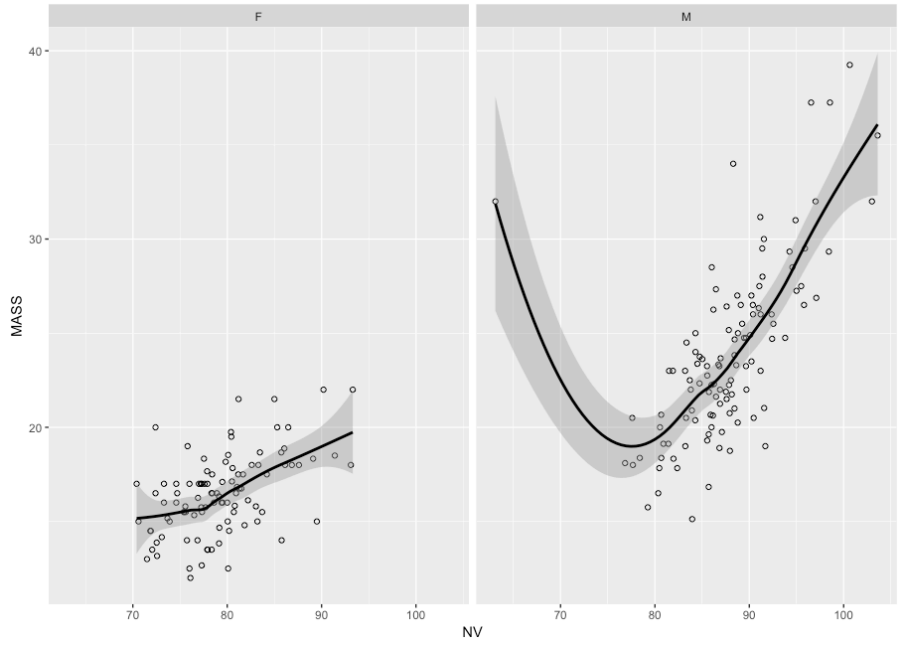
```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1)
+ geom_smooth(method=loess, col = "black")
```



Incidentally, this can be a good way to identify data-entry errors. The antechinus with a Nose-vent length of <60 mm is implausible for an adult. Something odd has happened there, and the data-point should probably be removed.

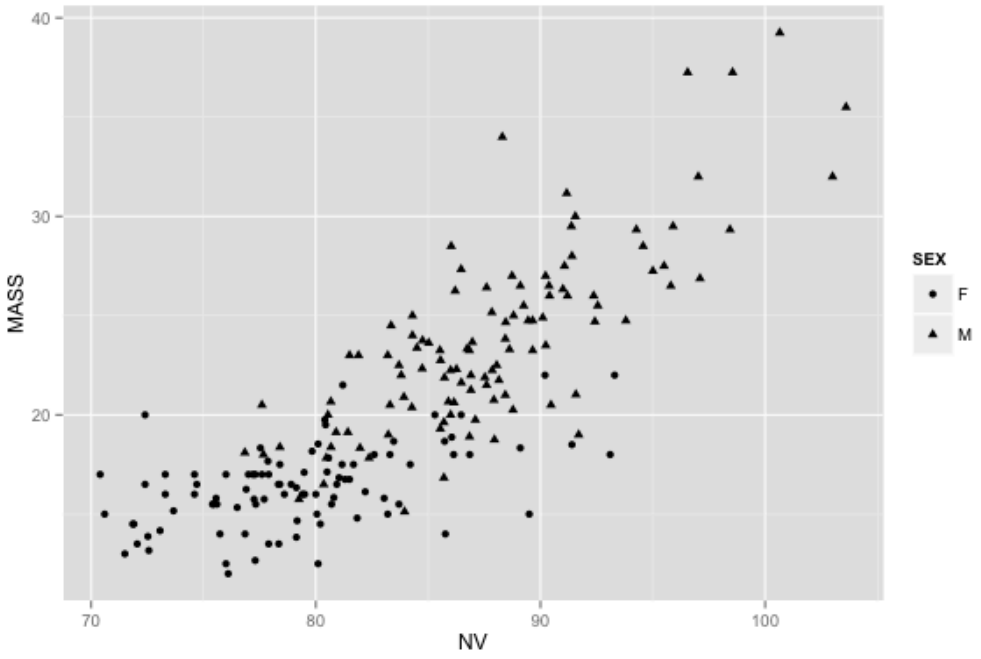
Wrap the scatterplot by SEX (MF) and use a loess smoother

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1) +
geom_smooth(method=loess, col = "black") + facet_wrap(~MF)
```



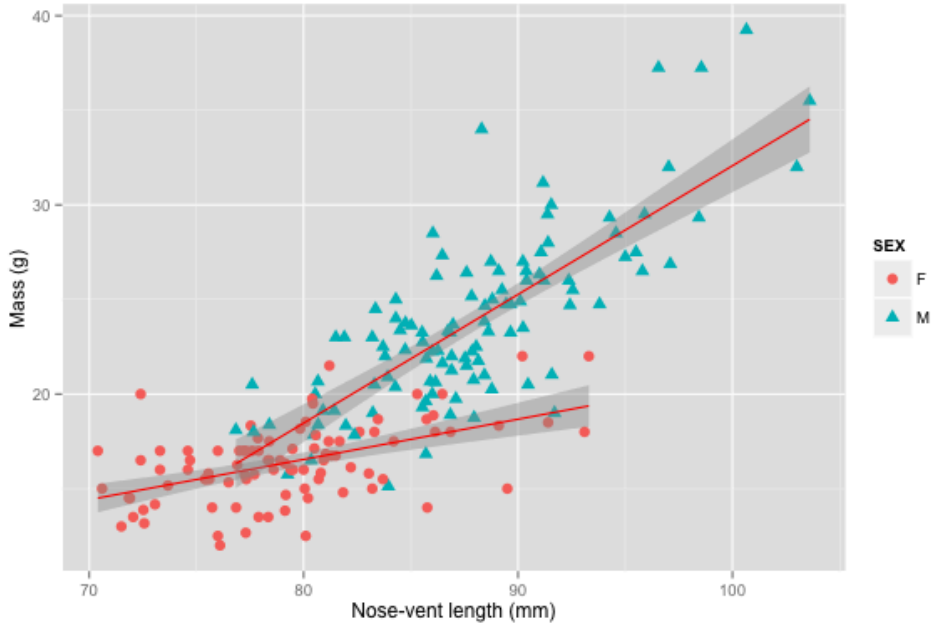
Split scatterplot point shapes by SEX on the same plot

```
ggplot(agilis, aes(y=MASS, x=NV, shape = MF)) + geom_point()
```



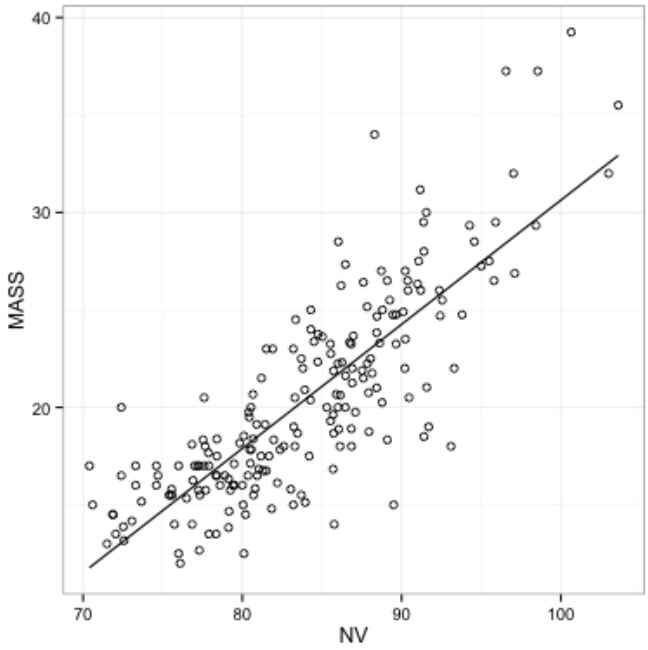
Split scatterplot by SEX on the same plot & change the x and y labels

```
ggplot(agilis, aes(y=MASS, x=NV, shape = MF, col = MF)) +
geom_point(size=3) + geom_smooth(method=lm, col = "red") +
xlab("Nose-vent length (mm)") + ylab("Mass (g)")
```



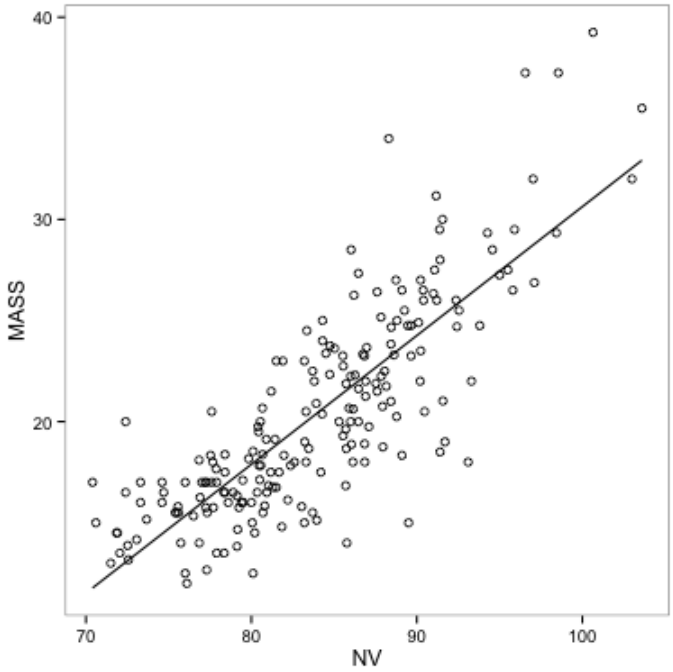
Remove background colour

```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1) +
geom_smooth(method=lm, se=FALSE, col = "black")+ theme_bw()
```



Remove background colour and grid

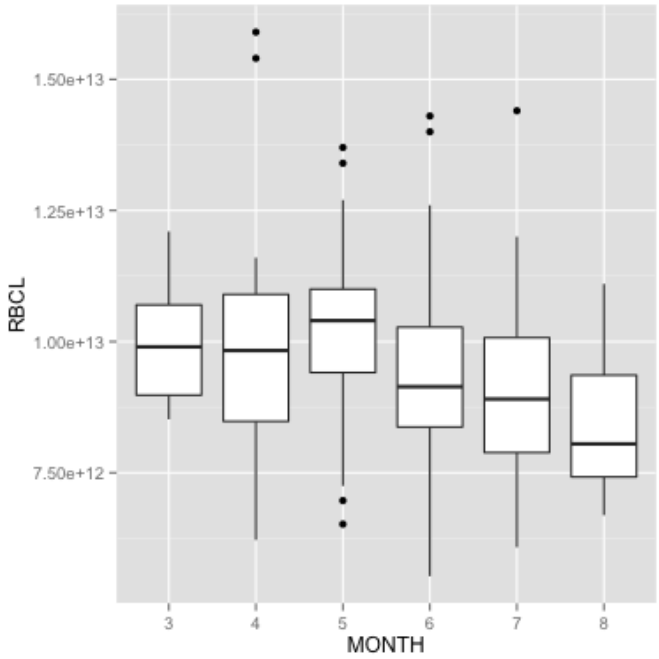
```
ggplot(agilis, aes(y=MASS, x=NV)) + geom_point(shape=1) +
geom_smooth(method=lm, se=FALSE, col = "black") + theme_bw() +
theme(panel.grid.major = element_blank(),
panel.grid.minor = element_blank())
```



Basic boxplot in ggplot2

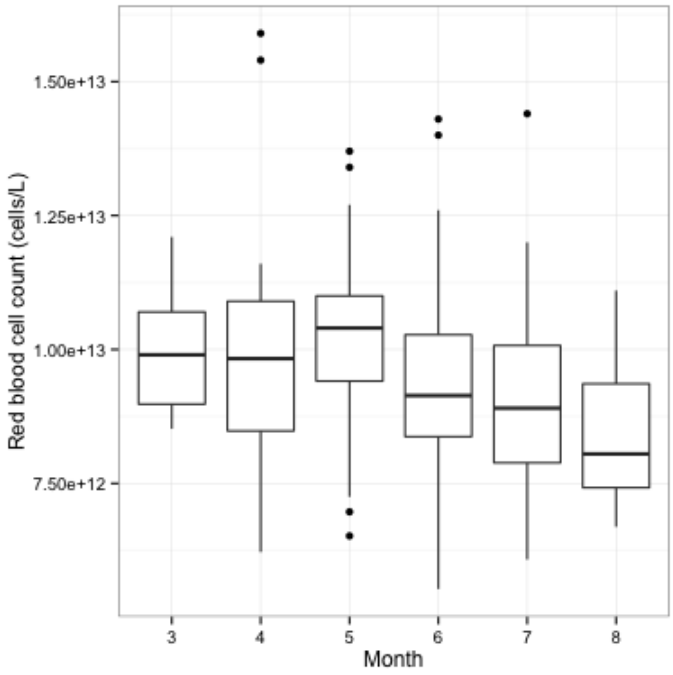
Change month to a factor

```
agilis$MONTH <- as.factor(agilis$MONTH)

ggplot(agilis, aes(y=RBC, x=MONTH)) + geom_boxplot()
```



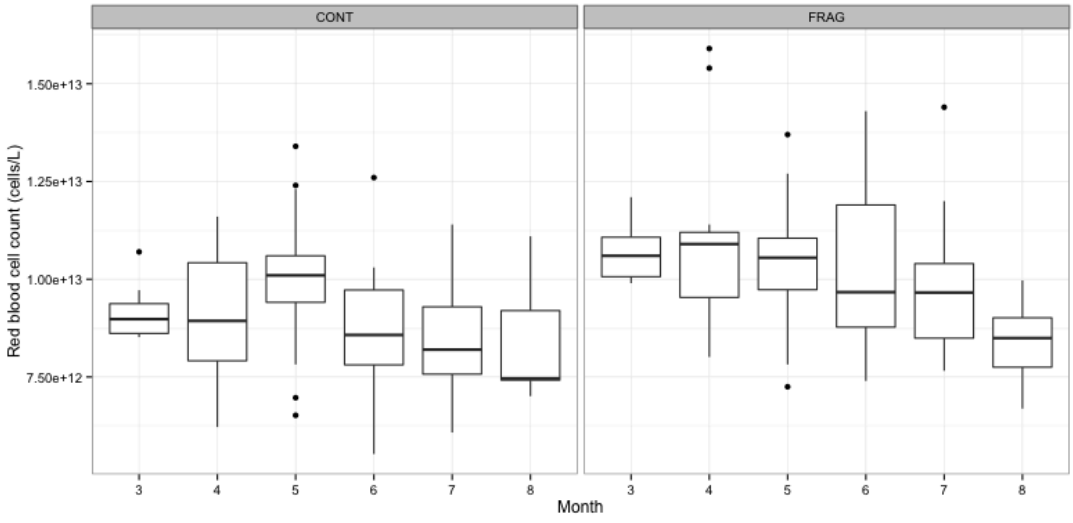
Add x and y labels & remove grey background

```
ggplot(agilis, aes(y=RBC,x=MONTH)) + geom_boxplot() +
xlab("Month") + ylab("Red blood cell count (cells/L)") +
theme_bw()
```



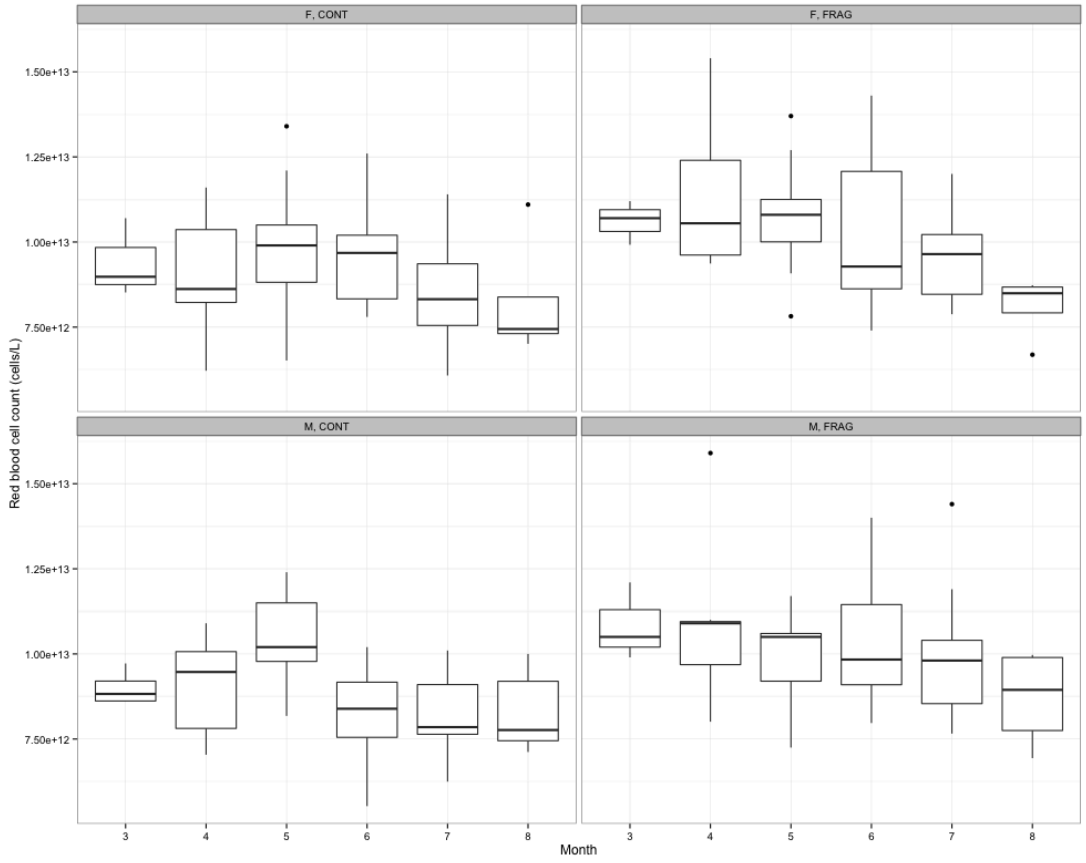
Wrap by habitat (continuous or fragmented)

```
ggplot(agilis, aes(y=RBC,x=MONTH)) + geom_boxplot() +
facet_wrap(~HABITAT) + xlab("Month") + ylab("Red blood cell
count (cells/L)") + theme_bw()
```



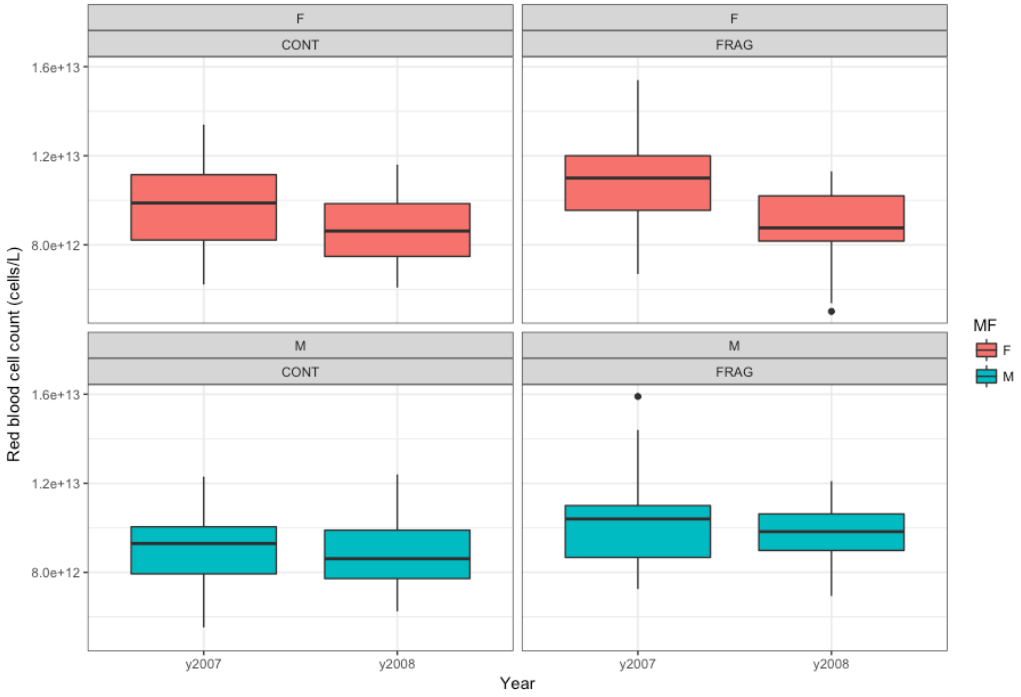
Wrap by habitat (continuous or fragmented) and sex

```
ggplot(agilis, aes(y=RBC,x=MONTH)) + geom_boxplot() +
facet_wrap(MF~HABITAT) + xlab("Month") + ylab("Red blood cell
count (cells/L)") + theme_bw()
```

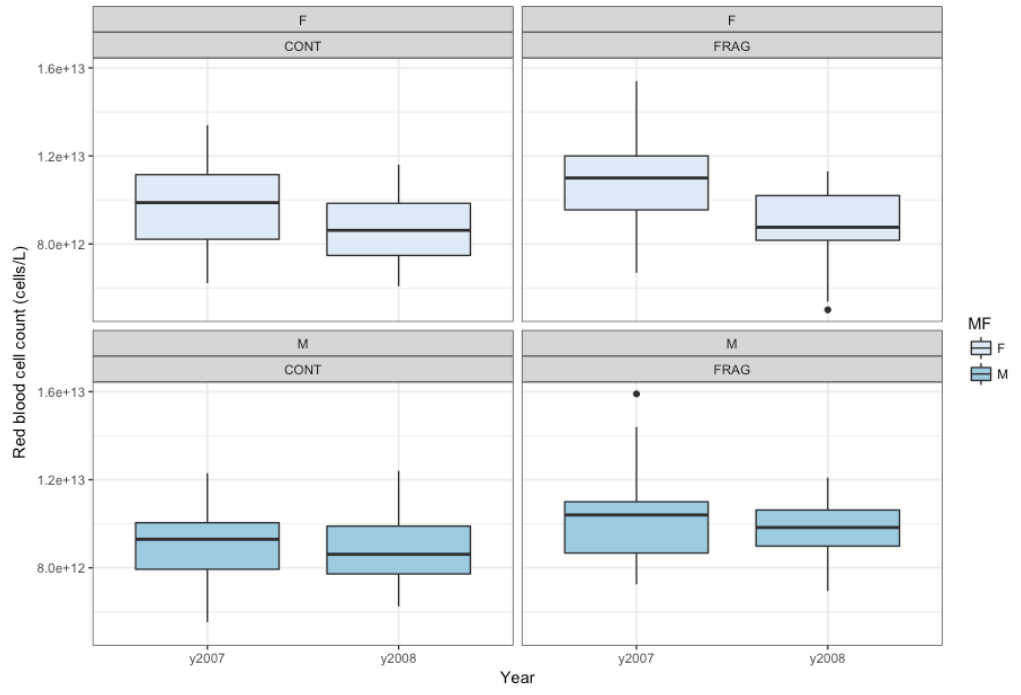


With colour

```
ggplot(agilis, aes(y=RBC,x=YEAR, fill=MF)) + geom_boxplot() +
facet_wrap(MF~HABITAT) + xlab("Year") + ylab("Red blood cell
count (cells/L)") + theme_bw()
```




```
ggplot(agilis, aes(y=RBC,x=YEAR, fill=MF)) + geom_boxplot() +
facet_wrap(MF~HABITAT) + xlab("Year") + ylab("Red blood cell
count (cells/L)") + theme_bw() + scale_fill_brewer(palette =
"Blues")
```



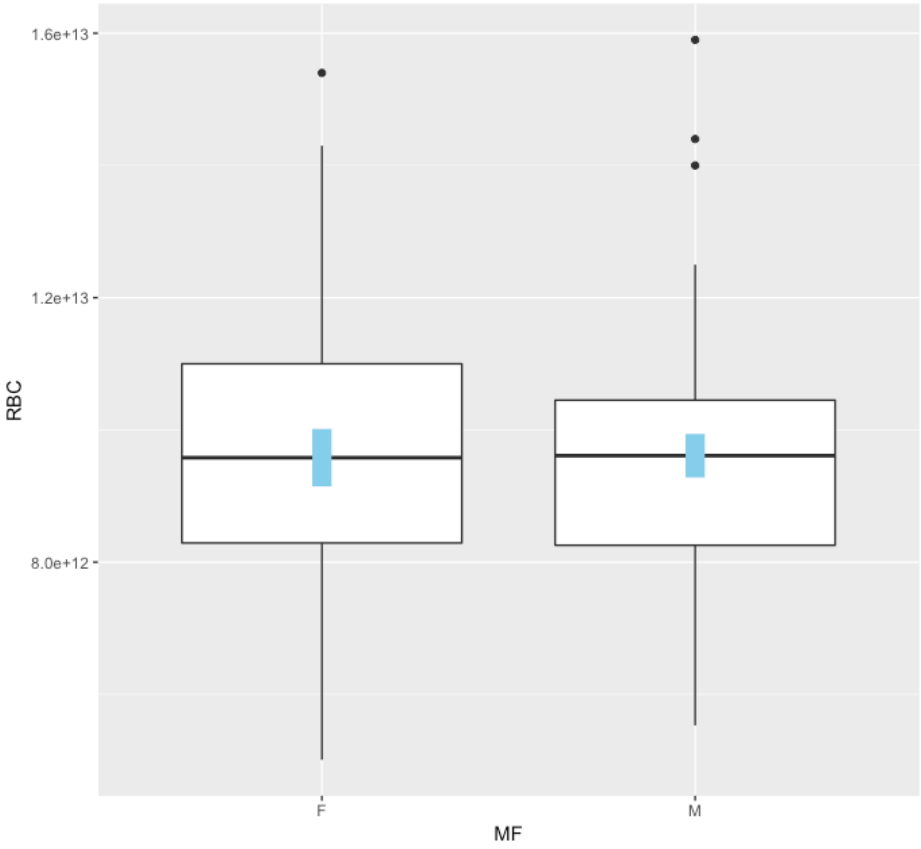
Creating a confidence interval calculated by boxplot.stats. This is effectively turning the 95% 'notches' from a boxplot to shaded bars instead.

Create a function:

```
f <- function(x) {
    ans <- boxplot.stats(x)
    data.frame(ymin = ans$conf[1], ymax = ans$conf[2])
}
```

Create the plot

```
ggplot(agilis, aes(y=RBC,x=MF)) + geom_boxplot() +
  stat_summary(fun.data = f, geom = "linerange", colour =
"skyblue", size = 5)
```



Same as above, except that the 95% CI for the meadian is a transparent blue bar the whole width of the box.

Create a function:

```
f <- function(x) {
    ans <- boxplot.stats(x)
    data.frame(ymin = ans$conf[1], ymax = ans$conf[2])
}
```

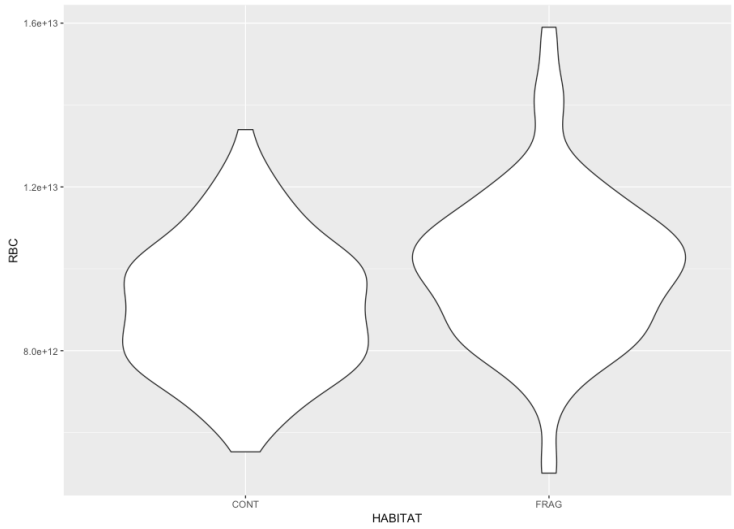
Create the plot

```
ggplot(agilis, aes(y=RBC,x=MF)) + geom_boxplot(width = 0.8) +
stat_summary(fun.data = f, geom = "crossbar", colour = NA, fill = "skyblue", width = 0.8,
alpha = 0.5)
```

Basic violin plot in ggplot2

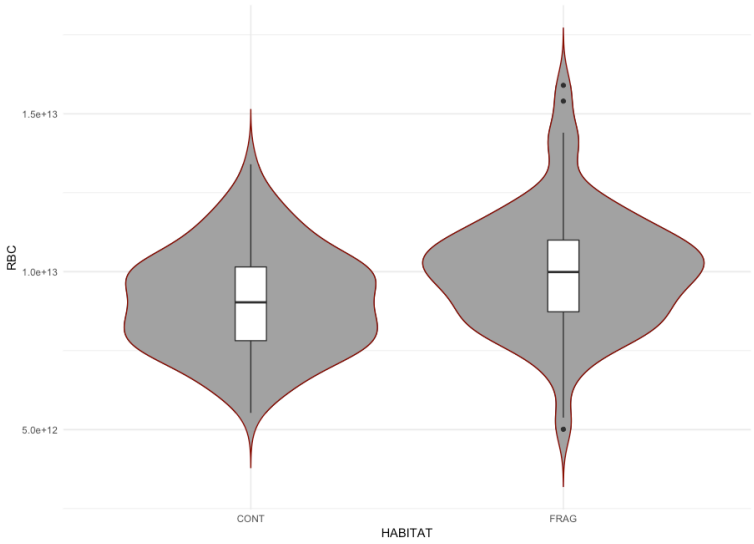
```
ggplot(dataset, aes(x=xvar, y=yvar)) + geom_violin()
```

```
ggplot(agilis, aes(y=RBC, x=HABITAT)) + geom_dotplot()
```



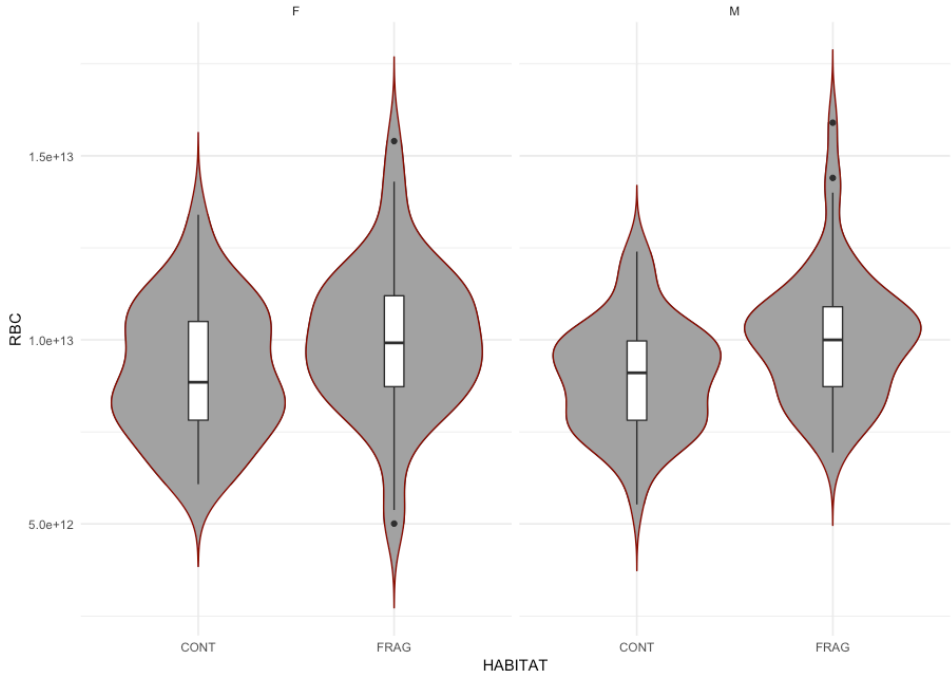
Remove trim, add colour and very simple boxplots

```
ggplot(agilis, aes(y=RBC, x=HABITAT)) +
geom_violin(trim=FALSE, fill='#A4A4A4', color="darkred",
width=0.9) + geom_boxplot(width=0.1) + theme_minimal()
```

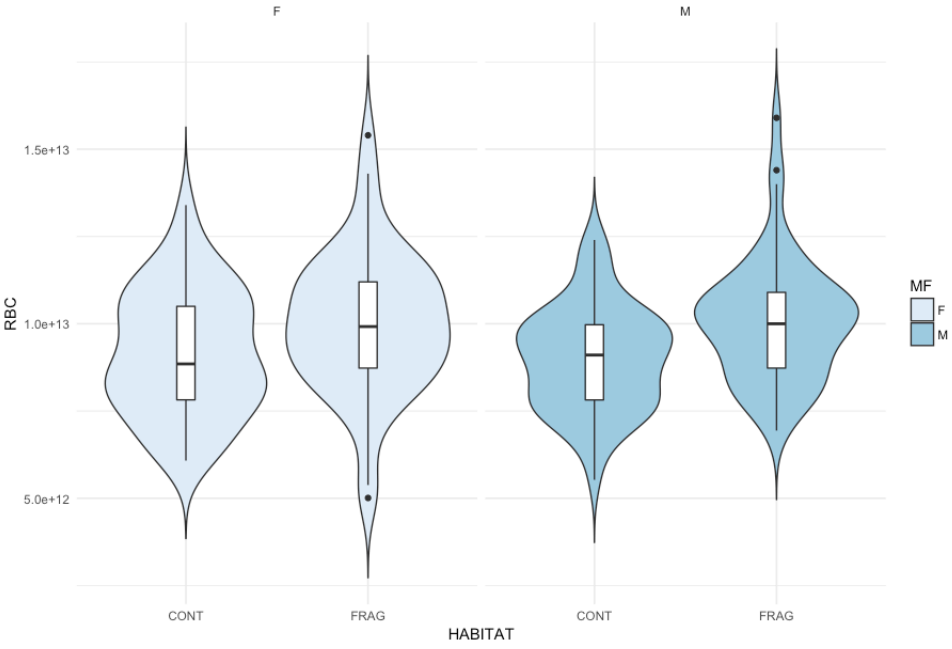


Same as above, but wrap by SEX

```
ggplot(agilis, aes(y=RBC, x=HABITAT)) +
geom_violin(trim=FALSE, fill='#A4A4A4', color="darkred",
width=0.9) + geom_boxplot(width=0.1) + theme_minimal()+
facet_wrap(~MF)
```



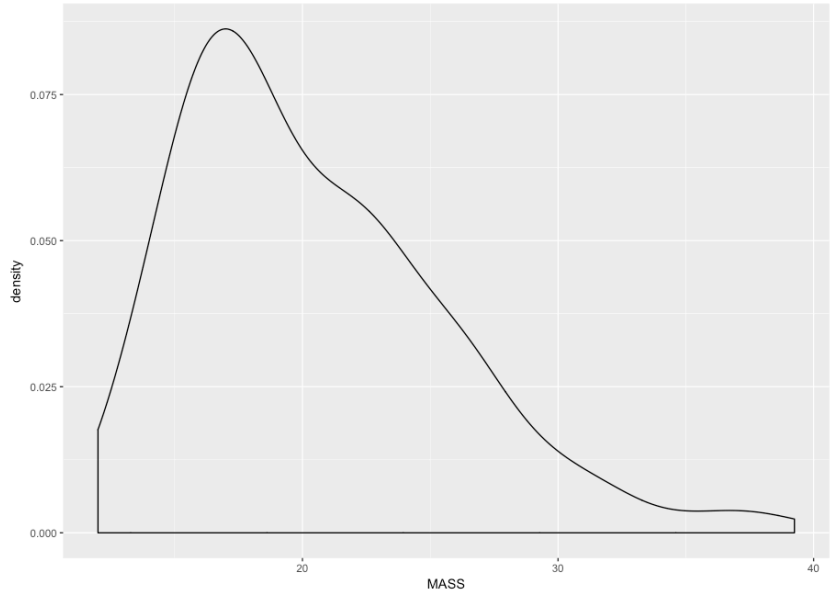

```
ggplot(agilis, aes(y=RBC, x=HABITAT, fill=MF)) +
geom_violin(trim=FALSE, width=0.9) + geom_boxplot(width=0.1,
fill="white") + theme_minimal() + facet_wrap(~MF) +
scale_fill_brewer(palette = "Blues")
```



Basic kernal density plots in ggplot2

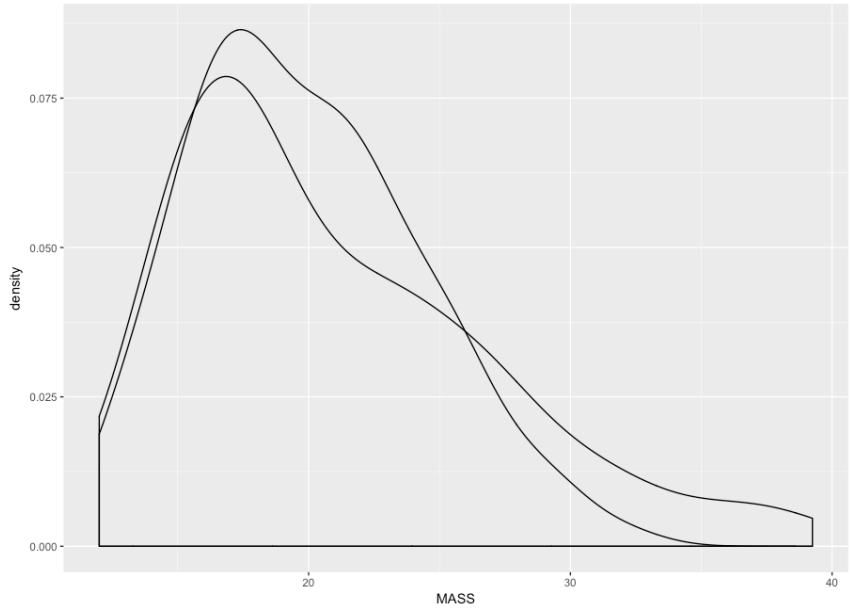
Grouped by habitat (fragment or continuous; indicated by colour). The **`alpha=0.5`** is used to set transparency.

```
ggplot(agilis, aes(x= MASS)) + geom_density()
```



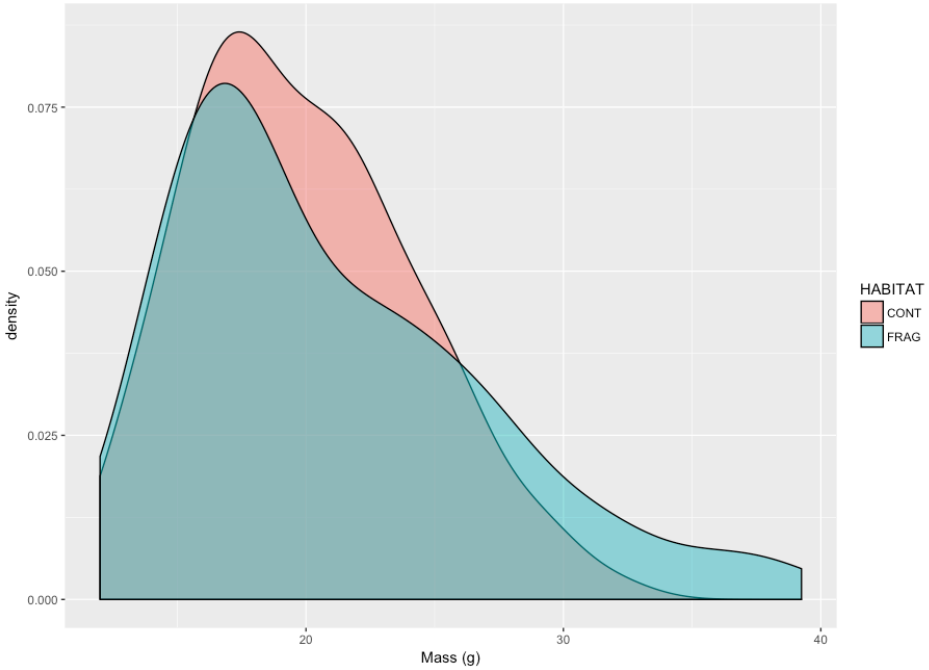
Same as above, but split by HABITAT

```
ggplot(agilis, aes(x= MASS, group=HABITAT)) + geom_density()
```



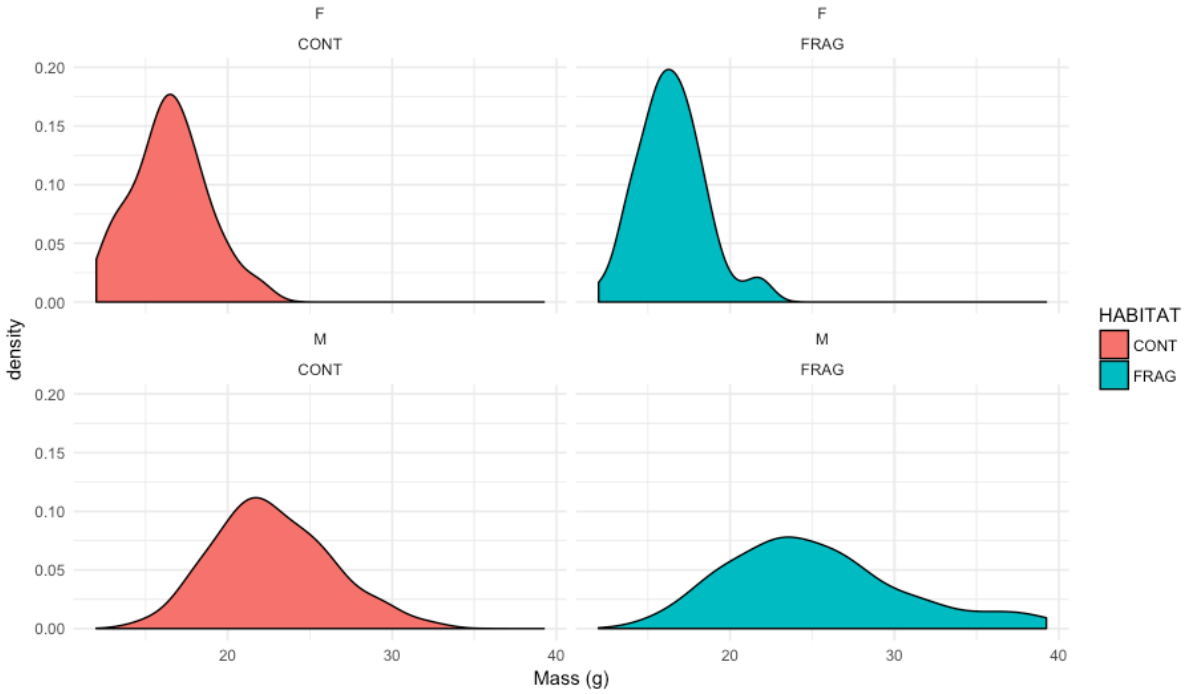
Same as above, but add colours with 50% transparency, and x label

```
ggplot(agilis, aes(x= MASS, group=HABITAT)) +
geom_density(aes(fill = HABITAT), alpha = 0.5) + xlab("Mass
(g)")
```



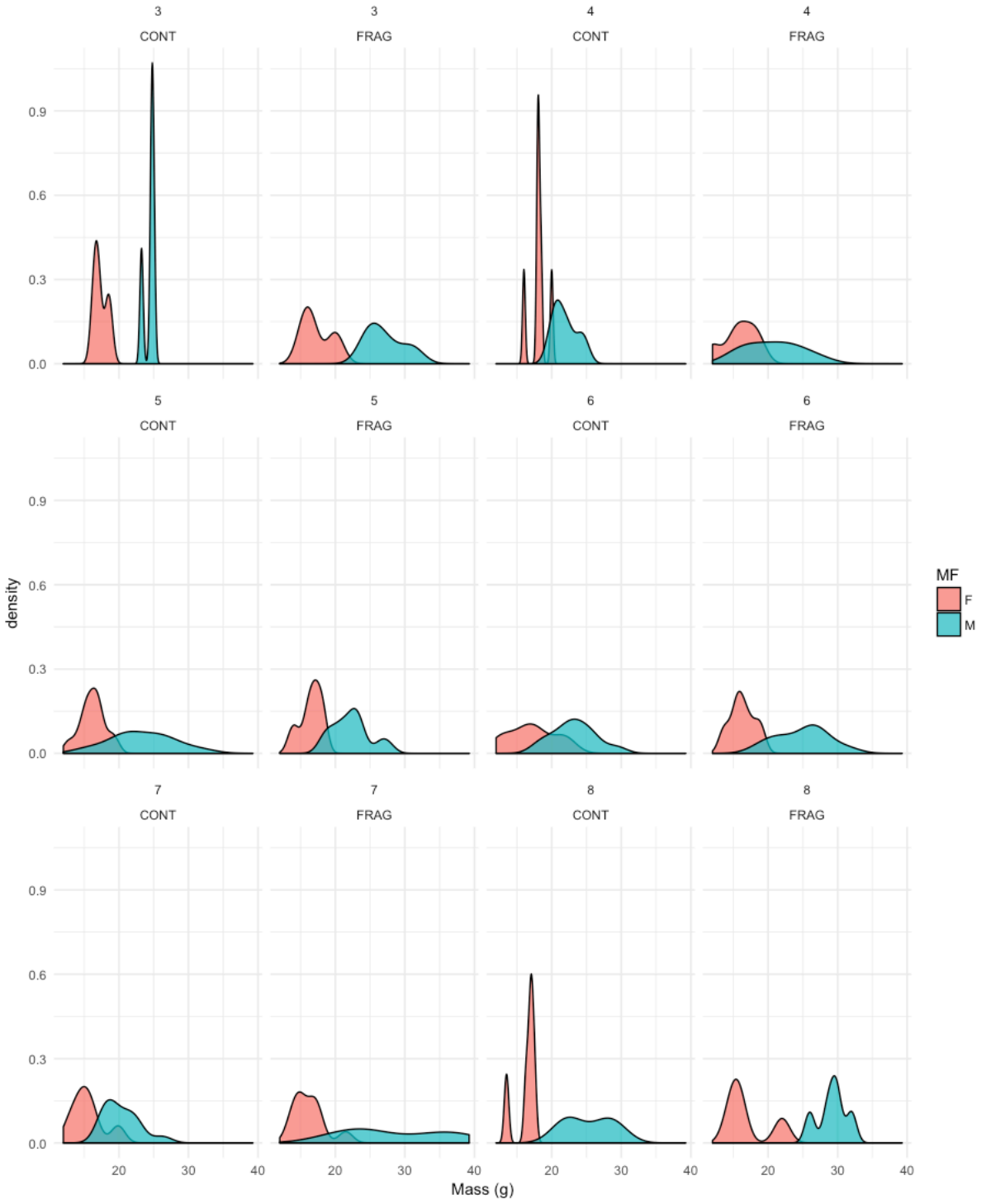
Same as above, but wrapped by sex and habitat, with minimal theme

```
ggplot(agilis, aes(x= MASS, group=HABITAT)) +
geom_density(aes(fill = HABITAT), alpha = 1.0) + xlab("Mass
(g)") + facet_wrap(MF~HABITAT) + theme_minimal()
```



Grouped by SEX, HABITAT and MONTH

```
ggplot(agilis, aes(x= MASS, group=MF)) + geom_density(aes(fill
= MF), alpha = 0.75) + xlab("Mass (g)") +
facet_wrap(MONTH~HABITAT) + theme_minimal()
```



Saving figures to files

Often a journal will ask you to save files using a high resolution size or you may need to save a very large figure so that you can scale it up to look nice on a poster. There are two types of image file that you can choose from.

Bitmap images are made up of pixels. A bitmap image will look pixellated or fuzzy if it is resized too large. A tiff, bmp or jpeg will store as a bitmapped file.

Vector images are made up of equations and information that draws lines. A vector image can be scaled up to any size. A PDF, EPS or PS file will store as a vector.

Sometimes we want to save directly to a PDF or an image file. First make sure to **set your working directory** to where you want the file to save to. In R Studio this will be **Session > Set Workinging Directory > Choose directory**

To save an image to a pdf

```
pdf(file="FILENAME.pdf", width = X, height = X, ...)
```

Height and width default to inches. Use **?pdf** and you'll find lots of ways to change this, as well as changing font sets etc.

Insert plotting code now: the code below is an example one

```
plot(MASS~NV, data = agilis)
```

Run to reset everything when done.

```
dev.off()
```

You won't see anything happen, but a file called "FILENAME.pdf" will appear in the working directory and will contain your graph. You can change the settings you use to make it look 'right'. Try starting with height=7, width=7, and then change things like **`cex`**, **`cex.axis`**, **`cex.lab`** in the plotting code. Of course, this is journal specific, many journals like their plots exactly one or two columns wide (in which case, check what the journal's column measures are in inches).

To save an image to a TIFF, JPEG or BMP

Same for TIFFs, JPEGs, BMPs, etc (see **`?tiff`** or **`?jpeg`** for links to help file):

Set your working directory: **Session > Set Working Directory > Choose directory**

```
tiff(file="FILENAME.tiff", width = X, height = X, units = px,
...)
```

Width and height default to pixels now, but this can be changed with the units parameter and setting units = **`in`** (inches), **`cm`** or **`mm`**

If you look at ?tiff, you can also find settings like compression, font type, etc.

Insert plotting code here
```
plot(MASS~NV, data = agilis)
```

Run to reset everything to deault.
```
dev.off()
```

As before, you won't see anything happen, but your file will appear in the working directory.

EXAMPLE PDF
```
pdf(file="TEST1.pdf", width = 7, height = 7)
plot(MASS~NV, data = agilis)
dev.off()
```

EXAMPLE TIFF
```
tiff(file="TEST1.tiff", width = 400, height = 400)
plot(MASS~NV, data = agilis)
dev.off()
```

EXAMPLE TIFF WITH COMPRESSIONS
```
tiff(file="TEST2.tiff", width = 400, height = 400, compression
= "lzw")
plot(MASS~NV, data = agilis)
dev.off()
```

Diversity Indices

Library 'adiv' has a number of useful diversity indices that can be applied to species counts.

```
install.packages("adiv")
library(adiv)

# import the macnally.csv dataset
birds <- read.table('macnally.csv', header=T, sep=',')
```

Mac Nally recorded bird abundances (V1GST to V102KING) at sites divided into six forest types (Mixed, Gippsland Mallee, Montane Forest, Foothills Woodland, Box-Ironbark, River Red Gum). We'll use the bird counts to generate diversity indices. There are a couple steps to this:

1) Generate a matrix of the counts
2) Apply the diversity function(s) to the matrix

We generate a matrix by nominating the columns we want to include. The easiest way to do this is pull out the columns by number. We want to avoid include the first two columns, because these are categorical data and the diversity function will refuse to work if you try to give it non-numerical data.

```
# generate a matrix of bird counts from column 3 to column 13
# just as an example
birds.matrix <- as.matrix(birds[,3:13])
birds.matrix

# generate a matrix of all bird counts
# from column 3 to 104 (note that the species names are coded)
birds.matrix <- as.matrix(birds[,3:13])
birds.matrix # look at matrix
```

We can generate diversity indices of all possible options, but this is not very useful, as we can't easily test these against variables in the original dataset.

```
diversities <- speciesdiv(birds.matrix) # all diversity indices
diversities # look at indices
```

Instead we'll just pull out a few of the more widely used indices.

```
birds$RICHNESS <- speciesdiv(birds.matrix, method = "richness")
birds$GINISIMPSON <- speciesdiv(birds.matrix, method = "GiniSimpson")
birds$SIMPSON <- speciesdiv(birds.matrix, method = "Simpson")
birds$SHANNON <- speciesdiv(birds.matrix, method = "Shannon")
```

And we'll extract a commonly used evenness index too, the Hills Numbers.

```
birds$EVENNESS <- eveparam(birds.matrix, method = "hill")
```

We wouldn't typically use all of these indices as responses, because several of them will tend to correlate anyway. If in doubt, GiniSimpson (which accounts for both diversity and abundance) and Hill's evenness is a reasonable suite to look at. We'll check these as responses against Habitat.

```
richness.aov <- aov(RICHNESS~HABITAT, data = birds)
ginisimpson.aov <- aov(GINISIMPSON~HABITAT, data = birds)
evenness.aov <- aov(EVENNESS~HABITAT, data = birds)
```

RESULT

```
> summary(richness.aov)
            Df Sum Sq Mean Sq F value   Pr(>F)
HABITAT      5 126.98  25.397   11.46 2.54e-06 ***
Residuals   31  68.69   2.216
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


> summary(ginisimpson.aov)
            Df  Sum Sq  Mean Sq F value Pr(>F)
HABITAT      5 0.07178 0.014356   2.454 0.0552 .
Residuals   31 0.18138 0.005851
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


> summary(evenness.aov)
            Df Sum Sq Mean Sq F value  Pr(>F)
HABITAT      5 0.1893 0.03786   4.553 0.00315 **
Residuals   31 0.2578 0.00831
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because we have three or more levels in a factor (i.e. Habitat is made up of three or more levels), we'd need to use a post hoc test such as a Tukey's to tease apart what is going on here, however, that is a relatively simple next step. We also might need to consider using a linear mixed effects model instead of an ANOVA to control for pseuoreplication if we have multiple samples from a single site, but that is covered under the linear mixed effects section.

Advanced Statistics

The following material covers a number of slightly more sophisticated statistical tests that you may require.

Principal Components Analysis

Multivariate datasets can have a large number of either predictor or response variables (or both). Traditionally, one way to analyse large multivariate datasets is to use principal components analysis (PCA). PCA is not as widely used as it once was, but it is still a useful function to be aware of.

Useful things about a PCA

Allows for variable reduction: if you have too many predictor variables (i.e. your ANOVA models will be overparametised) you can use PCA to create axes that represent gradients in the dataset

Allows multiple response variables to be analysed as a whole: this function is similar to the application of a test like a MANOVA, except that more information can be extracted from a PCA than from a MANOVA sometimes

Allows correlating predictor variables to be analysed together: including two correlating variables in a PCA will generate an axis that accounts for both of them.

The two native functions in R are **`prcomp`** and **`princomp`**, which differ in the mathematics used to calculate them. Generally, **`prcomp`** is preferred over **`princomp`**. However, here we're going to start with **`princomp`** and then provide **`prcomp`** for comparison.

princomp

Load libraries

```
library(vegan)
library(MASS)
```

Load data

```
agilis <- read.table('agilis-abundance.csv',header=T,sep=',')

str(agilis)
```

Generate a PCA using total area at basal height, median area at basal height, stumps, shrub count, shrub species and percentage of trees that were not *Eucalyptus* (measured in 20x20 m quadrats at study sites)

```
veg.pca <- princomp(~ ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT
+ SHRUB.SPECIES + pcNONEUC, cor=T, data=agilis)
```

Plot the first two PCA axes

```
biplot(veg.pca) # just picks the first two axes
```

The first two axes of the PCA have been plotted. The black numbers are site numbers, and their relative position tells you something about how similar sites are. The red arrows can be thought of as strength of contributions by the variables, and they are also indicators of correlation. **median.ABH** is pointing in the opposite direction to **pcNONEUC**. This means that these two variables are negatively correlated. **ABHm2** and **SHRUB.COUNT** are pointing in the same direction. They are positively correlated.

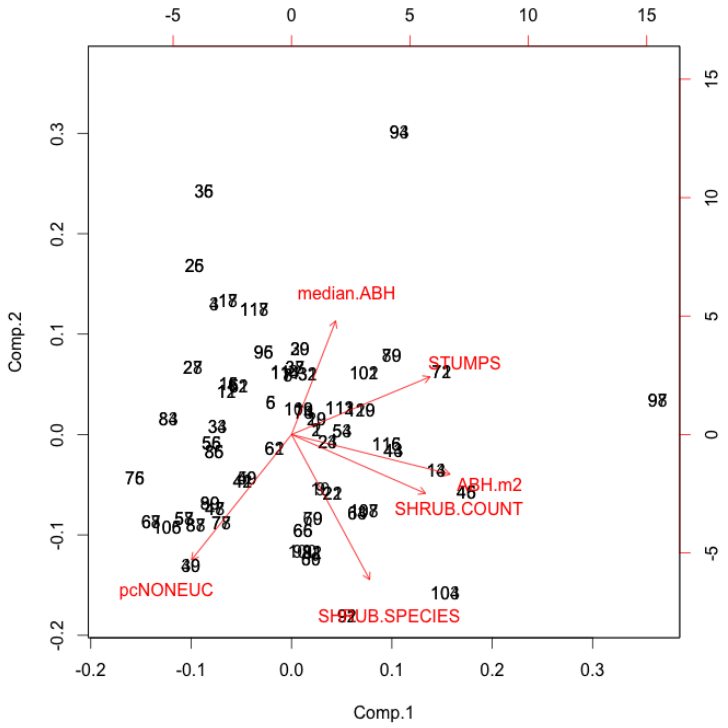


Figure 1. Biplot of a principal components analysis based on vegetation data collected at 120 sites in South Gippsland.

Plot an ordiplot with the type set to "none" (should be blank)

```
ordihull(ordiplot(veg.pca, type="none"), agilis$HABITAT)
```

Add labels to the ordiplot

```
text(veg.pca$scores, labels=agilis$HABITAT,
col=as.numeric(agilis$HABITAT))
```

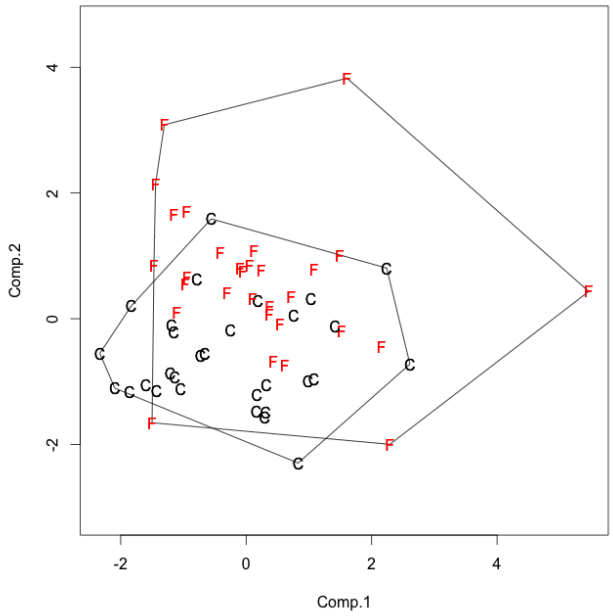


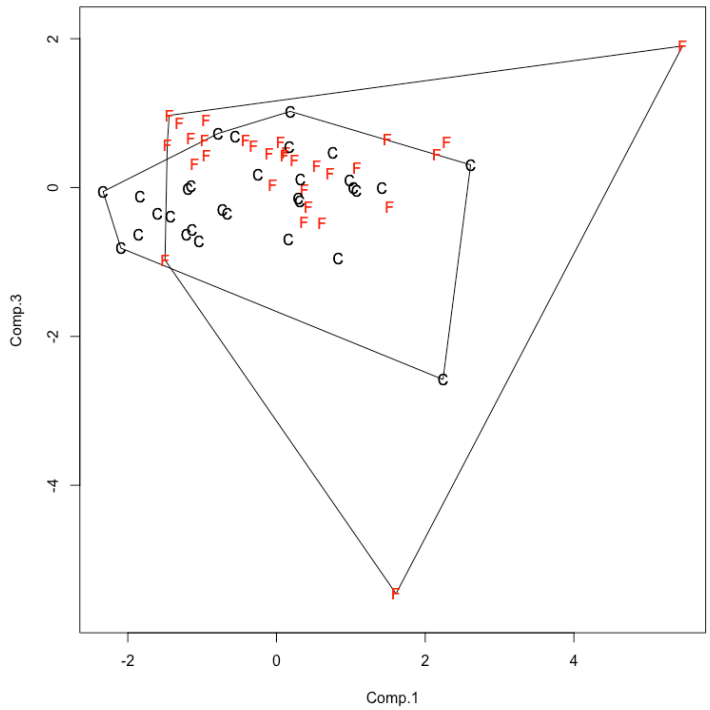
Figure 1. Ordiplot of the first two PCA axes. Forest fragment sites are in black (F) and continuous control sites are in red (C). There seems to be a reasonable amount of overlap between the two types of sites.

A PCA generates one axis per variable that was entered into it. There were six variables so we expect six axes. The first axis will explain the most variation, the second axis will explain the next most variation and so on. If we want to make use of the axes, we need to move them back to our original dataset.

Try these lines and look to see what happens:

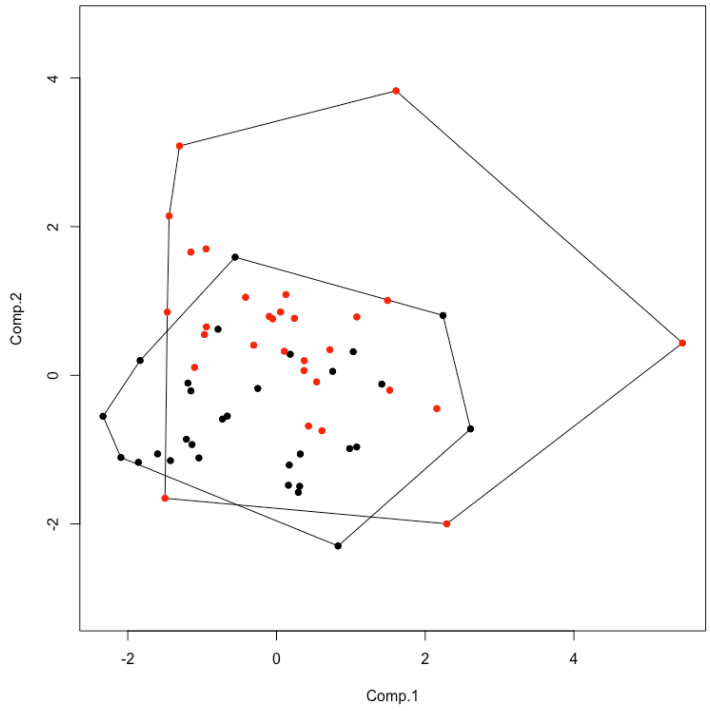
```
ordihull(ordiplot(veg.pca, type = "none", choices=c(1,3)),
agilis$HABITAT)
```

```
text(veg.pca$scores[,1],veg.pca$scores[,3],
labels=agilis$HABITAT, col=as.numeric(agilis$HABITAT))
```



What if you want to plot points instead of labels or letters? Have a go at this:

```
ordihull(ordiplot(veg.pca,type="none"),agilis$HABITAT)

points(veg.pca$scores, col=as.numeric(agilis$HABITAT), pch=16)
```



You can move the scores to the dataset also. The scores will be slotted in next to each observation.

```
agilis$pca.1 <- veg.pca$scores[,1]
agilis$pca.2 <- veg.pca$scores[,2]
agilis$pca.3 <- veg.pca$scores[,3]
agilis$pca.4 <- veg.pca$scores[,4]
agilis$pca.5 <- veg.pca$scores[,5]
agilis$pca.6 <- veg.pca$scores[,6]
```

Now check how the agilis data has changed.

```
str(agilis)
head(agilis)
View(agilis)
```

Classically, the scores are all assumed to be independent, and as such they can be analysed in a linear model like an ANOVA or ANCOVA without breaking assumptions. Have a go at the following, and interpret them. What do you think is probably the hypothesis and null hypothesis of the following tests? (**sqrtAGILIS** is a measure of agile antechinus abundance at sites.)

```
summary(aov(pca.1~HABITAT, data=agilis))

summary(aov(sqrtAGILIS~pca.1*pca.2*pca.3, data=agilis))

summary(aov(sqrtAGILIS~pca.1+pca.2+pca.3, data=agilis))
```

I've stated 'classically' above, because there is increasingly some debate about how independent pca scores really are. However, keep in mind that the reason we care about independence is that we don't want predictors to be explaining the same variance in a model. If you were concerned, you could always use **cor** to check the correlations among your pca axes and remove an axes if you find correlations > 0.6. Note that the 'independence' being discussed here is not the same as sample or replicate independence, which required for all statistical tests and is a function of good experimental design.

RESULT

```
> summary(aov(pca.1~HABITAT, data=agilis))
             Df Sum Sq Mean Sq F value Pr(>F)
HABITAT       1   7.45   7.451   4.063 0.0461 *
Residuals   118 216.40   1.834
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(aov(sqrtAGILIS~pca.1*pca.2*pca.3, data=agilis))
                  Df Sum Sq Mean Sq F value  Pr(>F)
pca.1              1  0.006  0.0058   0.102 0.75025
pca.2              1  0.442  0.4421   7.727 0.00638 **
pca.3              1  0.145  0.1453   2.540 0.11384
pca.1:pca.2        1  0.014  0.0144   0.252 0.61659
pca.1:pca.3        1  0.163  0.1628   2.846 0.09438 .
pca.2:pca.3        1  0.042  0.0420   0.735 0.39321
pca.1:pca.2:pca.3  1  0.098  0.0978   1.709 0.19381
Residuals        112  6.408  0.0572
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(aov(sqrtAGILIS~pca.1+pca.2+pca.3, data=agilis))
             Df Sum Sq Mean Sq F value  Pr(>F)
pca.1         1  0.006  0.0058   0.100 0.75182
pca.2         1  0.442  0.4421   7.625 0.00669 **
pca.3         1  0.145  0.1453   2.506 0.11611
Residuals   116  6.725  0.0580
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first ANOVA suggests that PCA1 is significantly associating with HABITAT (P = 0.046). The second ANOVA suggests that PCA2 is a significant predictor of agile antechinus abundance (sqrtAGILIS) (P = 0.006). The third ANOVA is examining the same relationship, but the non-significant interaction terms have been removed to simplify the model. PCA2 is still significantly associating with agile antechinus abundances (P=0.007).

PCA Loadings

`veg.pca$loading` shows the contribution of each variable to each axis. You can read the values as if they were correlation coefficients (i.e. the range from -1 to +1 where numbers that are further from zero have a strong association with an axes).

RESULT

```
> veg.pca$loading

Loadings:
              Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
ABH.m2         0.560 -0.165  0.302         -0.149 -0.739
median.ABH     0.156  0.471 -0.797  0.154 -0.123 -0.282
STUMPS         0.489  0.240  0.186  0.651  0.353  0.347
SHRUB.COUNT    0.473 -0.245 -0.300 -0.556  0.540  0.159
SHRUB.SPECIES  0.276 -0.602 -0.296  0.217 -0.556  0.343
pcNONEUC      -0.353 -0.520 -0.249  0.441  0.487 -0.334

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

Q. What is the strongest positive contributor to axis 1?

ABH.m2
Area at base height of trees has a strong positive 0.560 contribution to PCA1.

Q. What is the strongest negative contributor to axis 2?

SHRUB.SPECIES
Shrub species richness has a strong negative -0.602 contribution to PCA2.

Q. Which variable is not contributing to axis 4?

ABH.m2
Area at base height of trees is missing from PCA4 (i.e. it's contribution to this axes is zero).

Cumulative summing

```
(cumsum(veg.pca$sd^2))/6        # where 6 = number of axes
```

RESULT

```
   Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
0.3109087 0.5353282 0.6896440 0.8094749 0.9143534 1.0000000
```

This is the cumulative amount of variation explained by the axes. There are a number of axes equal to the variables that were included. However, we won't want to keep all the axes for analysis because there is a diminishing return on how much variation in the overall pattern of data is being explained by each axis.

Eigenvalues (keep axis if Eigenvalues > 1)

```
veg.pca$sd^2
```

RESULT

```
   Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6
1.8654520 1.3465174 0.9258945 0.7189857 0.6292710 0.5138793
```

You can read the Eigenvalues as being something like percentages of variation in the data explained by the axes, except that they are written as proportions, and the total variation available to explain is equal to the number of components x 100. I terms of the practical implantation, this use of Eigenvalues relates to one purpose for PCAs: variable reduction. We started with six environmental variables, but, we have concerns that some of these variables are co-correlating, and we would like to reduce these variables down to a smaller number (maybe 2-3) synthetic variables that capture a biological gradient or trend. To decide which axes to keep, we check the Eigenvalues. Any axes with an Eigenvalue >1 is explaining more than 100% of the variable we would expect it to explain, all things being equal. That is, it is explaining more than it's 'fair share' of variation. So, a very straightforward rule is to keep any axes with Eigenvalue >1 in your subsequent analyses, and discard the rest of the eaxes.

```
plot(veg.pca) # plots the Eigenvalues
```

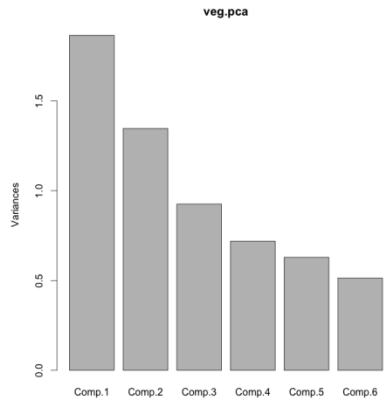


Figure 1. Plot of the Eigenvalues for the PCA.

Another way to view Eigenvalues is to plot them directly from the pca.

Proportion of variance explained by axes

```
summary(veg.pca)
```

RESULT

```
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4    Comp.5     Comp.6
Standard deviation     1.3658155 1.1603954 0.9622341 0.8479303 0.7932661 0.71685376
Proportion of Variance 0.3109087 0.2244196 0.1543158 0.1198310 0.1048785 0.08564655
Cumulative Proportion  0.3109087 0.5353282 0.6896440 0.8094749 0.9143534 1.00000000
```

A note on 'construct validity'

In psychology, PCAs are often used to check whether questions on a survey are contributing in a balanced way to a 'construct', that is an underlying psychological trait (such as empathy, or anxiety). This means that there is quite a bit of advice online about how to asses construct validity in a PCA, and when to keep or remove variables. Because we tend to use PCA in biological sciences to look at actual, real living systems, 'construct validity' is less of a concern, and in actuality, some of the things that would be problematic in psychology (such as a variable contributing to just one principal component axes), would be of genuine biological interest in a PCA derived from environmental gradients, morphology or genetic expression. This means that rather than remove variables that are not forming part of a coherent pattern, we tend to be interested in interrogating why a given variable appears to be (relatively) independent of the other biological variables included in the PCA. That is, it is generally preferable in biology to leave all variables in the PCA that you included in the first place, and treat these as a sort of *a priori* inclusion in a model, that then can be examined using significance tests.

Analysis

We're going to use these axes as explanatory variables in a regression analysis of agile antechinus abundances. The abundances have already been transformed for normality (**sqrtAGILIS**) so you should be able to simply apply the regression analysis without needing to further transform the data.

The PCA axes are independent and normally distributed by default

Normally, the rule of thumb is to retain all axes that explain more than their 'fair share' of variance, which will roughly equate with an Eigenvalue of > 1. I've retained axis 3 as well, because although it has an Eigenvalue of < 1, it isn't much below 1 and it is probably worth looking at, even if just quickly (i.e axes 3 might not make it into a final report, but my preference is just to check Eigenvalues >0.9 just to see if there is anything of interest there).

```
abundance.lm <- lm(sqrtAGILIS ~ pca.1* pca.2 * pca.3,
data = agilis)

summary(abundance.lm)
```

RESULT

```
> summary(abundance.lm) ## No significant interactions ##

Call:
lm(formula = sqrtAGILIS ~ pca.1 * pca.2 * pca.3, data = agilis)

Residuals:
     Min       1Q   Median       3Q      Max
-0.45095 -0.15859 -0.03536  0.15458  0.67305

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        0.393394   0.030317  12.976   <2e-16 ***
pca.1             -0.018010   0.023524  -0.766   0.4455
pca.2             -0.059790   0.035798  -1.670   0.0977 .
pca.3             -0.007091   0.078351  -0.091   0.9281
pca.1:pca.2       -0.008623   0.029305  -0.294   0.7691
pca.1:pca.3       -0.024637   0.025813  -0.954   0.3419
pca.2:pca.3       -0.050280   0.032221  -1.560   0.1215
pca.1:pca.2:pca.3  0.030042   0.022981   1.307   0.1938
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2392 on 112 degrees of freedom
Multiple R-squared:  0.1244,  Adjusted R-squared:  0.06966
F-statistic: 2.273 on 7 and 112 DF,  p-value: 0.03349
```

No significant interactions. Best to remove the interaction terms at this point (unless they were a part of you original hypothesis).

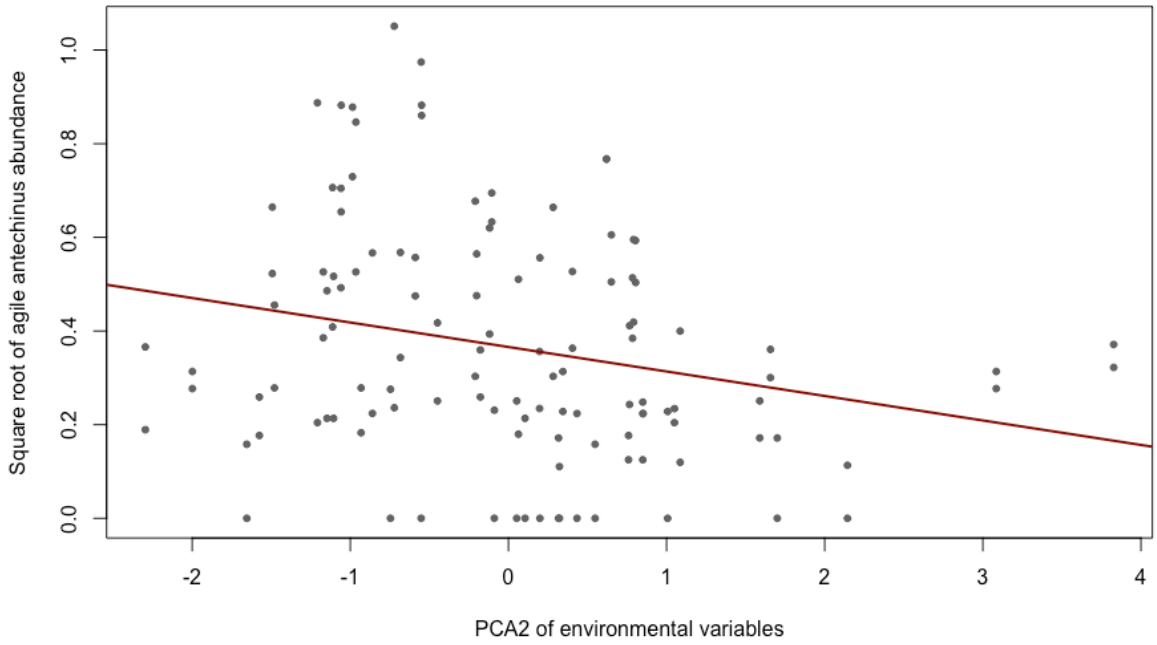
```
abundance.lm <- lm(sqrtAGILIS ~ pca.1 + pca.2 + pca.3,
data = agilis)

summary(abundance.lm)
```

RESULT

```
Call:
lm(formula = sqrtAGILIS ~ pca.1 + pca.2 + pca.3, data = agilis)

Residuals:
     Min       1Q   Median       3Q      Max
-0.49533 -0.16478 -0.02824  0.13692  0.67158

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.365900   0.021980  16.647  < 2e-16 ***
pca.1       -0.005101   0.016093  -0.317  0.75182
pca.2       -0.052306   0.018942  -2.761  0.00669 **
pca.3       -0.036164   0.022843  -1.583  0.11611
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2408 on 116 degrees of freedom
Multiple R-squared:  0.08106, Adjusted R-squared:  0.05729
F-statistic: 3.411 on 3 and 116 DF,  p-value: 0.01991
```

So, this leaves us with a significant intercept (not especially meaningful), and a significant effect of PCA2 on agile antechinus abundance. The effect size is negative, but we can plot the relationship to see what is happening.

```
plot(sqrtAGILIS~pca.2, data = agilis, pch = 20, col = "grey40",
ylab = "Square root of agile antechinus abundance", xlab = "PCA2
of environmental variables")

abline(lm(sqrtAGILIS~pca.2,  data  =  agilis),  col="darkred",
lwd=2)
```



In order to interpret the result we need to look at the loadings for PCA2.

RESULT

```
> veg.pca$loading

Loadings:
              Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
ABH.m2         0.560 -0.165  0.302        -0.149 -0.739
median.ABH     0.156  0.471 -0.797  0.154 -0.123 -0.282
STUMPS         0.489  0.240  0.186  0.651  0.353  0.347
SHRUB.COUNT    0.473 -0.245 -0.300 -0.556  0.540  0.159
SHRUB.SPECIES  0.276 -0.602 -0.296  0.217 -0.556  0.343
pcNONEUC      -0.353 -0.520 -0.249  0.441  0.487 -0.334

               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

PCA2 has an overall negative association with agile antechinus abundance (t = -2.76, P = 0.007). As values of PCA2 increase, there were declines in total area at base height of trees (loading = -0.165), shrub count in a 20x20 quadrat (loading = -0.245), shrub species richness (loading = -0.602) and percentage of non-eucalyptus species in the tree stand (loading = -0.560). In contrast the median area at base height of trees (loading = +0.471) and number of tree stumps (loading = +0.240) increased with increasing values of PCA2. This suggests that agile antechinus were at the highest abundances in sites with a large sum of total tree areas (typical of densely treed areas), large numbers of shrubs that had greater species diversity, as well as greater overall tree diversity (as indexed by percentage of trees that were not eucalyptus species). Sites that had a large median individual tree area (perhaps indicating stands where 2-3 large trees were dominating a given area), and stumps (indicative of disturbance), tended to have lower abundances of agile antechinus.

As stumps tend to be indicative of tree felling, and non-eucalyptus trees tends to be the more valuable timber (e.g. blackwoods) in the area studied, and are also individually smaller trees species that eucalyptus on average, this PCA gradient may be an index of human disturbance and selective logging and milling of commercially valuable non-eucalyptus trees.

Now have a go at these analyses:

```
veg.pca <- princomp(~ ABH.m2 + median.ABH + STUMPS +
SHRUB.COUNT + SHRUB.SPECIES + TREE.SPECIES + WOODY.DEBRIS +
CANOPY + MIDSTOREY + UNDERSTOREY + GROUNDCOVER + LEAF.LITTER +
pcDEAD + pcNONEUC, cor=T, data=agilis)
```

Move the scores to the dataset

```
agilis$pca.1  <- veg.pca$scores[,1]
agilis$pca.2  <- veg.pca$scores[,2]
agilis$pca.3  <- veg.pca$scores[,3]
agilis$pca.4  <- veg.pca$scores[,4]
agilis$pca.5  <- veg.pca$scores[,5]
agilis$pca.6  <- veg.pca$scores[,6]
agilis$pca.7  <- veg.pca$scores[,7]
agilis$pca.8  <- veg.pca$scores[,8]
agilis$pca.9  <- veg.pca$scores[,9]
agilis$pca.10 <- veg.pca$scores[,10]
agilis$pca.11 <- veg.pca$scores[,11]
agilis$pca.12 <- veg.pca$scores[,12]
agilis$pca.13 <- veg.pca$scores[,13]
agilis$pca.14 <- veg.pca$scores[,14]
```

Try using the PCA axes (you pick how many) as an explanatory variable for agile antechinus abundance in a regression analysis (**lm**). You could try examining bush rat abundances by using **sqrtFUSCIPES** as a response instead of **sqrtAGILIS**.

prcomp

The function **prcomp** is generally preferred, as it is considered mathematically more reliable than **princomp**. The **prcomp** function differs only in some minor code differences.

princomp()	prcomp()	Description
sdev	sdev	Standard deviations of the principal components
loadings	rotation	Matrix of variable loadings (columns are eigenvectors)
center	center	Variable means (means that were subtracted)
scale	scale	Variable standard deviations (the scalings applied to each variable)
scores	x	The coordinates of the individual observations on the principal component

The following is the code necessary to run the same analysis as above, but using **prcomp** instead of **princomp**.

Load libraries

```
library(vegan)
library(MASS)

agilis <- read.table('agilis-abundance.csv',
header=T, sep=',')

str(agilis)
veg.pca <- prcomp(~ ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT
+ SHRUB.SPECIES + pcNONEUC, scale=T, data=agilis)

biplot(veg.pca) # just picks the first two axes

ordihull(ordiplot(veg.pca, type="none"),agilis$HABITAT)
text(veg.pca$x, labels=agilis$HABITAT,
col=as.numeric(agilis$HABITAT))

ordihull(ordiplot(veg.pca, type = "none",
choices=c(1,3)),agilis$HABITAT)

text(veg.pca$x[,1],veg.pca$x[,3],
labels=agilis$HABITAT, col=as.numeric(agilis$HABITAT))

ordihull(ordiplot(veg.pca, type="none"), agilis$HABITAT)
points(veg.pca$x, col = as.numeric(agilis$HABITAT), pch = 16)
```

You can move the scores to the dataset also. The scores will be slotted in next to each observation.

```
agilis$pca.1 <- veg.pca$x[,1]
agilis$pca.2 <- veg.pca$x[,2]
agilis$pca.3 <- veg.pca$x[,3]
agilis$pca.4 <- veg.pca$x[,4]
agilis$pca.5 <- veg.pca$x[,5]
agilis$pca.6 <- veg.pca$x[,6]
```

Now check how the agilis data has changed.

```
str(agilis)
head(agilis)
View(agilis)

veg.pca$rotation # contribution of each variable to each axis

plot(veg.pca) # plots the Eigenvalues

summary(veg.pca) # summary of the Eigenvalues
```

Comparison of prcomp and princomp

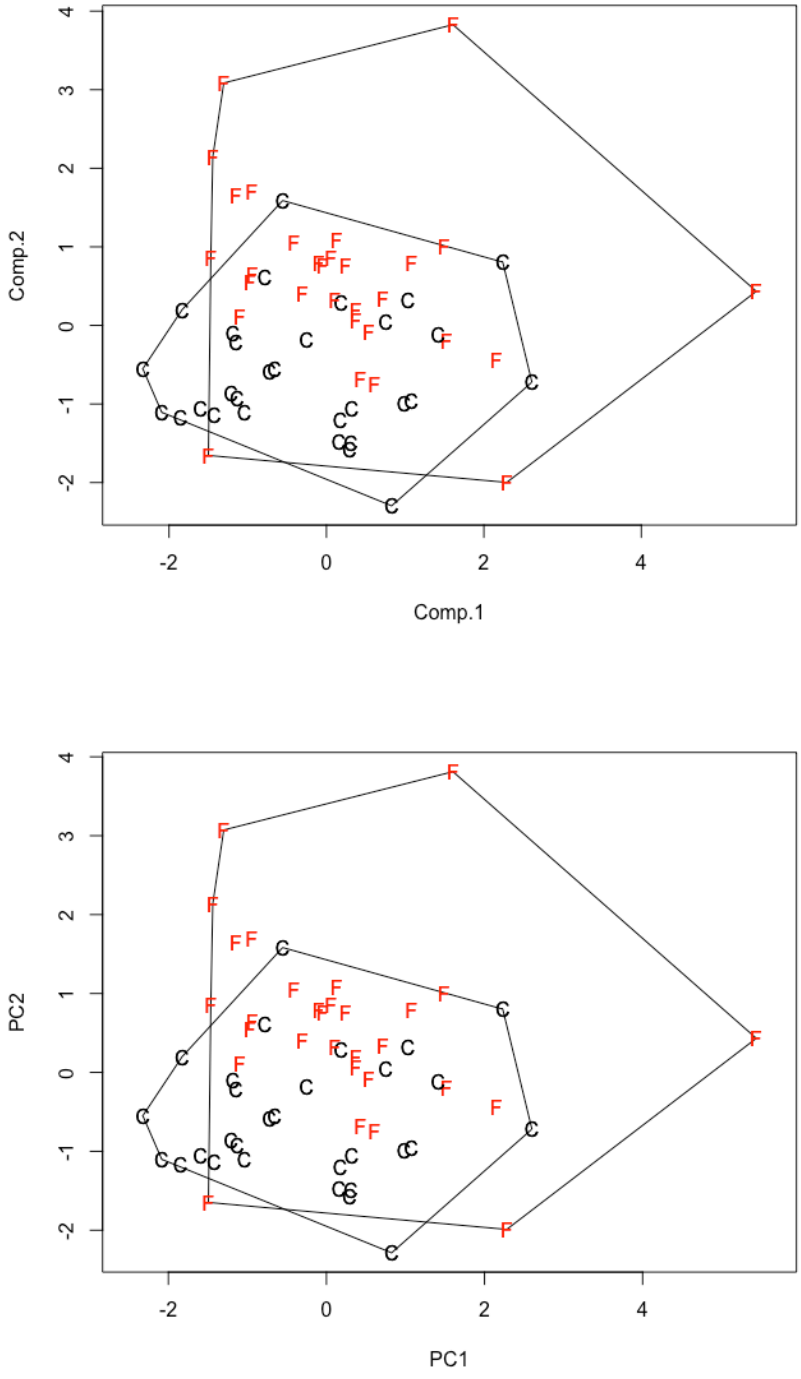


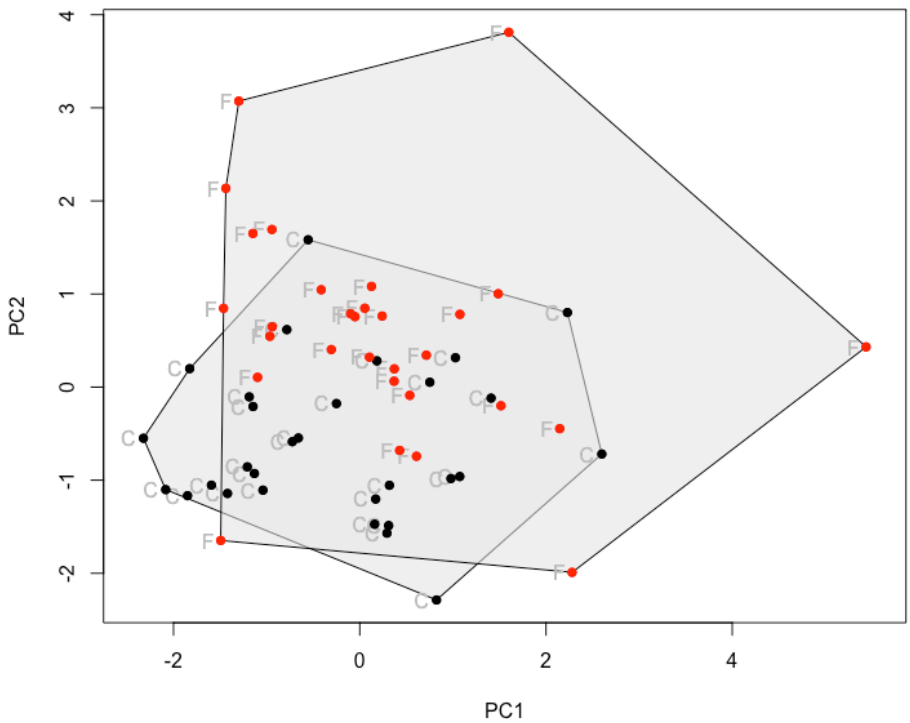
Figure 1. Ordiplot of the first two PCA axes using princomp (ABOVE) and prcomp (BELOW). In this case, the two methods seem to have generated identicle results.

Ordination Plots

A major use of PCA outputs is to graph groups and look for overlaps. This exploratory approach takes the view that groups with substantial overlaps are more similar in terms of the underlying axes, whereas groups that are disparate are more distinct. We'll use forest fragmented and control sites to look at the degree of overlap using our environmental variables measures at sites.

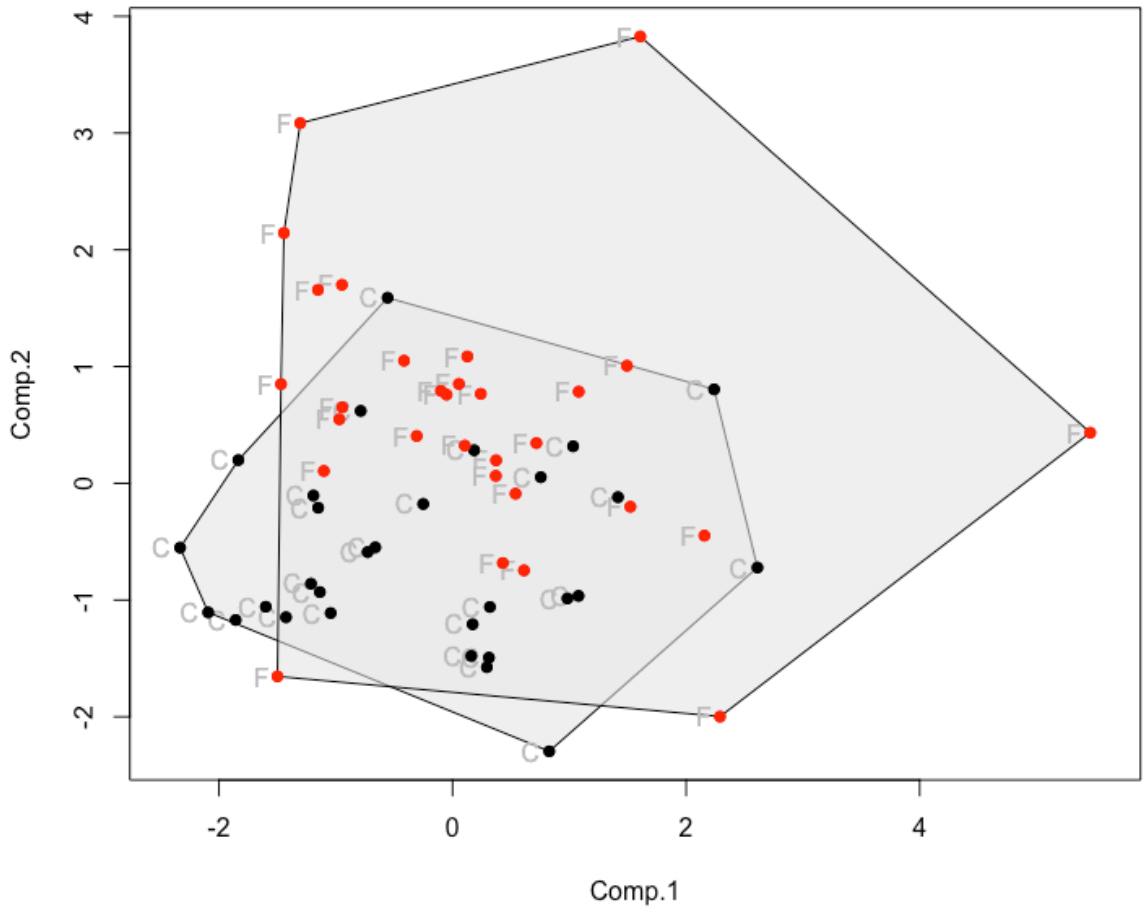
Further graphing: prcomp (polygons)

```
veg.pca <- prcomp(~ ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT
+ SHRUB.SPECIES + pcNONEUC, scale=T, data=agilis)

ordiplot(veg.pca,type="n") # blank canvas plot
ordihull(veg.pca, groups=agilis$HABITAT, draw="polygon",
col="grey90", label=F) # Polygons with no labels

text(veg.pca$x,lab=agilis$HABITAT,col="grey", adj =1.5) # Text
added in grey with a slight offset of 1.5 characters

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$x, pch=16,col=as.numeric(agilis$HABITAT))  #
Add points
```



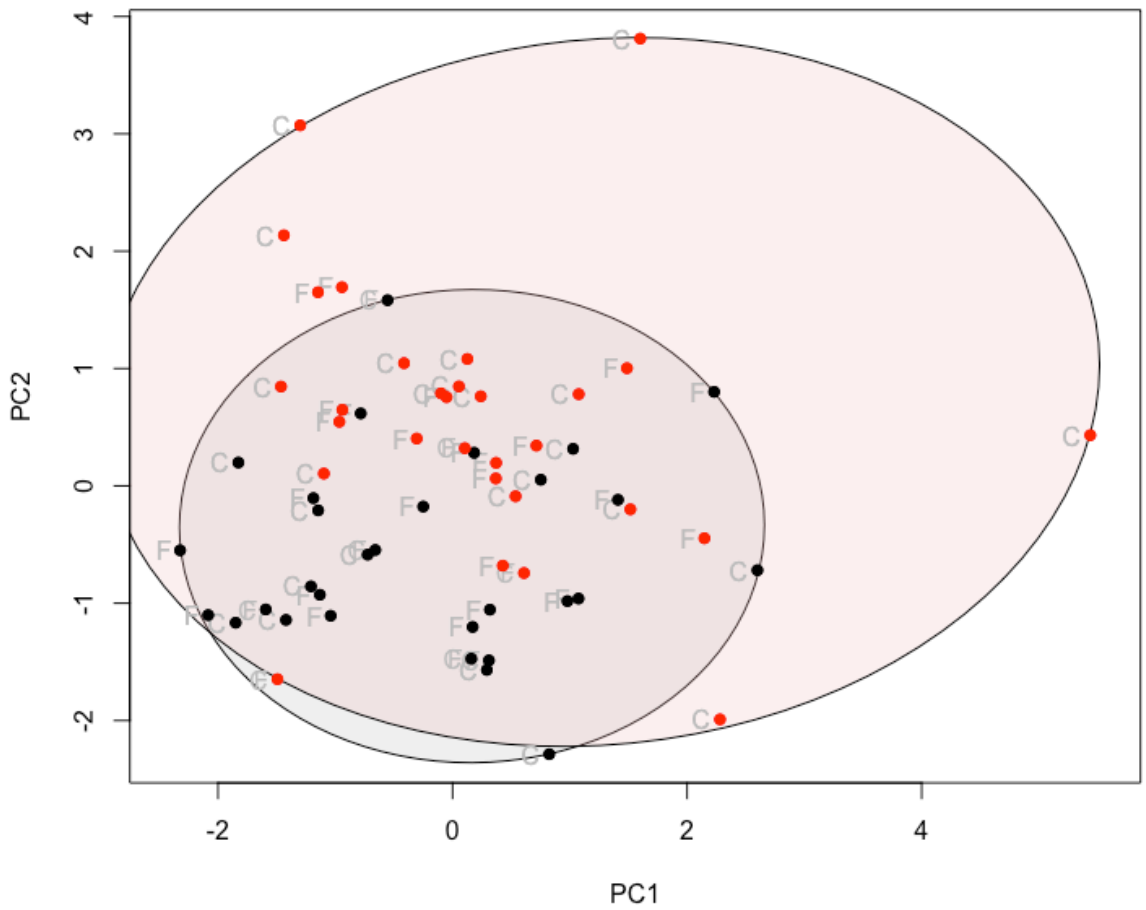
Further graphing: princomp (polygons)

```
veg.pca <- princomp(~ ABH.m2 + median.ABH + STUMPS +
SHRUB.COUNT + SHRUB.SPECIES + pcNONEUC, cor=T, data=agilis)

ordiplot(veg.pca,type="n") # blank canvas plot
ordihull(veg.pca, groups=agilis$HABITAT, draw="polygon",
col="grey90", label=F) # Polygons with no labels

text(veg.pca$scores,lab=agilis$HABITAT,col="grey", adj =1.5) #
Text added in grey with a slight offset of 1.5 characters

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$scores, pch=16,col=as.numeric(agilis$HABITAT))
# Add points
```



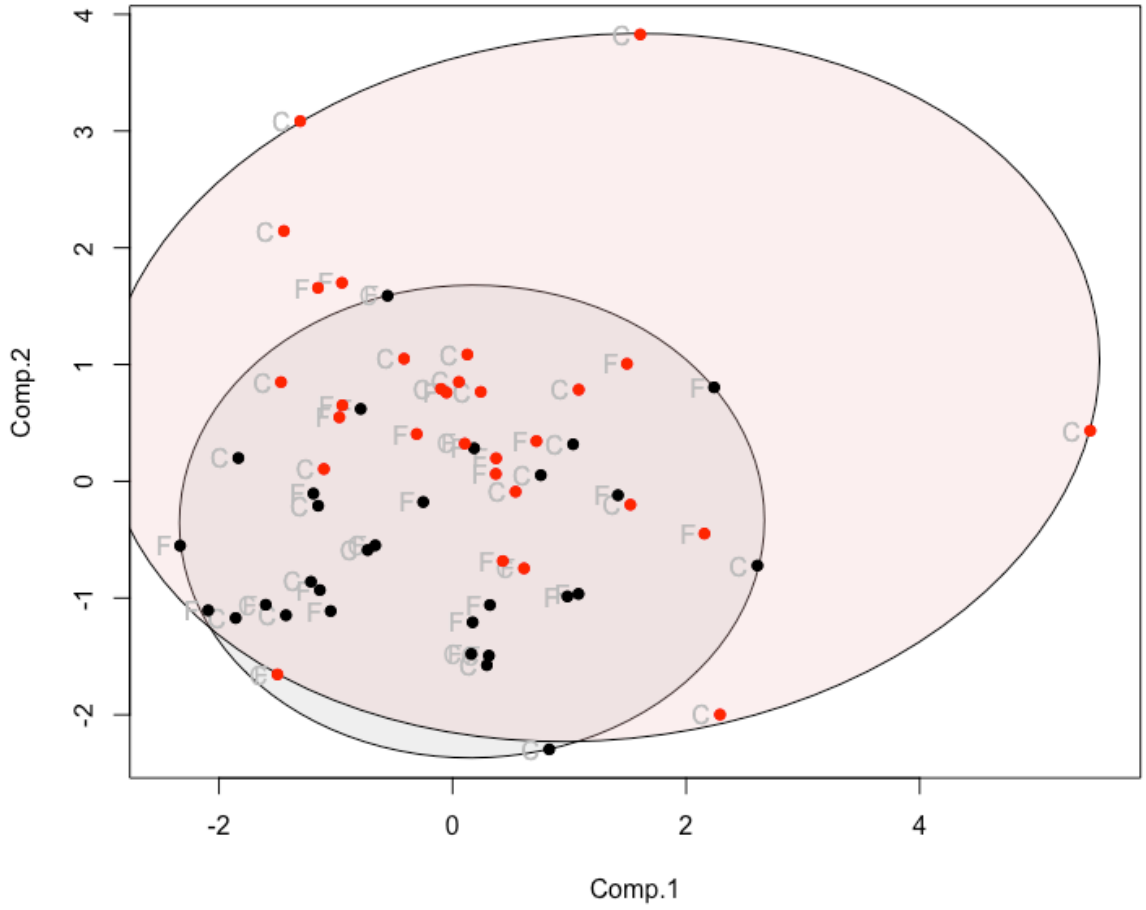
Further graphing: prcomp (ovals)

```
veg.pca <- prcomp(~ ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT
+ SHRUB.SPECIES + pcNONEUC, scale=T, data=agilis)

ordiplot(veg.pca,type="n") # blank canvas plot

ordiellipse(veg.pca, groups=agilis$HABITAT, draw="polygon",
col="grey90", label=F, kind="ehull") # Polygons with no labels


text(veg.pca$x,lab=agilis$HABITAT,col="grey", adj =1.5) # Text
added in grey with a slight offset of 1.5 characters

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$x, pch=16,col=as.numeric(agilis$HABITAT))  #
Add points
```



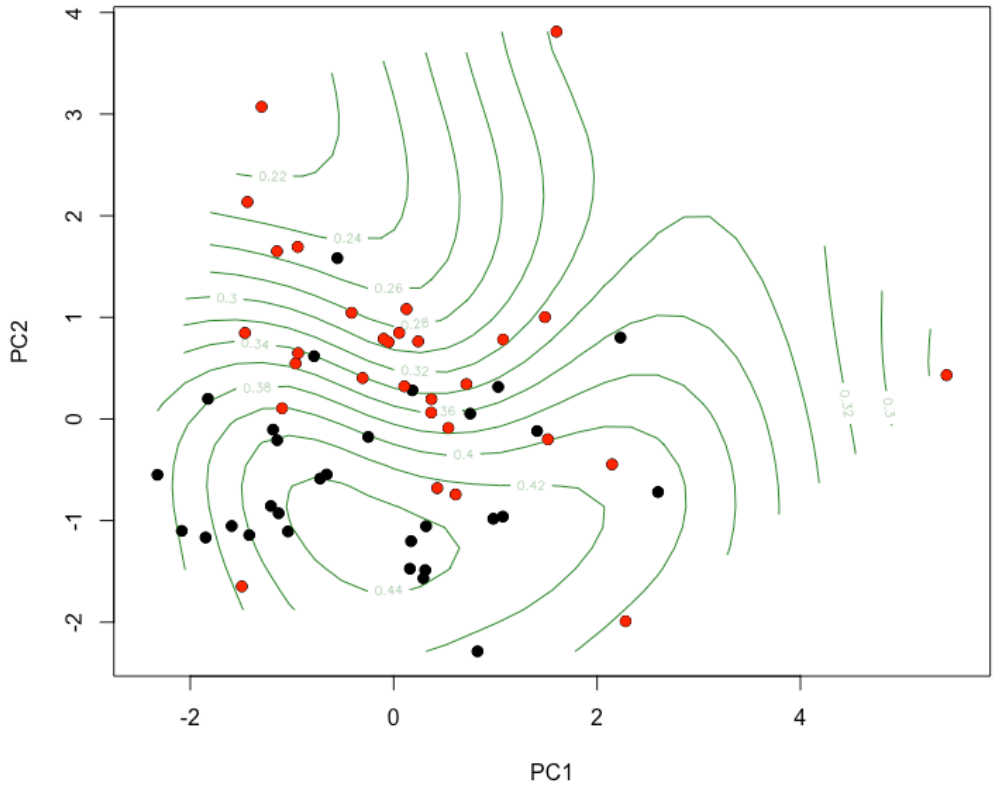
Further graphing: princomp (ovals)

```
veg.pca <- princomp(~ ABH.m2 + median.ABH + STUMPS +
SHRUB.COUNT + SHRUB.SPECIES + pcNONEUC, cor=T, data=agilis)

ordiplot(veg.pca,type="n") # blank canvas plot

ordiellipse(veg.pca, groups=agilis$HABITAT, draw="polygon",
col="grey90", label=F, kind="ehull") # Polygons with no labels

text(veg.pca$scores,lab=agilis$HABITAT,col="grey", adj =1.5) #
Text added in grey with a slight offset of 1.5 characters

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$scores, pch=16,col=as.numeric(agilis$HABITAT))
# Add points
```



Further graphing: prcomp (isobars)

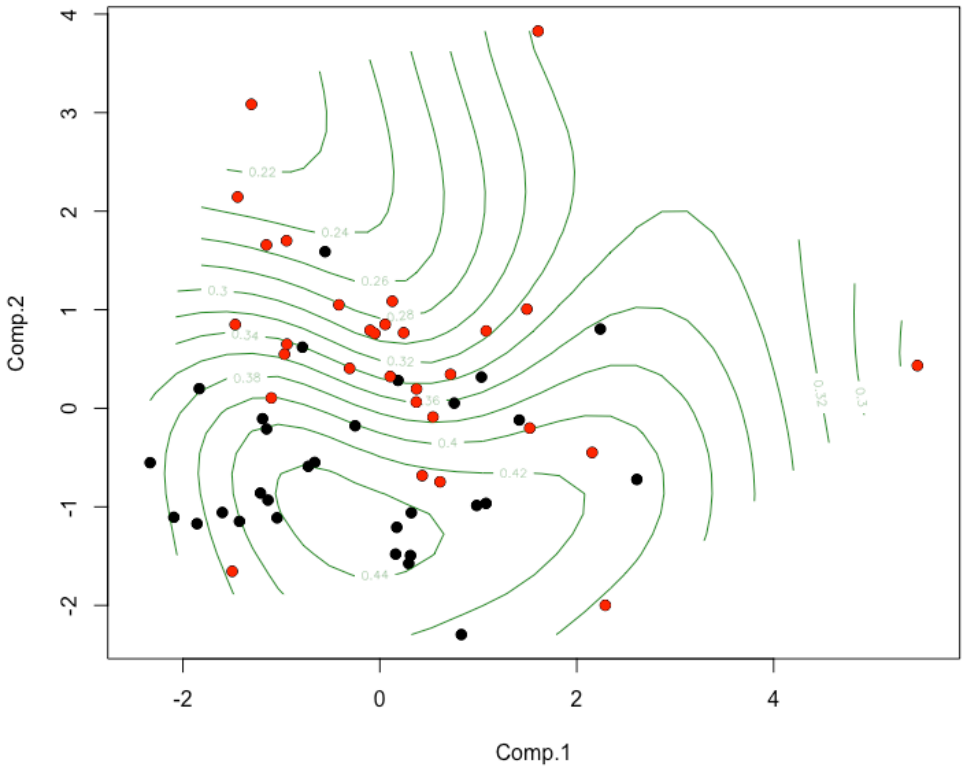
```
veg.pca <- prcomp(~ ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT
+ SHRUB.SPECIES + pcNONEUC, scale=T, data=agilis)

ordisurf(veg.pca,agilis$sqrtAGILIS, main="",col="forestgreen")

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$x, pch=16,col=as.numeric(agilis$HABITAT))  #
Add points
```



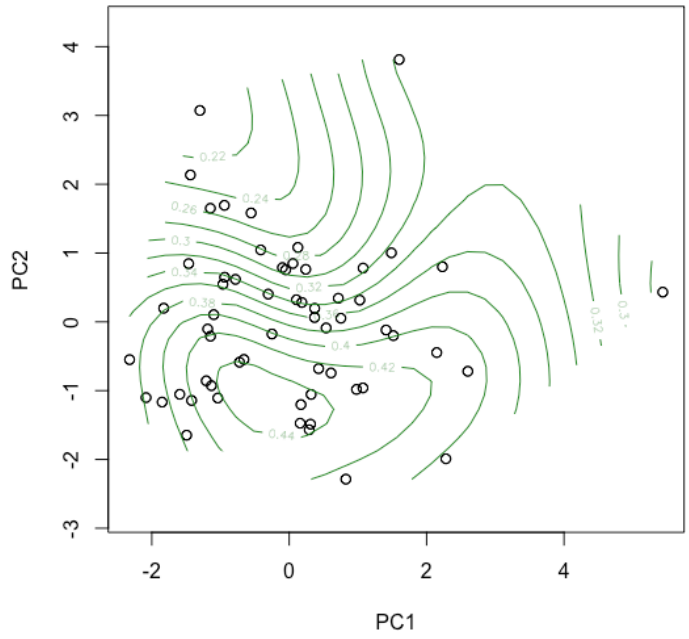
Agile antechinus abundance (sqrtAGILIS) plotted as a set of isobars over the first and second axes for the PCA. The peak abundhance is 0.44 sqrt captures per trap, which is highest in the cluster of continuous (black) forest sites to the bottom left.

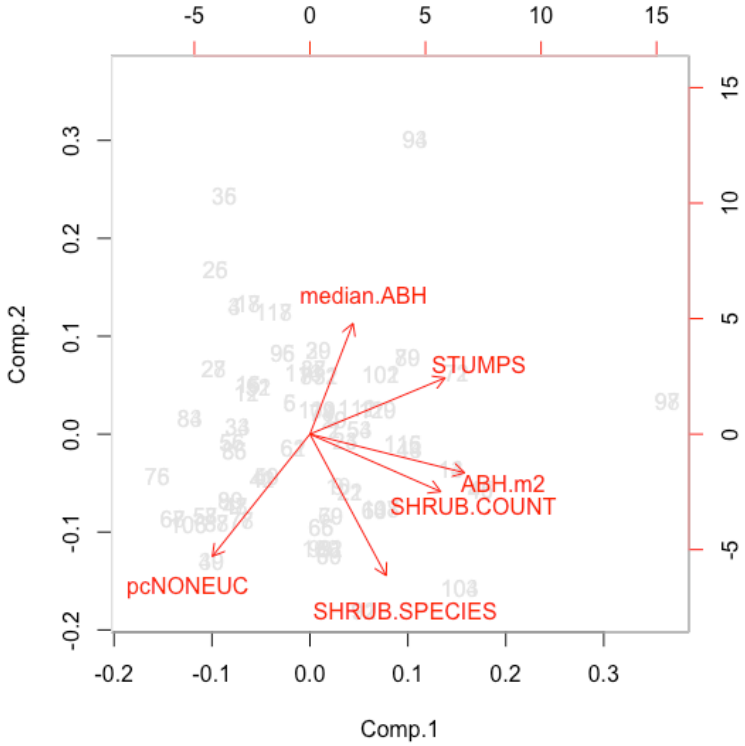
Further graphing: princomp (isobars)

```
veg.pca <- princomp(~ ABH.m2 + median.ABH + STUMPS +
SHRUB.COUNT + SHRUB.SPECIES + pcNONEUC, cor=T, data=agilis)

ordisurf(veg.pca, agilis$sqrtAGILIS, main="",
col="forestgreen")

agilis$HABITAT<-as.factor(agilis$HABITAT) # Change habitat to
a factor to get the next step to work

points(veg.pca$scores, pch=16,col=as.numeric(agilis$HABITAT))
# Add points
```



Note that the high abundance seems to correspond to a greater percentage of non-eucalyptus tree species and greater shrub diversity.






```
biplot(veg.pca, col=c("grey90","red"))
```

More fancy graphing for PCAs

Install and load libraries (ggbiplot is currently in beta so it has to be downloaded as a beta)

```
install.pacakges("ggplot2")
library(ggplot2)
install.packages("devtools")
library(devtools)
install_github('vqv/ggbiplot')
library(ggbiplot)
```

Load data

```
agilis <- read.table('agilis-abundance.csv',
header=T,sep=',')

str(agilis)
```

Construct a PCA

```
veg.pca <- prcomp(~ABH.m2 + median.ABH + STUMPS + SHRUB.COUNT +
SHRUB.SPECIES+pcNONEUC,scale=T,data=agilis)
```

Set up your response category as a separate factor for ease of coding

```
HABITAT<- agilis$HABITAT
```

Create a basic plot for axis 1 and 2 based on ggplot

```
g<-ggbiplot(veg.pca,obs.scale = 1,var.scale =
1,groups=HABITAT,ellipse = TRUE)
```

Have a look at it

```
print(g)
```

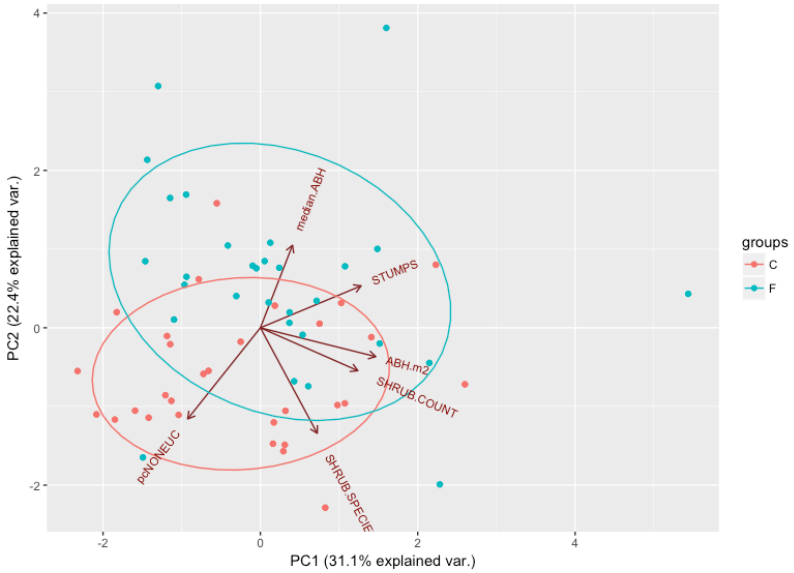
Add some modifications to the plot

```
g<-g+scale_color_discrete(name = '')+theme_bw()
```

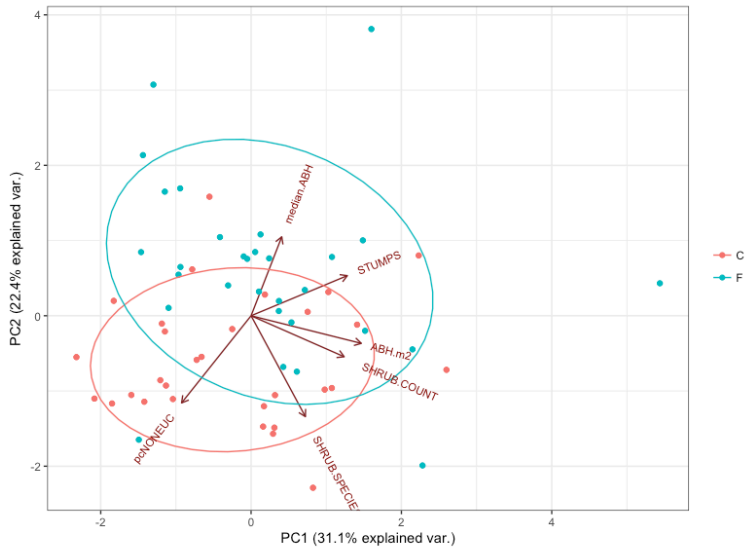
Have a look at the new plot

```
print(g)
```

PCA plot, version one...



PCA plot, version two with modifications...



MANOVA

A MANOVA (multivariate analysis of variance) is typically reported alongside PCA to provide P values. It is a test of multiple numeric (ideally continuous) responses against a single categorical predictor. The assumptions of the MANOVA should (strictly speaking) be met for the PCA to be valid too, and it is certainly worth checking them if you are running a PCA, regardless of whether you plan to use a MANOVA.

ASSUMPTIONS OF MANOVA (& PCA too, really)

(1) Observations must be independent

(2) Univariate assumptions of ANOVAs are met

- Normal distributions of dependent variables within groups
- Equal variances of dependent variables within groups

(3) Multivariate normality is met

(4) Multivariate equal variances are met

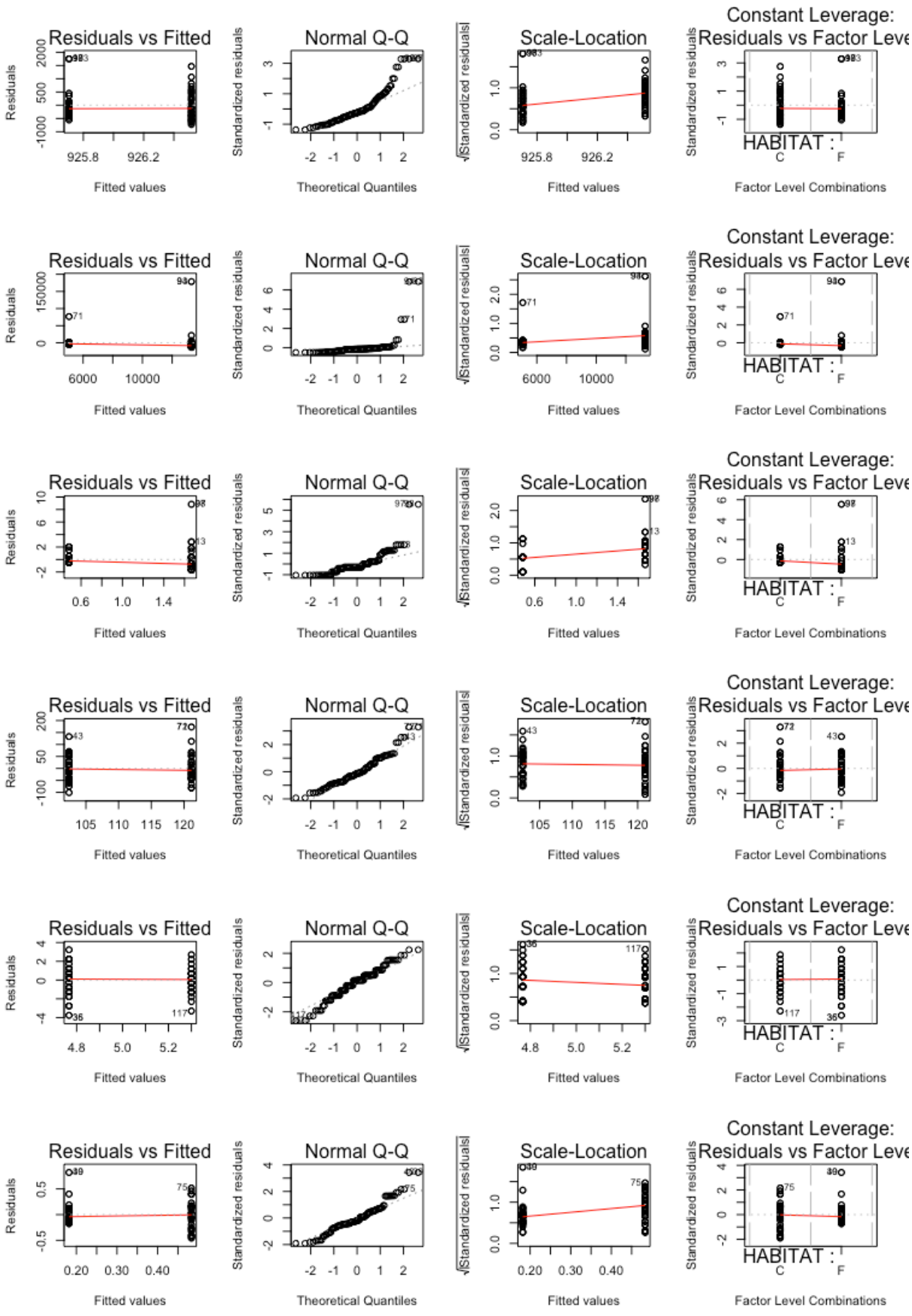
To test that the univariate assumptions of ANOVAs are met, the most straightforward thing to do is generate a set of univariate ANOVAs and check the diagnostic plots for each MANOA. We will use the same environmental data we use above and generate diagnostic plots for all relationships of

```
agilis <- read.table('agilis-abundance.csv',
header=T, sep=',')

str(agilis)
```

1) Univariate assumptions of ANOVAs are met

```
par(mfrow=c(6,4))
plot(aov(ABH.m2~HABITAT, data = agilis))
plot(aov(median.ABH~HABITAT, data = agilis))
plot(aov(STUMPS~HABITAT, data = agilis))
plot(aov(SHRUB.COUNT~HABITAT, data = agilis))
plot(aov(SHRUB.SPECIES~HABITAT, data = agilis))
plot(aov(pcNONEUC~HABITAT, data = agilis))
```

The diagnostic plots are shown over-page. Arguably, some of the QQ plots are indicating non-normality of residuals, and the residuals-vs-fitted do look 'wedgy' in places, suggesting that variances are not equal. If we were publishing this data, we would consider transforming the responses. However, for simplicity of explanation, I'm just going to continue with the data as is. Note that I'm doing this to make the demonstration easier: the following plots really don't look great.


Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Standardized residuals
Theoretical Quantiles
Scale-Location
√|Standardized residuals|
Fitted values
Constant Leverage:
Residuals vs Factor Leve
Standardized residuals
HABITAT :
C
F
Factor Level Combinations

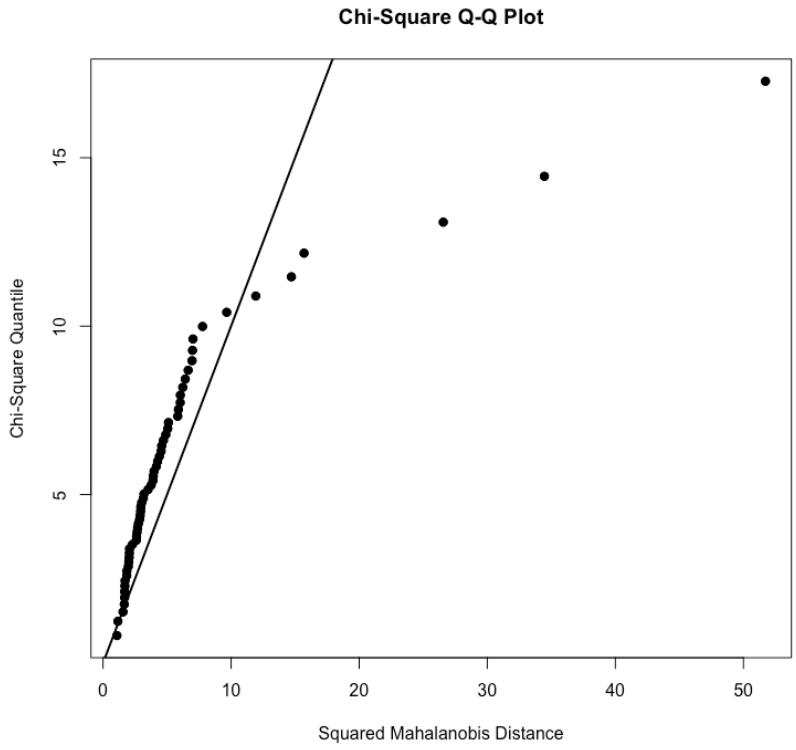
2) Multivariate normality is met

To check multivariate normality we need to set up a numeric matrix of responses.

```
attach(agilis)

y <- cbind(ABH.m2, median.ABH, STUMPS, SHRUB.COUNT, SHRUB.SPECIES,
pcNONEUC)
```

We will then use the `mvn` in library `MVN` to check multivariate normality. This is read in the same way as a standard univariate Shapiro-Wilks test, where P < 0.05 indicates a departure from normality. Other multivariate tests of normality, `hz`, `royston`, `dh` and `energy` are also available in the `MVN` package. The code is the same as below except that `mardia` is replaced with the name of the other test.

```
install.packages("MVN")
library(MVN)

par(mfrow=c(1,1))


mvn(y, mvnTest = "mardia", multivariatePlot = "qq")
# Marida's Multivariate Normality Test
# kurtosis & skew should not be significant
```



RESULT

```
      $multivariateNormality
              Test         Statistic                p value Result
1 Mardia Skewness 1264.84949032525 8.92474964193033e-228      NO
2 Mardia Kurtosis 30.5199762850272                      0      NO
3             MVN              <NA>                   <NA>      NO

$univariateNormality
           Test       Variable Statistic   p value Normality
1 Shapiro-Wilk     ABH.m2         0.8509  <0.001      NO
2 Shapiro-Wilk  median.ABH       0.2956  <0.001      NO
3 Shapiro-Wilk     STUMPS         0.6394  <0.001      NO
4 Shapiro-Wilk  SHRUB.COUNT      0.9577   8e-04      NO
5 Shapiro-Wilk SHRUB.SPECIES     0.9674  0.0052      NO
6 Shapiro-Wilk   pcNONEUC        0.8870  <0.001      NO
```

The mutlivariate test of normality is indicating that we have problems with both skewness and kurtosis (both a significant). The dataset definitely needs transformation, or might be more suitable for a non-parametric method, such as NMDS (described below). Let's persist and look at whether the multivariate variances are equal by group.

2) Multivariate equal variances

To check multivariate equal variances we will use the numeric matrix again. What we are checking here is whether there is equal covariances of the responses across the predictor groups we are interested in.

We will then use Box's M test of covariance matrices (`boxM` in library `biotools`).

```
install.packages("heplots")
library(biotools)

boxM(y, agilis$HABITAT)
```

RESULT

```
            Box's M-test for Homogeneity of Covariance Matrices

data:  y
Chi-Sq (approx.) = 166.68, df = 21, p-value < 2.2e-16
```

It appears that we also have unequal multivariate variances across the groups. This data is almost certainly not suitable for a MANOA. We would probably opt to use a NMDS and ANOSIM at this point, but for completion, let's run the MANOVA test and look at the results.

Running the MANOVA test

```
habitat.mva <- manova(y~HABITAT)
summary(habitat.mva)
```

RESULT

```
                     Df  Pillai approx F num Df den Df    Pr(>F)
HABITAT     1 0.41998   13.637       6    113 1.342e-11 ***
Residuals 118
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

There is a significant difference in environmental variables by habitat type (continuous or fragmented), but keep in mind that all of the assumptions were failed, so this result is not reliable.

The univariate results:

```
summary.aov(habitat.mva)
```

Results are shown over the page. There was a significant difference in stumps, shrub species richness and the percentage of trees that were not eucalyptus, but as per above, keep in mind that with the assumptions so badly failed, these results are not reliable. We definitely want to take this data and apply an NMDS approach instead.

RESULT

```
          > summary.aov(habitat.mva)
 Response ABH.m2 :
             Df   Sum Sq Mean Sq F value Pr(>F)
HABITAT       1       20      20   1e-04 0.9934
Residuals   118 33997556  288115


 Response median.ABH :
             Df     Sum Sq    Mean Sq F value  Pr(>F)
HABITAT       1 2.0413e+09 2041343300  2.9604 0.08795 .
Residuals   118 8.1366e+10  689546127
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


 Response STUMPS :
             Df  Sum Sq Mean Sq F value    Pr(>F)
HABITAT       1  42.008  42.008  16.262 9.819e-05 ***
Residuals   118 304.817   2.583
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


 Response SHRUB.COUNT :
             Df Sum Sq Mean Sq F value  Pr(>F)
HABITAT       1  10509 10509.4  3.7392 0.05555 .
Residuals   118 331652  2810.6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


 Response SHRUB.SPECIES :
             Df  Sum Sq Mean Sq F value  Pr(>F)
HABITAT       1   8.533  8.5333  4.0224 0.04719 *
Residuals   118 250.333  2.1215
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


 Response pcNONEUC :
             Df Sum Sq Mean Sq F value    Pr(>F)
HABITAT       1 2.7242 2.72422  46.902 3.582e-10 ***
Residuals   118 6.8538 0.05808
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nonmetric Multidimensional Scaling

One limitation of PCAs is that the result becomes more 'sub-optimal' the further away the data is from a set of normal distributions. Nonmetric multidimensional Scaling (NMDS) is another ordination approach that does not rely on normality of data. The disadvantage of NMDS compared to PCAs is that NMDS does not (easily) allow for 'variable reduction'--that is, there is no easy way to extract something similar to a PCA axis and compare the values to loadings for interpretation against statistical tests like ANOVAs or t-tests. Nonetheless, especially for species count data, where the data may be extremely non-normal, NMDS may be the best option available.

Load libraries

```
library(vegan)
library(MASS)
```

Load data. Remember to change your working directory first.

```
agilis <- read.table('agilis-abundance.csv', header=T,sep=',')
str(agilis)
```

First, we need to bind data together into a single set. We'll call this set **y**, although you could name it anything you want to. We will use the same data as was used for the PCA for comparison. This is a set of six environmental variables measured in eucalyptus forest sites in South Gippsland.

```
attach(agilis)
y <- cbind(~ABH.m2, median.ABH, STUMPS, SHRUB.COUNT,
SHRUB.SPECIES, pcNONEUC)
```

An NMDS won't work with duplicated rows. We can check this with the **duplicated** command. Here's what you would see if there were some duplicated rows.

```
duplicated(y)
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
FALSE  TRUE FALSE
```

We need to take a moment to think about why there are duplicated rows. The **unique** command can be used to strip out duplicates, but before you do this, you need to work out whether this is a sensible step. These might be data entry errors, or they might be real data duplicates. Here is the code, but don't do this just yet...

```
y <- unique(y)
duplicated(y)
```

But this is quite drastic. Is it a good idea to do here?

Why might we have duplicates? This is because the dataset is made up of male and female abundances, and males and females were (obviously) living in the same sites. So, the environmental variables are 'duplicated' for the two sexes. This means that the most sensible thing to do is probably to split by male and female and produce two NMDS (or average the data by site for the two sexes).

Keep in mind that we are generating two NMDS that should be identical. We wouldn't expect any difference between the female and male NMDS because they both are based on the same set of environmental variables. Producing two NMDS, rather than one, is just a matter of playing on the safe side, as we want to compare the NMDS results to the original dataset, and working with males and females separately is just a little safer.

Create subsets of only the variables you are interested in using for the basis of the NMDS.

```
female <- subset(agilis,subset=agilis$SEX=='F')
male <- subset(agilis,subset=agilis$SEX=='M')

attach(female)
female.y <- cbind(ABH.m2, median.ABH, STUMPS, SHRUB.COUNT,
SHRUB.SPECIES, pcNONEUC)

attach(male)
male.y <- cbind(ABH.m2, median.ABH, STUMPS, SHRUB.COUNT,
SHRUB.SPECIES, pcNONEUC)
```

Now, we will generate distances based on Bray-Curtis dissimilarities. These are non-parametric distances, which is one of the big advantages of NMDS over PCA. An NMDS is much more suitable for data like species counts than is a PCA, because species counts (for example) are often not normally distributed. The command is called 'vegdist' because the package was originally created for botanists.

```
female.dist <- vegdist(female.y, method = "bray")
male.dist <- vegdist(male.y, method = "bray")
```

You have a number of dissimilarity matric options. Use **`?vegdist`** to view them. You will need to do a bit of reading up on various approaches to decide which is best, but as a starting point Bray-Curtis dissimilarity matrices are generally well liked.

Perform the multidimensional scaling. The command **k=2** tells R to try and create two axes. Different numbers of axes will alter the 'stress' on the model (the degree to which the data is having to be twisted up to match the axes). We would tend to prefer stress in an NMDS to be below 0.15 or 15%. You can also try an iterative approach where you try **k = 2**, **k = 3**, **k = 4** and pick the number of axes with the lowest stress.

```
female.mds <-isoMDS(female.dist, k=2)
male.mds <-isoMDS(male.dist, k=2)
```

View the results. We'll just look at **female.mds** for now... the results are on the next page.

```
female.mds
```

A key thing to pay attention here is the 'stress'. You can think of this as a measure of the degree to which the data has to be contorted and twisted to fit into the number of axes you have asked for (here, 2 axes). The isoMDS function multiplies the stress by 100, so that the stress value of 4.8 should be read as either 4.8% or 0.048. The typical breakdown is that stress < 0.05 is an excellent representation of the original data in reduced dimensions; < 0.1 is great; <0.15 is good; < 0.2 is acceptable, and stress > 0.3 provides a very poor representation.We have a stress of 0.048, which is well below 0.15, which is the usual threshold that is used.

As a test, see what happens if you increase the number of axes... try this:

```
female.mds <-isoMDS(female.dist, k=3)
female.mds
```

and

```
female.mds <-isoMDS(female.dist, k=5)
female.mds
```

Does the stress increase or decrease? It should decrease with increasing numbers of axes, because the data doesn't need as much contortion to fit more axes. However, in our instance, we already have a perfectly good stress at k=2 (two axes), so we will stick with that. If your stress is 15%, the you may need to increase the number of axes to bring it under 15%. Notice also that as you increase the k, the number of columns (containing synthetic axis values) increases to match k.

RESULT

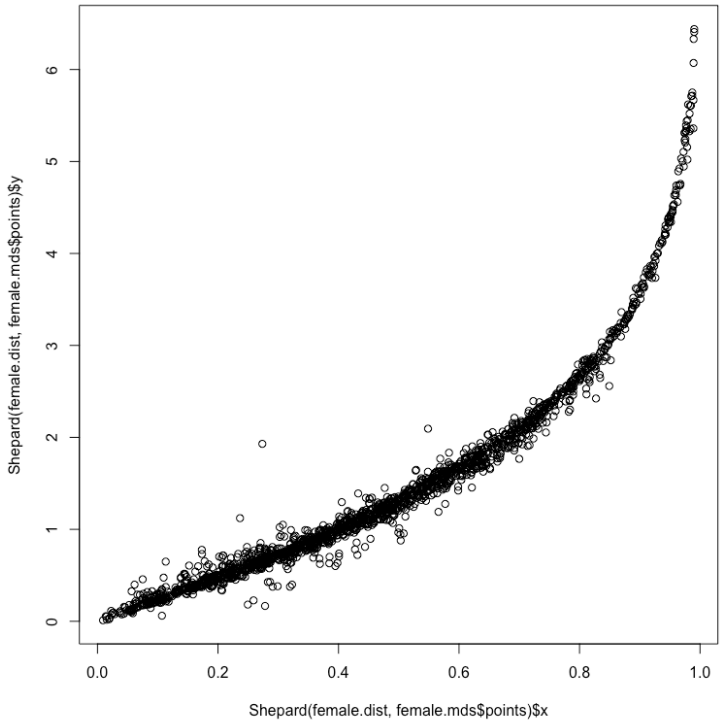
```
$points
             [,1]         [,2]
 [1,] -0.96615479  0.144868265
 [2,] -1.44357877 -0.000543712
 [3,] -0.21788957  0.082905390
 [4,] -1.82468080 -0.363232668
 [5,]  0.60275613  0.208649013
 [6,]  1.13070582  0.013311258
 [7,]  0.45859702 -0.097704434
 [8,]  0.17326574  0.405981457
 [9,] -1.26389869  0.022445365
[10,] -1.47021456 -0.289450802
[11,] -0.57205734  0.067410399
[12,] -0.20821781  0.028843803
[13,] -1.19447307  0.089615265
[14,]  0.56380012  0.335212120
[15,] -0.94317527  0.053315377
[16,] -1.55030903 -0.209722507
[17,]  1.58721462  0.124209746
[18,] -2.50460748 -0.538924605
[19,] -0.76625775 -0.047346446
[20,]  1.85246162  0.365160488
[21,]  0.75593823  0.418793619
[22,] -0.37849132 -0.076688801
[23,] -0.77491349 -0.335736327
[24,]  0.57780265  0.487023542
[25,]  0.31862263  0.371708129
[26,] -0.21904967  0.032116977
[27,]  0.69845165 -0.118854730
[28,]  0.87865561  0.573727596
[29,]  0.83292493  0.289940260
[30,]  0.24749676 -0.090787429
[31,]  0.99372597  0.115986274
[32,] -0.18146465 -0.157593561
[33,]  0.67243172 -0.428235084
[34,] -0.93018726  0.378949247
[35,]  0.23756426 -0.104480319
[36,] -3.17185459 -1.416230295
[37,]  0.08611587 -0.206890877
[38,]  0.66506936  0.488828549
[39,]  0.53450984  0.550959299
[40,] -1.02845028 -0.191097564
[41,]  0.99472534 -0.335686944
[42,]  0.03405531  0.568535584
[43,]  0.03856378  0.269935961
[44,]  1.05849896  0.179479569
[45,]  1.86119470  0.425256543
[46,]  1.35458597 -0.427008818
[47,] -4.05496686 -2.117757748
[48,] -0.29532518  0.109827988
[49,] -0.05328914  0.426629844
[50,]  1.29541224 -0.108002043
[51,] -0.96480595 -0.012379569
[52,]  1.19632978 -1.043637889
[53,]  1.71140122  0.498575204
[54,]  0.90976718 -0.182672910
[55,]  1.21162000  0.088424282
[56,]  0.47828599  0.229274784
[57,] -0.27916480  0.125922023
[58,]  1.32376763 -0.084686297
[59,]  0.65067383  0.379054257
[60,] -0.72951434  0.034474899

$stress
[1] 4.81661
```

Shepard Plot

A stress of 4.8 is read as 4.8% or 0.048. This is well below the usual 15% threshold, and so we can accept this NMDS as appropriate.

You can create a Shepard Plot, which in NMDS should appear as a monotonic series of points, and in a situation where dissimilarities are used (as in here) the plot should run from the bottom left to the top right (MDS can be based on similarities, but we won't cover this here).

```
plot(Shepard(female.dist,female.mds$points))
```



The Shepard plot should follow a smooth line or curve with a reasonably tight scatter around the line. We seem to have a reasonably nice-looking curve here, and our results are probably sound. If the Shepard plot looks like 'steps' or has a clear bend or L shape to it, then NMDS may not be suitable for the data, and other methods should be investigated.

Contribution of variables to axes

The 'envfit' command in library 'vegan' allows you to check the relative contribution of the original variables against your axes. It is call envfit because the authors assume you will be working with environmental data, but, of course it could be applied to anything that was used to build an NMDS (i.e. genetic or morphometric data would work fine too).

```
agilis.envfit <- envfit(female.mds,female.y, choices=c(1,2))
agilis.envfit
```

RESULT

```
***VECTORS

                   Dim1     Dim2     r2 Pr(>r)
ABH.m2          0.17196 -0.98510 0.2477  0.002 **
median.ABH     -0.26006 -0.96559 0.6692  0.001 ***
STUMPS         -0.99840 -0.05647 0.0340  0.336
SHRUB.COUNT     0.21907 -0.97571 0.0844  0.076 .
SHRUB.SPECIES   0.44290 -0.89657 0.2602  0.001 ***
pcNONEUC        0.98416  0.17727 0.2409  0.001 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
1
Permutation: free
Number of permutations: 999
```

The 'envfit' function runs a permutation test to identify associations between the measured variables (here vegetation measures) and the NMDS axes. The permutation test works by shuffling data and working out how often a test statistic of the magnitude of the observed from the actual data might occur by chance alone. The R^2 value is the proportion of times that the actual data has a greater magnitude than the random data. The results present:

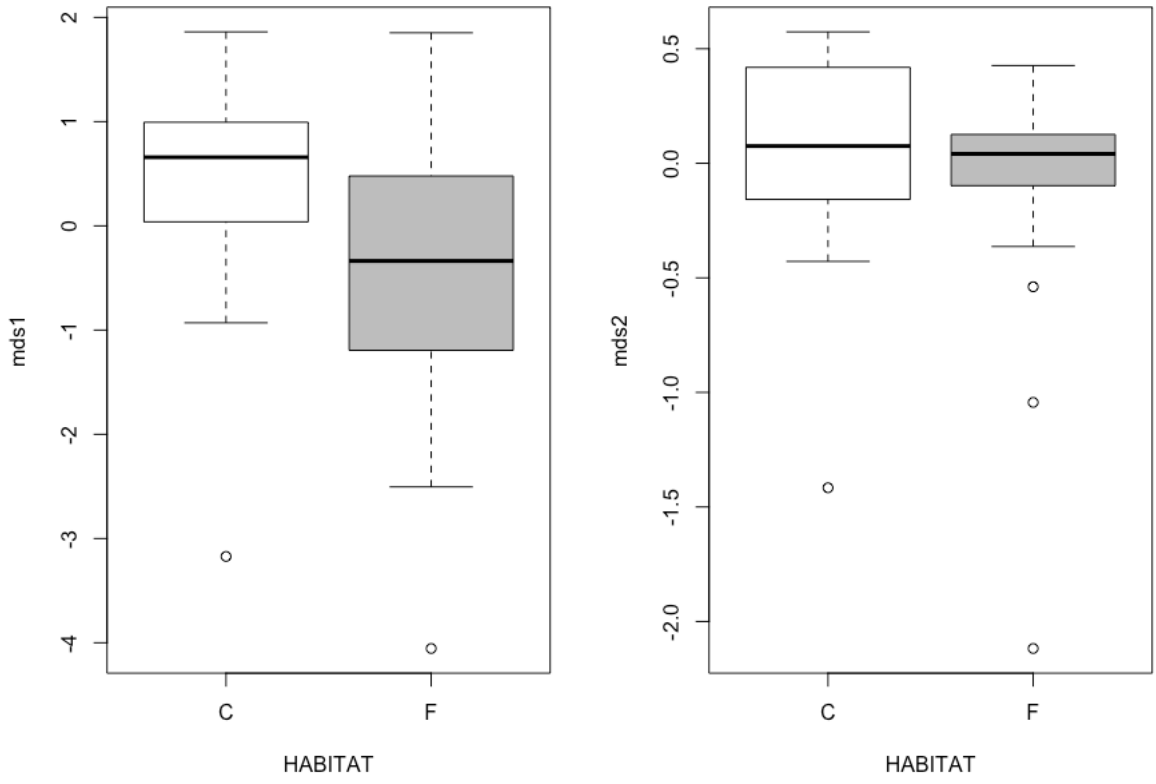
- Cosine of the angle to each axis: you can read these as if they were Pearson's r values. The closer to +1, the closer we are to a perfect positive association between a variable and an axis. The closer the -1, the closer we are to a perfect negative fit. In the above example, stumps has a -0.998 value for Dim1 and a -0.056 value for Dim2. This means that stumps is strongly aligning with Dim1 and as values of Dim1 increase, the number of stumps decreases. With regards to Dim2, stumps has no particular relationship. As a rule of thumb, anything below 0.4 is probably not worth interpreting.
- R^2 of the fit of the original data to the ordination.
- P value based on permutations.

This means that the R2 and the P value may be of only limited use to you. These are informative about whether a variable is showing a strong relationship with both axes combined. However, it may be interesting to examine axes one-by-one. In the above example Stumps is not significantly captured by the whole ordination, but it is strongly negatively associating with Dim1. Percentage on non-eucalyptus trees is also strongly associating with Dim1, but in the other direction. Non-eucalyptus trees in the study area are often high-value timber. It may not be a coincidence that the more stumps there are, the fewer non-eucalyptus trees there are. Trees like blackwoods may have been be selectively logged out of a forest stand, leaving only stumps behind. This relationship wouldn't be interpretable if we fixate on interpreting both axes together.

Of the output from the

Let's move the MDS1 and MDS2 (Dim1 and Dim2) values back into the dataset for visualisation and statistical use.

```
female$mds1<- female.mds$points[,1]
female$mds2<- female.mds$points[,2]

par(mfrow=c(1,2)) # set plotting window to a 1x2 array
boxplot(mds1~HABITAT,data=female, col=c("white","grey"))
boxplot(mds2~HABITAT,data=female, col=c("white","grey"))
par(mfrow=c(1,1)) # return plotting window to a 1x1 array
```



```
t.test(mds1~HABITAT,data=female)
t.test(mds2~HABITAT,data=female)
```

RESULT

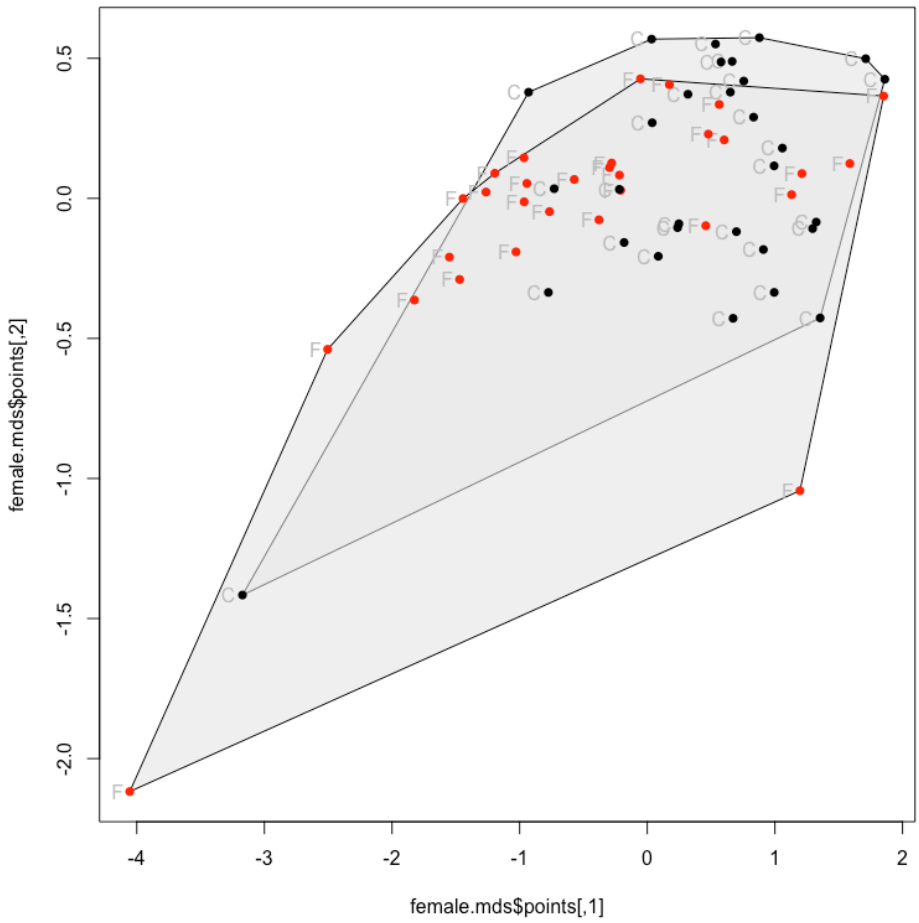
```
> t.test(mds1~HABITAT,data=female)

     Welch Two Sample t-test

data:  mds1 by HABITAT
t = 2.9138, df = 54.133, p-value = 0.005181
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 0.2646574 1.4320055
sample estimates:
mean in group C mean in group F
      0.4241657      -0.4241657

> t.test(mds2~HABITAT,data=female)

     Welch Two Sample t-test

data:  mds2 by HABITAT
t = 1.1684, df = 57.068, p-value = 0.2475
alternative hypothesis: true difference in means is not equal
to 0
95 percent confidence interval:
 -0.0983413  0.3738719
sample estimates:
mean in group C mean in group F
     0.06888266     -0.06888266
```

The interpretation is that there is a significant difference between Continuous and Fragmented forest for mds1 (P = 0.005), but there is no significant difference for MDS1 (P = 0.248). In order to understand what this means, we have to look at the envdist table and the boxplot on the previous page. Continuous forest fragments have high values of MDS1 and fragmented forest sites had lower values of MDS1, on average. High values of MDS1 associate with more non-eucalyptus trees (pcNONEUC), more shrub species and fewer stumps (all have cosine values stronger than 0.3). There is no strong association with area at breast height (ABH), median area at breast height (median.ABH) and shrub count. The implication is that continuous forest sites have significantly greater percentages of non-eucalyptus trees, more shrubs species richness and fewer stumps than do forest fragments.

Now, onto plotting the NMDS as an ordination. We'll do this for females only at this stage, as the Males and Females are going to look identical, as we are simply using the presence of antechinus at a site as a guide to where environmental variables were recorded. The groups are fragmented and continuous forest habitats.

An NMDS ordination plot using polygons

```
plot(female.mds$points,type="n") # blank canvas plot
ordihull(female.mds,groups=female$HABITAT, draw="polygon",
col="grey90", label=F) # Polygons with no labels

text(female.mds$points,lab=female$HABITAT,col="gray", adj =1.5)
# Text added in grey with a slight offset of 1.5 characters

female$HABITAT<-as.factor(female$HABITAT)
# Change habitat to a factor to get the next step to work

points(female.mds$points,pch=16,col=as.numeric(female$HABITAT))
# Add coloured points, using point character 16 (solid circle)
```

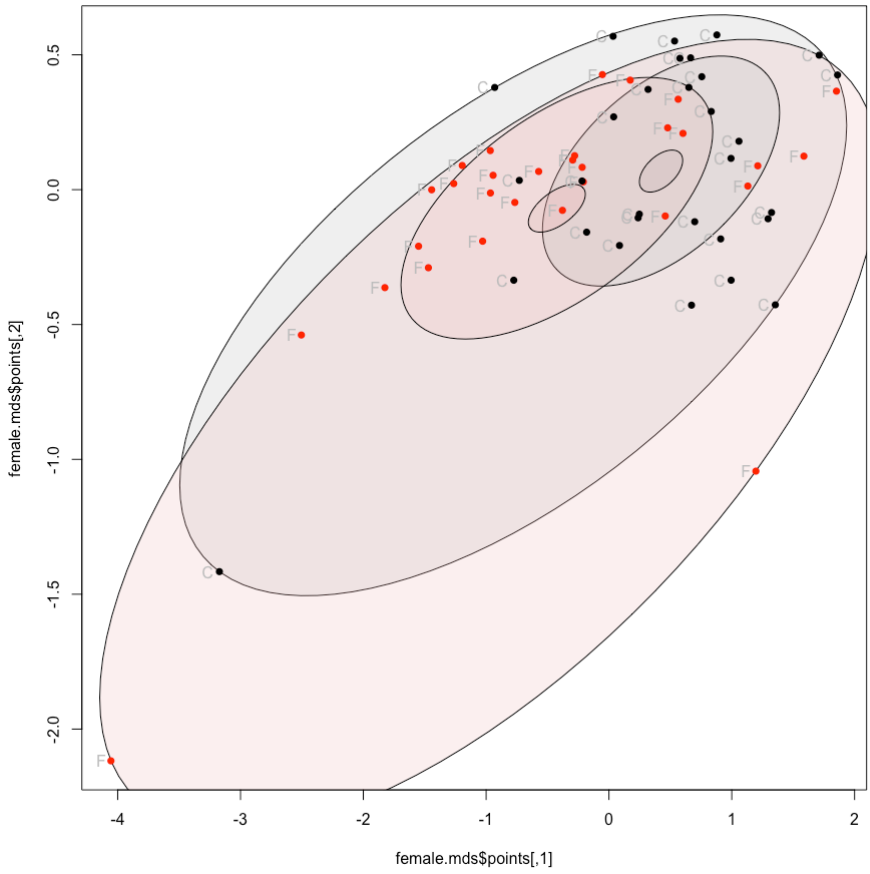


The interpretation is similar to any other ordination: the greater the overlap of polygons the more similar (overall) are the underlying variables measured at the sites.

An NMDS ordination plot using ellipses

The ehull command draws an ellipse around all points. The sd command draws an ellipse around the standard devition of the points. The se command draws an ellipse around the standard error of the points. I've left out the text labels, as they cluttered the figure enough to make it hard to read, but they could be added back in by removing the hash.

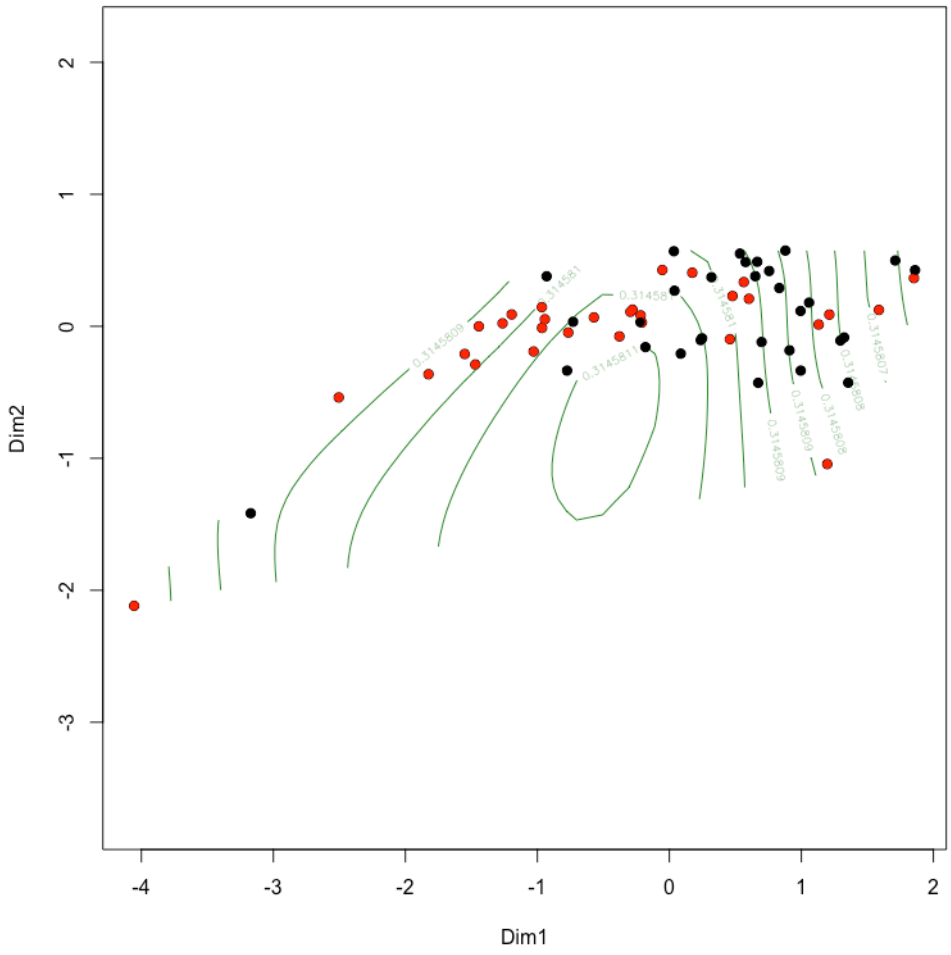
```
plot(female.mds$points,type="n")

ordiellipse(female.mds,groups=female$HABITAT,alpha =
50,draw="polygon",col=c("grey70","rosybrown2"),label=F,kind="ehull")

ordiellipse(female.mds,groups=female$HABITAT,alpha =
50,draw="polygon",col=c("grey70","rosybrown2"),label=F,kind="sd")

ordiellipse(female.mds,groups=female$HABITAT,alpha =
50,draw="polygon",col=c("grey70","rosybrown2"),label=F,kind="se")

text(female.mds$points,lab=female$HABITAT,col="gray", adj = 1.5)
# Text added with a 1.5 offset

female$HABITAT<-as.factor(female$HABITAT)
points(female.mds$points,pch=16,col=as.numeric(female$HABITAT)) # Add
coloured points, using point character 16 (solid circle)
```



An NMDS plot using isobars

Finally, we might be interested in placing a continuous variable onto the NMDS. In the case of the antechinus we have a measure of abundance taken from a square root of captures per trap (sqrtAGILIS). We can plot this for males and females: the figures should be different because although male and female antechinus were present at the same sites, they were present at different levels of abundance.

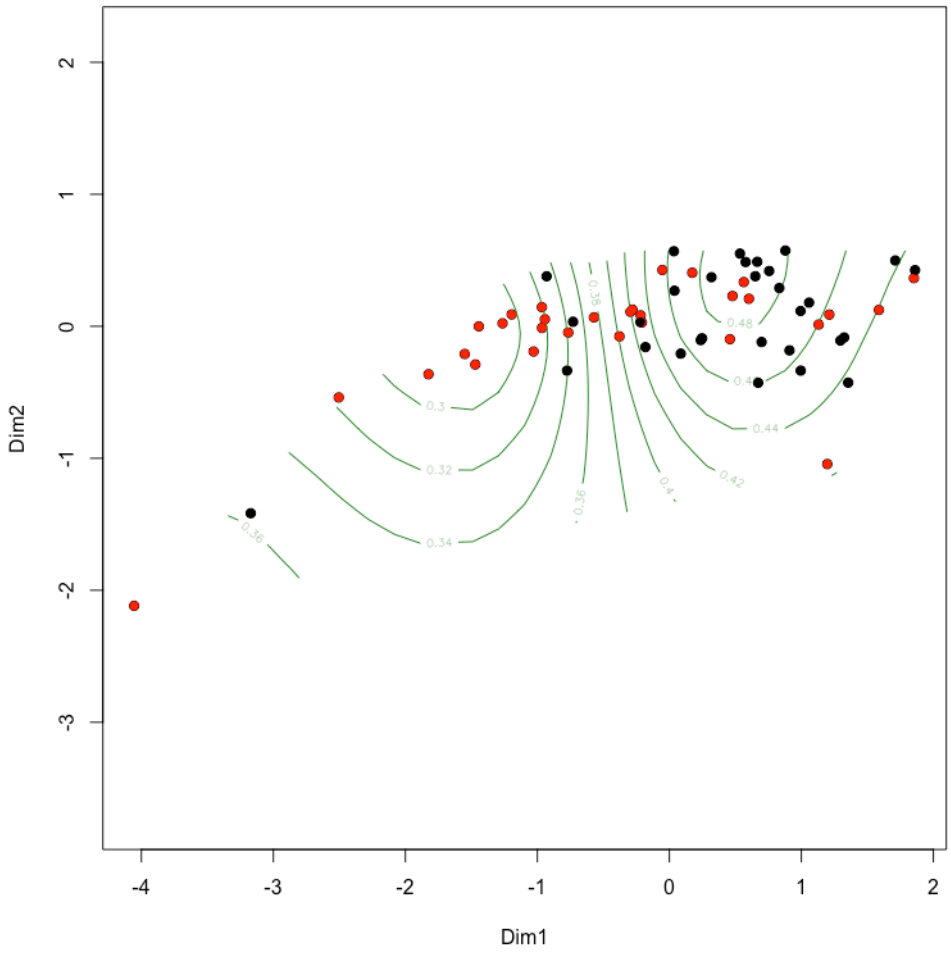
```
ordisurf(female.mds, female$sqrtAGILIS, main="", col="forestgreen")
female$HABITAT<-as.factor(female$HABITAT)
points(female.mds$points, pch=16, col=as.numeric(female$HABITAT)) # Add
coloured points using point character 16 (solid circle)
```



The isobars for females suggest a fairly flat abundance with a peak abundance of 0.3145811 sqrtAGILIS towards the middle of the distribution, not clearly in the fragmented (red) or continuous (black) forest.

The same plot for males...

```
ordisurf(male.mds, male$sqrtAGILIS, main="", col="forestgreen")
male$HABITAT<-as.factor(male$HABITAT)
points(male.mds$points, pch=16,col=as.numeric(male$HABITAT)) # Add
points using point character 16 (solid circle)
```



The isobars for males has a bit more steepness to it. There's a clear peak of 0.48 in the continuous forested sites (black), and a trough of 0.30 in the fragmented sites (red).

ANOSIM

An Analysis of Similarity is sometimes reported with a NMDS because it is based off the same fundamental methodology of generating a dissimilarity (or similarity) matrix. Continuing from the example above, we already created a distances matrix, but if you had not done so, you need to do this as a first step.

```
female.dist<-vegdist(female.y, method = "bray")
```

Create the ANOSIM

```
female.ano<-anosim(female.dist,female$HABITAT)  # Run an ANOSIM
```

Look at the results of the ANOSIM

```
summary(female.ano) # examine ANOSIM
```

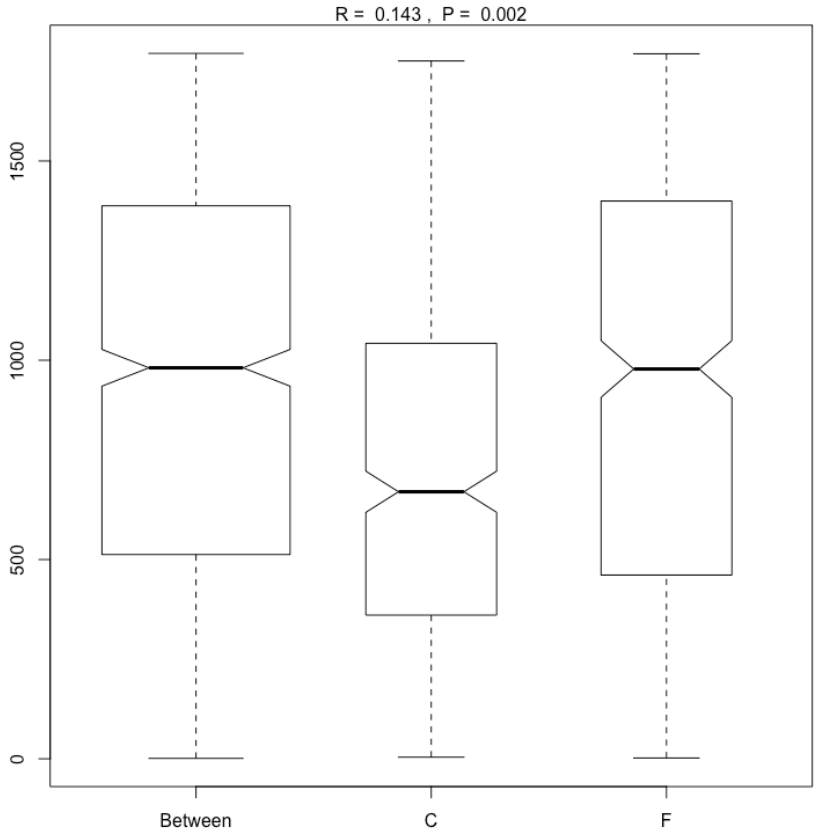
RESULT

```
Call:
anosim(dat = female.dist, grouping = female$HABITAT)
Dissimilarity: bray

ANOSIM statistic R: 0.1431
      Significance: 0.001

Permutation: free
Number of permutations: 999

Upper quantiles of permutations (null model):
   90%    95%  97.5%    99%
0.0269 0.0386 0.0545 0.0754

Dissimilarity ranks between and within classes:
         0%    25% 50%     75% 100%   N
Between   1 512.75 981 1387.25 1770 900
C         4 360.50 670 1042.50 1751 435
F         2 461.00 978 1399.50 1769 435
```

We have a significant result (P = 0.001) but, ANOSIMs are extremely powerful to the point of inflating Type I error. They tend to be significant more often than not. The relatively low R of 0.1431 requires a bit of caution with the interpretation.

Plot results of the ANOSIM

```
plot(female.ano)
```



The notches in the boxplots are read as per standard notched boxplots: if notches overlap horizontally, then the groups are not different. The notches do not overlap for continuous and fragmented forest, so we can consider these different habitat structures based on the environmental variables measured and included.

Survival analysis

Survival analysis deals with the time elapsed before an event occurs. In classically statistics, this is death (time to death after exposure to a poison, time to death when exposed to cold etc), but the analysis can be applied to any fixed event, such as the time it takes chicks in a nest to fledge (leave the nest) or the time it takes an animal to find hidden food in an enclosure.

Survival analysis therefore has two response variables. Whether or not the event occurred (1 = YES, 0 = NO) and the time elapsed before the time occurred. If trials are for a limited time (in the food searching experiment we might only run each trial to 45 min) then an animal that reaches the end of the trial without finding food would have a score of 0, 45 min. An animal that found the food in 10.5 min would have a score of 1, 10.5 min, and so on.


Parametric Survival analysis

For some reason we have decided that we really want to know whether agile antechinus (left) or yellow-footed antechinus (right) are better at running through mazes. We have three mazes (treatments T1, T2 and T3) and 30 individuals of each species.

Import the data and have a look at it.

```
maze <- read.table('survival.csv',header=T, sep = ',')
```

Look at the whole dataset so that you can see how it is laid out.

```
maze
```

The first thing we need to work out here are out predictor and response variables. In each trial (maze 1, maze 2, maze 3), there are two predictor variables:

- **Did the event happen?** (categorical, yes or no)
- **How long did it take, if the event happened?** (continuous measurement of time)

We have (potentially) two predictor variables:

- **Species** (a two level factor)
- **Body size** (continuous)

RESULT

```
> maze
    SPECIES INDIVIDUAL MASS.g T1.goal T1.time T2.goal T2.time T3.goal T3.time
1    agilis         A1     33       1     5.1       1    11.2       0    45.0
2    agilis        A10     33       1     2.4       1    11.2       1     2.1
3    agilis        A11     30       1    33.7       1    10.2       0    45.0
4    agilis        A12     31       1     9.6       1     1.7       1     2.3
5    agilis        A13     30       1     4.0       1    18.3       0    45.0
6    agilis        A14     29       1     2.5       1     1.6       1     9.8
7    agilis        A15     30       1     1.2       1     9.4       0    45.0
8    agilis        A16     34       1     4.3       1     4.4       0    45.0
9    agilis        A17     32       1    35.0       1     3.2       0    45.0
10   agilis        A18     34       1    30.8       1    16.2       1     9.1
11   agilis        A19     30       1    26.5       1     2.1       1     4.2
12   agilis         A2     33       1     1.9       0    45.0       1     9.4
13   agilis        A20     32       1     5.6       1    44.2       1     3.3
14   agilis        A21     29       1     4.3       1     2.3       1    19.2
15   agilis        A22     32       1     7.6       1     5.1       1     2.2
16   agilis        A23     37       0    45.0       1     2.1       0    45.0
17   agilis        A24     29       1    12.4       1    10.5       1     2.6
18   agilis        A25     28       1    14.0       1     0.6       0    45.0
19   agilis        A26     34       1     2.4       1     4.7       0    45.0
20   agilis        A27     30       1    33.7       1     4.2       1    15.3
21   agilis        A28     28       1    25.4       1     1.6       1    12.1
22   agilis        A29     32       1    17.2       1    15.0       1     9.3
23   agilis         A3     32       1     6.2       1    34.3       1     3.7
24   agilis        A30     34       1    37.5       1    10.8       1     3.1
25   agilis         A4     30       1    12.4       1     8.8       1    29.3
26   agilis         A5     34       1     9.6       1    18.8       1     0.3
27   agilis         A6     34       1    14.4       1    13.3       1    10.8
28   agilis         A7     35       0    45.0       1    17.6       0    45.0
29   agilis         A8     30       0    45.0       1     5.8       1     2.0
30   agilis         A9     31       1     0.9       1    24.2       1    16.5
31 flavipes         F1     35       1    20.8       1     0.8       0    45.0
32 flavipes        F10     34       0    45.0       0    45.0       0    45.0
33 flavipes        F11     32       0    45.0       1    10.1       0    45.0
34 flavipes        F12     36       1     8.5       1     2.3       0    45.0
35 flavipes        F13     30       1     6.8       0    45.0       0    45.0
36 flavipes        F14     30       1     5.0       1    28.2       0    45.0
37 flavipes        F15     32       1    11.7       0    45.0       0    45.0
38 flavipes        F16     30       1     8.9       0    45.0       0    45.0
39 flavipes        F17     30       1    40.9       0    45.0       1    42.5
40 flavipes        F18     32       0    45.0       1     5.1       0    45.0
41 flavipes        F19     33       1     5.9       1     8.1       0    45.0
42 flavipes         F2     34       0    45.0       1     8.0       0    45.0
43 flavipes        F20     30       1     5.2       0    45.0       0    45.0
44 flavipes        F21     43       1    43.2       0    45.0       0    45.0
45 flavipes        F22     32       0    45.0       1    31.0       0    45.0
46 flavipes        F23     34       1     7.8       1    17.6       1    15.6
47 flavipes        F24     31       0    45.0       1     4.5       0    45.0
48 flavipes        F25     33       0    45.0       1    22.3       0    45.0
49 flavipes        F26     33       1    29.2       0    45.0       1    23.2
50 flavipes        F27     27       1     9.3       1    14.6       0    45.0
51 flavipes        F28     35       1    38.6       0    45.0       0    45.0
52 flavipes        F29     32       0    45.0       1    28.7       0    45.0
53 flavipes         F3     30       0    45.0       1     5.8       1     8.8
54 flavipes        F30     40       1    29.9       1    10.5       0    45.0
55 flavipes         F4     34       1    36.4       0    45.0       0    45.0
56 flavipes         F5     33       1    42.0       1     5.4       0    45.0
57 flavipes         F6     30       1    32.5       0    45.0       0    45.0
58 flavipes         F7     34       0    45.0       0    45.0       0    45.0
59 flavipes         F8     34       0    45.0       0    45.0       0    45.0
60 flavipes         F9     34       0    45.0       1    13.2       0    45.0
```

We're going to have to use a package in R to do a survival analysis. The basic R setup doesn't support survival analyses very well:

```
library(survival)
```

Create a survival object

```
maze.T1.surv <- with(maze, Surv(T1.time,T1.goal))
```

From this we can create a survival curve. We will set colours for the two species. R always sets things like colours alphabetically, so first we'll check the categories:

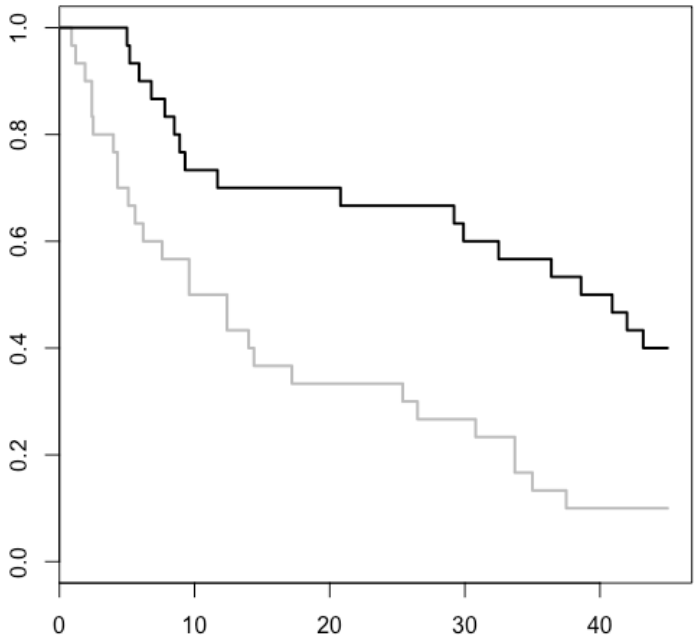
RESULT

```
levels(maze$SPECIES)
[1] "agilis"   "flavipes"
```

And now the plot:

```
maze.fit <- survfit(maze.T1.surv~SPECIES,data=maze)
plot(maze.fit, col=c("grey","black"), lty=1, lwd=2)
```

lty="dashed" instructs R to make the lines dashed. You can try different colours if you like.



Now we can run a survival test. There are a number of different possible tests. They shouldn't vary a lot in the final *P*-values, though, so we're going to start by applying one of the more straightforward tests.

```
survdiff(maze.T1.surv~SPECIES,data=maze)
```

RESULT

```
Call:
survdiff(formula = maze.T1.surv ~ SPECIES, data = maze)

                  N Observed Expected (O-E)^2/E (O-E)^2/V
SPECIES=agilis   30       27     16.3      6.96      11.4
SPECIES=flavipes 30       18     28.7      3.97      11.4

 Chisq= 11.4  on 1 degrees of freedom, p= 0.00072
```

This suggests that there is a difference between the species, but, we might be concerned that body size is having an effect here too. Perhaps larger antechinus explore more quickly or vice versa. Because we are adding a continuous variable, we need to use regression. Because we are using a parametric approach here, we need to build a number of different models and look for the best fit to various distributions:

```
surv.reg.W <- survreg(maze.T1.surv ~ SPECIES * MASS.g, data =
maze, dist = "weibull")

surv.reg.E1 <- survreg(maze.T1.surv ~ SPECIES * MASS.g, data =
maze, dist = "exponential")

surv.reg.E2 <- survreg(maze.T1.surv ~ SPECIES * MASS.g, data =
maze, dist = "extreme")

surv.reg.G <- survreg(maze.T1.surv ~ SPECIES * MASS.g, data =
maze, dist = "gaussian")

surv.reg.L <- survreg(maze.T1.surv ~ SPECIES * MASS.g, data =
maze, dist = "logistic")
```

We can use an **anova** function to compare models.

Pick the model with the lowest **-2*LL** (this is related to the notion of information criterion and is similar to how an AIC is used. We'll look at AICs in a subsequent section, but for now all you need to know is that the lower **-2*LL**, the better the model).

```
anova(surv.reg.W, surv.reg.E1, surv.reg.E2, surv.reg.G,
surv.reg.L)
```

RESULT

```
> anova(surv.reg.W, surv.reg.E1, surv.reg.E2, surv.reg.G, surv.reg.L)
             Terms Resid. Df    -2*LL Test Df     Deviance  Pr(>Chi)
1 SPECIES * MASS.g        55 386.9572      NA           NA        NA
2 SPECIES * MASS.g        56 386.9938    = -1  -0.03661785 0.8482452
3 SPECIES * MASS.g        55 430.7299    =  1 -43.73603636        NA
4 SPECIES * MASS.g        55 421.5351    =  0   9.19475192        NA
5 SPECIES * MASS.g        55 424.2975    =  0  -2.76243548        NA
```

Both the Weibull and exponential distributions appear to be similarly good fits. The Weibull does have a marginally lower AIC, so we'll pick it as the distribution to use (Weibull is a quite flexible form of distribution, and you'll tend to find it is often the best, or at least in the top couple options).

```
summary(surv.reg.W)
```

RESULT

```
> summary(surv.reg.W)

Call:
survreg(formula = maze.T1.surv ~ SPECIES * MASS.g, data = maze,
    dist = "weibull")
                          Value Std. Error      z     p
(Intercept)             -0.2431     2.4104 -0.101 0.920
SPECIESflavipes          3.1758     3.6509  0.870 0.384
MASS.g                   0.0996     0.0764  1.303 0.193
SPECIESflavipes:MASS.g -0.0696     0.1125 -0.618 0.536
Log(scale)             -0.0242     0.1258 -0.192 0.848

Scale= 0.976

Weibull distribution
Loglik(model)= -193.5   Loglik(intercept only)= -200.1
      Chisq= 13.32 on 3 degrees of freedom, p= 0.004
Number of Newton-Raphson Iterations: 5
n= 60
```

The interaction is not significant, so it can be removed. Re-run with model without the interaction term.

```
surv.reg.W<- survreg(maze.T1.surv ~ SPECIES + MASS.g, data =
maze, dist = "weibull")
summary(surv.reg.W)
```

RESULT

```
> summary(surv.reg.W)

Call:
survreg(formula = maze.T1.surv ~ SPECIES + MASS.g, data = maze,
    dist = "weibull")
                 Value Std. Error      z       p
(Intercept)      0.7019     1.8351  0.382 0.70210
SPECIESflavipes  0.9302     0.3139  2.964 0.00304
MASS.g           0.0696     0.0577  1.206 0.22796
Log(scale)      -0.0216     0.1255 -0.172 0.86356

Scale= 0.979

Weibull distribution
Loglik(model)= -193.7   Loglik(intercept only)= -200.1
      Chisq= 12.95 on 2 degrees of freedom, p= 0.0015
Number of Newton-Raphson Iterations: 5
n= 60
```

Species is having a significant effect, but mass is not. The ‘value’ and ‘Std.Error’ is the effect size ± standard error. The final result appears to be that flavipes take significantly longer to solve a maze than do agilis.

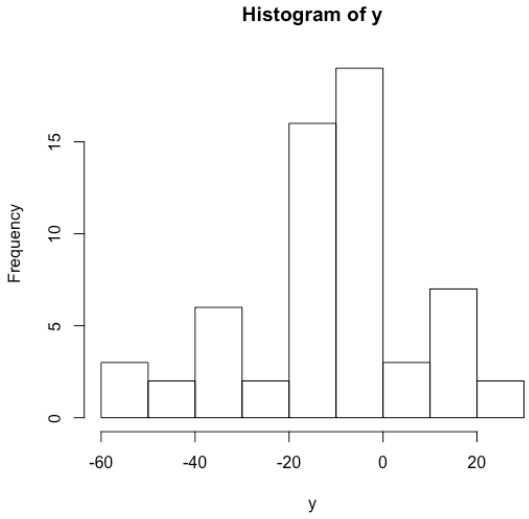
At this point it would also be senibsle to check the basic assumptions of the model. There is no straightforward way to generate a standard set of diagnostic plots from a survival regression, so we'll simply extract the residuals and use them to check for normality and equal variances. If the model is not following the specified distribution, we would expect to see a non-normal distribution of residuals.

First, extract the residuals of the model and drop them into an object called y.

```
y <- resid(surv.reg.W)
```

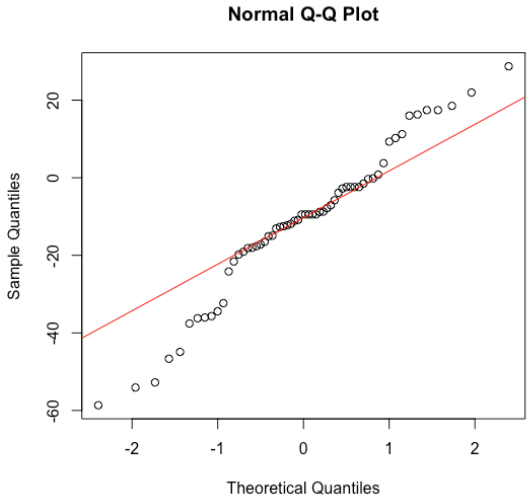
Now, check the histogram of the residuals.

```
hist(y)
```



Now try a qqplot:

```
qqnorm(y)
qqline(y, probs = c(0.25, 0.75), col = "red")
```



The distribution looks a little odd, but let's try a Shapiro-Wilk's test as well.

```
shapiro.test(y)
```

RESULT

```
> shapiro.test(y)

      Shapiro-Wilk normality test

data:  y
W = 0.96544, p-value = 0.08706
```

Given that Shapiro-Wilk's tests are quite aggressive, a P value of 0.08 is probably acceptable. We'll continue with the parametric analysis of this trial, but if you were to obtain a Shapiro-Wilk's result of P < 0.01 it is probably worth also checking the non-parametric Cox's option as well. If the results differ substantially, the non-parametric Cox's probably hazard test is likely the better option.

We can also have a go at generating something akin to a residuals vs fitted plot, but we have to extract the fitted values from a dummy linear model manually. First, drop the residuals into the dataset.

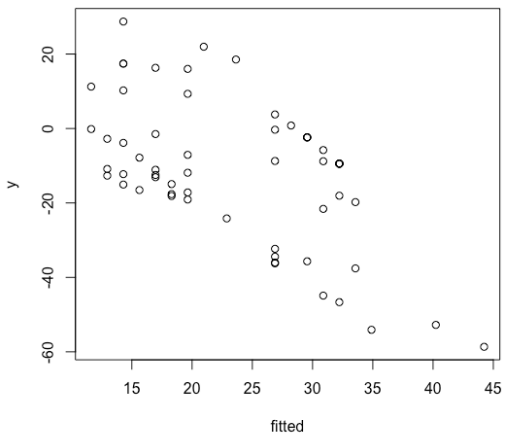
```
maze$y <- resid(surv.reg.W)
```

Create a dummy linear model to obtain fitted values. Drop these into the dataset too.

```
fit.lm<-lm(T1.time ~ SPECIES + MASS.g,data=maze)
maze$fitted<-fitted.values(fit.lm)
```

Plot the residuals against the fitted values.

```
plot(y~fitted,data=maze)
```



As with standard residuals versus fitted plots, we would be concerned if there was a clear 'wedge' or 'arrowhead' shape in the cloud of data points. That doesn't seem to be the case here, so at this point we will move along with the analysis.

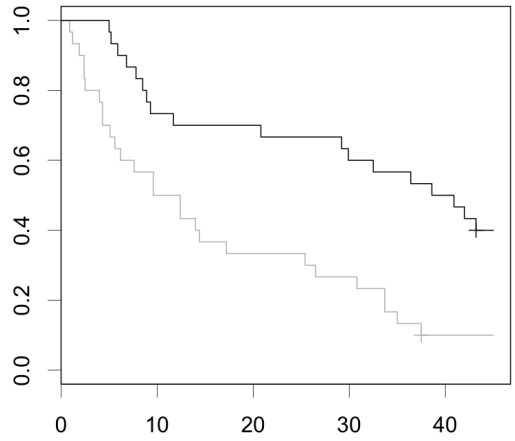


Figure X. Survival analysis plot for Maze 1. Grey: *A. agilis*. Black: *A. flavipes*.

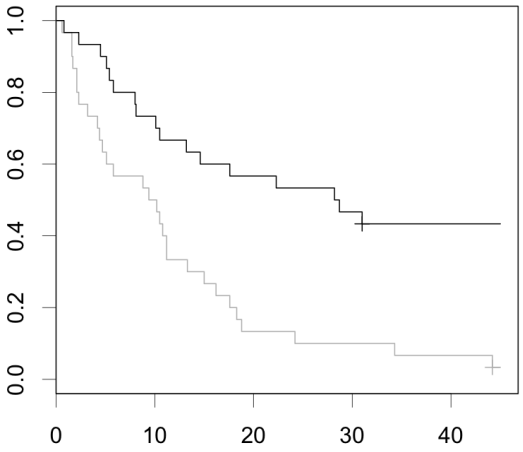


Figure X. Survival analysis plot for Maze 2. Grey: *A. agilis*. Black: *A. flavipes*.

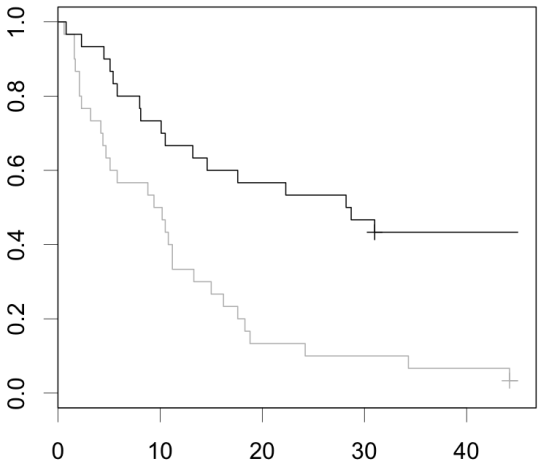


Figure X. Survival analysis plot for Maze 3. Grey: *A. agilis*. Black: *A. flavipes*.

Non-parametric Cox Probable Hazards Survival Analysis

Another option is to try a non-parametric Cox's Probable Hazard (Cox PH) survival analysis. Like all non-parametric tests, Cox's PH will (very slightly) inflate your Type II error, which is probably the reason it tends to be avoided in biological sciences. However, it is a very popular choice in medical studies, and will tend to give a (more or less) similar result to a properly fitted parametric test.

```
maze.T1.surv <- with(maze, Surv(T1.time,T1.goal))

coxph(maze.T1.surv ~ SPECIES * MASS.g, data = maze)
```

RESULT

```
Call:
coxph(formula = maze.T1.surv ~ SPECIES * MASS.g, data = maze)

                          coef exp(coef) se(coef)     z    p
SPECIESflavipes        -3.6018    0.0273   3.7370 -0.96 0.34
MASS.g                 -0.1075    0.8981   0.0797 -1.35 0.18
SPECIESflavipes:MASS.g  0.0819    1.0853   0.1151  0.71 0.48

Likelihood ratio test=13  on 3 df, p=0.00458
n= 60, number of events= 45
```

Remove the non-significant interaction term.

```
coxph(maze.T1.surv ~ SPECIES + MASS.g, data = maze)
```

RESULT

```
Call:
coxph(formula = maze.T1.surv ~ SPECIES + MASS.g, data = maze)

                   coef exp(coef) se(coef)     z      p
SPECIESflavipes -0.9550    0.3848   0.3182 -3.00 0.0027
MASS.g          -0.0710    0.9315   0.0596 -1.19 0.2337

Likelihood ratio test=12.5  on 2 df, p=0.0019
n= 60, number of events= 45
```

Mixed Effects Cox Probable Hazards Survival Analysis

If you need to include a random effect, there is an option in the 'coxme' package. There is no parametric mixed effects model survival analysis (so far as I am aware) in R, so where random effects are needed to control for pseudoreplication, you will need to use the non-parametric option (although, this is only non-parametric in a sense: the model assumes a Gaussian distribution for the random effect).

The below code is similar to what we have been running above, except that we have included a random effect 'T1.time.of.day' because some trials were run in the evening and other trials were in the afternoon, and we are concerned this might affect behaviour as antechinus are nocturnal.

```
maze <- read.table('survival.csv',header=T, sep = ',')

fit.coxme <- coxme(Surv(T1.time,T1.goal) ~ SPECIES + MASS.g +
(1|T1.time.of.day), data=maze)

fit.coxme # no need to use 'summary'
```

RESULT

```
Cox mixed-effects model fit by maximum likelihood
  Data: maze
  events, n = 45, 60
  Iterations= 5 28
                    NULL Integrated    Fitted
Log-likelihood -160.7289  -154.1803 -153.4261

                  Chisq  df         p  AIC  BIC
Integrated loglik 13.10 3.0 0.0044309 7.10 1.68
 Penalized loglik 14.61 2.6 0.0014141 9.41 4.71

Model:    Surv(T1.time,  T1.goal)  ~  SPECIES  +  MASS.g  +  (1  |
T1.time.of.day)
Fixed coefficients
                        coef exp(coef)   se(coef)     z      p
SPECIESflavipes -0.97516716 0.3771293 0.31923006 -3.05 0.0023
MASS.g          -0.07118842 0.9312864 0.05916841 -1.20 0.2300

Random effects
 Group          Variable  Std Dev    Variance
 T1.time.of.day Intercept 0.26328661 0.06931984
```

Repeatabilities

Repeatabilities are used to determine whether a sequence of measurements or values are showing a tendency towards similarity or 'repeatableness' within groups. One of the classic areas in which repeatability is used is when determining whether bird nestlings within nests are showing repeatability for a given trait, such as mass, haemoglobin, parasite load or similar. Another field where repeatability might be used would be to run animals through several behavioural tests, maybe days or weeks apart, to see whether there is repeatability at the level of the individual animal in terms of its response (i.e. lizard response to a novel object).

In this example, however, we will be using nestlings, as we have a nestling dataset available.

Import the dataset **swallow-nestlings-blood.csv** (remembering to change your working directory if needed). This is an actual dataset from an honours project.

```
nestlings <- read.table('swallows-nestlings-blood.csv',
header=T,sep=',')
```

Check the data:

```
head(nestlings)
str(nestlings)
```

There may be missing data because this is a real dataset and sometimes birds escape before they are fully measured. A quick but drastic way to remove all lines that have missing data uses this code:

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

We'll use a repeatability library to calculate repeatabilities.

```
install.packages("rptR")
library(rptR)
```

Repeatability: Normal distribution of response

The rptR package allows for determining repeatability of Gaussian (normal), Binary, Proportion and Poisson distributed response data. The following code will generate a bootstrapped repeatability for a Gaussian distributed variable. Here, we will check whether Hct is repeatable by nest.

```
rpt(Hct ~ (1|NEST.ID), grname="NEST.ID", datatype="Gaussian",
data=nestlings)
```

RESULT

```
Repeatability for NEST.ID
R  = 0.408
SE = 0.043
CI = [0.318, 0.489]
P  = 3.35e-19 [LRT]
     NA [Permutation]
```

Not that the datatype options are **"Gaussian"**, **"Binary"**, **"Proportion"** and **"count"** (i.e. **not** binomial or poisson, and count is with a lower case 'c').

A repeatability can be read something like a correlation coefficient, with high values indicating a strong degree of repeatability. The repeatability for Hct by nest is 0.408 and it is significantly different to a repeatability of zero ($P < 0.001$). Note that the 95% confidence interval runs from 0.318 to 0.489. Because it does not include zero, we can state with a 95% level of confidence that the repeatability is not zero.

Adjusting for uneven sample size

When the number of samples per group is uneven some authors like to apply a penalty to the repeatability value to take into account that unbalanced sample sizes introduce additional uncertainty into the data. This is called a repeatability adjustedment (n_0). This quote from Nakagawa & Schielzeth (*Repeatability for Gaussian and non-Gaussian data: a practical guide for biologists*; 2010) explains the underlying reasoning:

> The correction term n_0 is equal to the sample size per individual if sample sizes are equal for all groups, but n_0 is lower than the average sample size if sample sizes vary among individuals. This downward correction accounts for the overestimation of variance among smaller groups compared to larger groups that is a characteristic of least-squares estimation such as ANOVA

There is no library that compute the n_0, so we have to calculate it using some coding:

```
# compute n0, the repeatability adjustment

n <- as.data.frame(table(nestlings$NEST.ID))
k <- nrow(n)
N <- sum(n$Freq)
n0 <- (N-(sum(n$Freq^2)/N))/(k-1)
n0
```

RESULT

```
[1] 3.29088
```

The next step is to use the n0 to adjust the repeatability measurement we obtained above.

```
# compute the adjusted repeatability
R <- 0.408
Rn <- R/(R+(1-R)/n0)
Rn
```

RESULT

```
[1] 0.6940061
```

The adjustment has actually increased the repeatability. We can take this to indicate that there is no particular problem with uneven sample sizes in this dataset. If the repeatability was substantially decreased, we might wish to interpret the result with some caution. Note that if you do have even sample sizes (i.e. the same number of nestlings in all nests) then this final adjustment step is not necessary.

Repeatability: Others distribution of response

In order to check repeatability of other types of distribution, we would simply change the distribution parameter:

Poisson (counts)

```
rpt(RESPONSE ~ (1|GROUP), grname = "GROUP",
datatype="count",data=nestlings)
```

Binomial (binary)

```
rpt(RESPONSE ~ (1|GROUP), grname = "GROUP",
datatype="Binary",data=nestlings)
```

Proportion (percentages)

```
rpt(RESPONSE ~ (1|GROUP), grname = "GROUP",
datatype="Proportion",data=nestlings)
```

The **rptR** package actually allows for some quite sophisticated models to be tests for repeatability. Use the **?rpt** command and scroll down to the examples provided to get a feel for the variety and types of models that can be accommodated.

Species Accumulation Curves

The concept behind a species accumulation curve (SAC) is actually quite straightforward. Imagine you were hired to go into a forest and count all the species of frogs. Each day you go out, looking for more frog species. To start with you would get one, two or three new species each day, but over time you'd find fewer and fewer new species. Eventually you'd be looking for a few days, or a week or a month and not find any new species. At this point, we can probably assume you've found all or most of the frog species in the forest. This doesn't necessarily mean you've found all the species. Some might be cryptic and hard to see. Some might be inactive at the time of year you are looking. However, broadly speaking we can build a curve from the number of new species you find each day, and check where it levels out to get an estimate of how many species are probably present.

Assumptions & Data Set-up

Species accumulation curves do not have any testable assumptions per se, but they can provide meaningless or misleading results if the experimental design is faulty. The most common reason for faulty design is sampling across a clear, heterogeneous boundary. What might cause such a boundary? Imagine you are identifying frog species, and you are collecting data over several weeks. If it was dry for most of the time, and then it rains for three days, then returns to dry weather, you could easily find that there is a clear difference between frogs that you identified during the dry and wet weather. The same could happen if you cross a geographic boundary. If you are running transects through forest, then cross into a swamp, the species assortment would probably suddenly jump.

Mostly, avoiding problems of heterogeneity is a matter of good design and thinking carefully about field sites. When you come to plotting the data, a heterogeneous boundary will be obvious as a 'jump' in the data. If the curve is smooth, and then jumps or kinks upwards you may have a heterogeneity problem. How to cope with this is trickier. To some degree, methods like the random bootstrapping approach smooth out these kinks and assume the whole environment is homogeneous, and is some instances this may actually be acceptable. Otherwise, it may be necessary to split the data (i.e. wet and dry days, forest and swamp transects).

Gradual changes across a sampling range are less problematic than sudden jumps, although keep in mind that by applying a species accumulation curve you are assuming that the environment does represent a coherent whole. It may be possible to run a transects 3km up a mountainside and see only gradual changes in species assortment, but it would still be questionable whether or not such a dataset might be better split into altitudinal regions. The clearest indicator that you are running collection through several different and gradually changing environments is that you will not see a levelling out of the curve. It will just continue to increase. Imagine if you ran a sample all the way across a continent. It's quite possible the curve would never really level out, because you are always encountering new species.

Measuring Effort

Species accumulation curves require some sort of index of effort. This could be quadrats along one or more transect lines, or days spent looking for frogs, or number of new species per individual counted (i.e. new frog species per frog).

In the example we'll look at the data was collected in Borneo, examining morphotypes of understory plants. The data was collected in six transects through the rainforest with nine quadrats per transect. The biologists were interesting in identifying whether there might be a change in species richness near the tourist boardwalks in Mulu Forest Park. To this end, they positioned Quadrat 1 (1x1m) immediately next to the boardwalk, and then ran a transect at a right angle into the forest, so that Quadrat 9 was always the furthest quadrat and always the same distance into the forest.

We will import quite a few datasets. The first dataset records presence and absence of a species in a given quadrat.

```
borneo <- read.table('borneo_binary.csv',header=T, sep = ',')
View(borneo)
```

This is not easily used for species accumulation curves, because the quadrat and transect will require extra coding to get around. Instead we'll just delete those columns and work with a matrix of the data.

```
borneo <- read.table('borneo_presabs.csv',header=T, sep = ',')
View(borneo)
```

We've also added the occurrences together and grouped these by quadrat:

```
borneo.out <- read.table(borneo_away_fr_boardwalk.csv',header=T,
sep = ',')
```

And split these by transect:

```
transect1 <- read.table('borneo_transect1.csv',header=T, sep = ',')
transect2 <- read.table('borneo_transect2.csv',header=T, sep = ',')
transect3 <- read.table('borneo_transect3.csv',header=T, sep = ',')
transect4 <- read.table('borneo_transect4.csv',header=T, sep = ',')
transect5 <- read.table('borneo_transect5.csv',header=T, sep = ',')
transect6 <- read.table('borneo_transect6.csv',header=T, sep = ',')
```

Finally, because we are interested in whether there may be a different pattern if we reverse the data (i.e. flip the transects and run them from the forest towards the boardwalk) we have an inverted dataset as well:

```
borneo.in <- read.table('borneo_towards_boardwalk.csv',header=T,
sep = ',')
```

Most of the code we will be using is from the vegan package, so you will need to load it now if you don't have it already installed and loaded.

```
install.packages("vegan")
library(vegan)
```

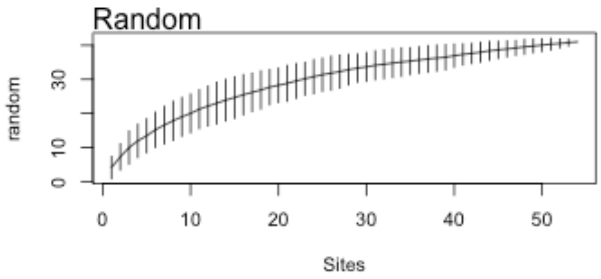
Let's start by applying some different methods for generating species accumulation curves. The method **collector** will add data in the order it was collected. The method **random** is a form of bootstrapping and adds data in a random order. The method **exact** finds the expected mean species richness and will only work if the data is in a presence/absence format (the other methods will accept accumulated species counts). The method **coleman** finds the expected species richness following Coleman et al. (1982) and **rarefaction** finds the mean when accumulating individuals instead of sites. If you data is already presented in the form of new species per individual, then you have already set it up as a rarefaction analysis by default.

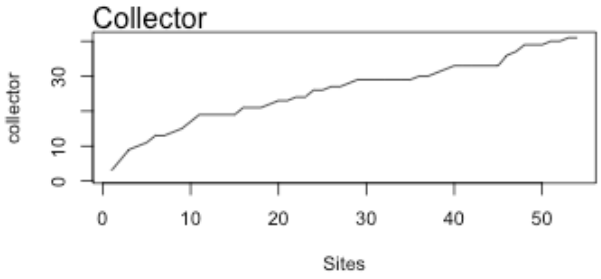
Let's work with the presence/absence data first.

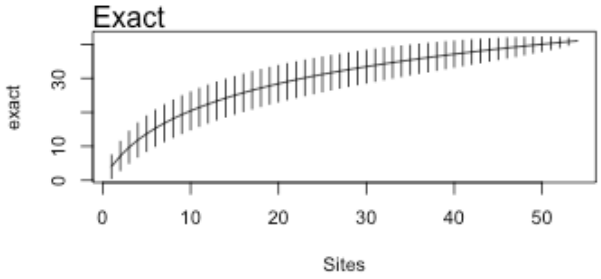
```
sp1 <- specaccum(borneo, "random")
sp2 <- specaccum(borneo, "collector")
sp3 <- specaccum(borneo, "exact")
sp4 <- specaccum(borneo, "coleman")
sp5 <- specaccum(borneo, "rarefaction")
```

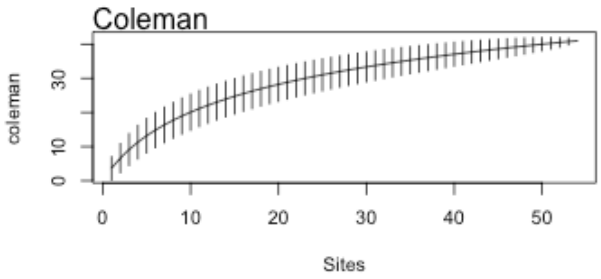
Graphing species accumulation curves

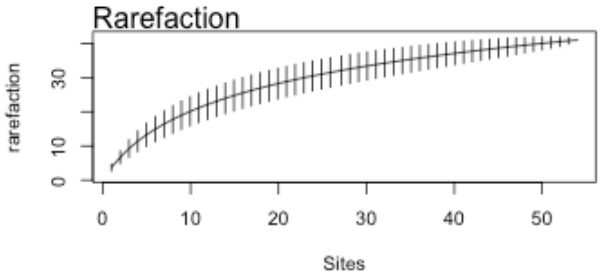
```
par(mfrow=c(5,1)) # Run this one line at a time

plot(sp1)
mtext("Random", side = 3, adj = 0, cex = 1, col = "black")

plot(sp2)
mtext("Collector", side = 3, adj = 0, cex = 1, col = "black")

plot(sp3)
mtext("Exact", side = 3, adj = 0, cex = 1, col = "black")

plot(sp4)
mtext("Coleman", side = 3, adj = 0, cex = 1, col = "black")

plot(sp5)
mtext("Rarefaction", side = 3, adj = 0, cex = 1, col = "black")
```


Random
random
0
10
30
0
10
20
30
40
50
Sites


Collector
collector
0
10
30
0
10
20
30
40
50
Sites


Exact
exact
0
10
30
0
10
20
30
40
50
Sites

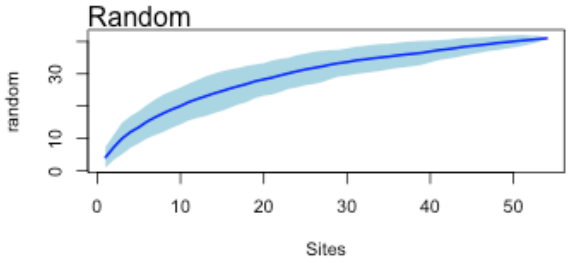

Coleman
coleman
0
10
30
0
10
20
30
40
50
Sites

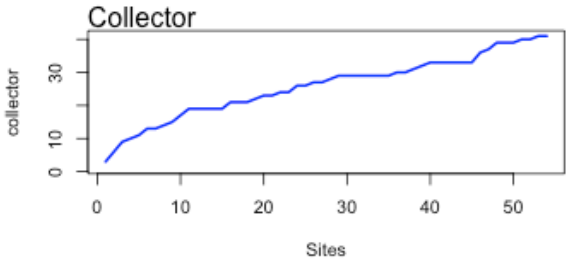

Rarefaction
rarefaction
0
10
30
0
10
20
30
40
50
Sites

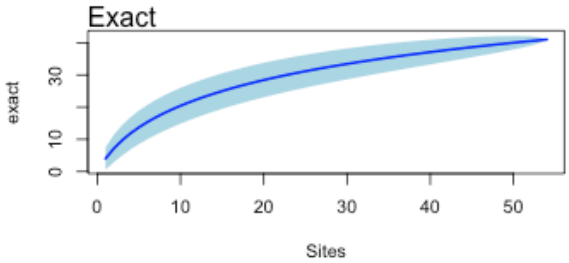
Graphing species accumulation curves: adding colour

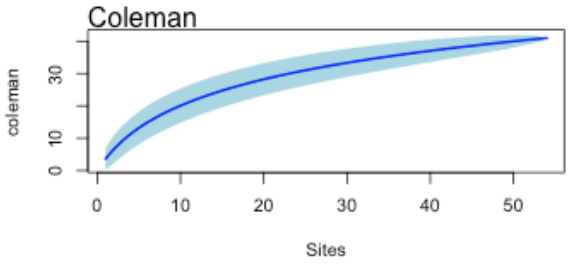
Let's try adding some colour and creating some nicer looking graphs.

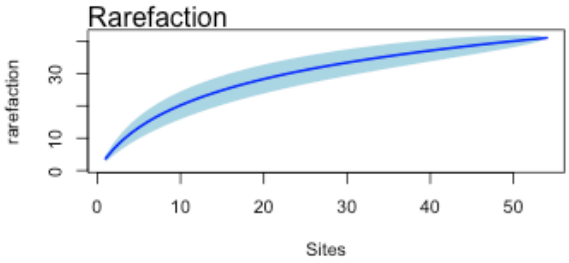
```
par(mfrow=c(5,1))

plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Random", side = 3, adj = 0, cex = 1, col = "black")

plot(sp2, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Collector", side = 3, adj = 0, cex = 1, col = "black")

plot(sp3, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Exact", side = 3, adj = 0, cex = 1, col = "black")

plot(sp4, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Coleman", side = 3, adj = 0, cex = 1, col = "black")

plot(sp5, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Rarefaction", side = 3, adj = 0, cex = 1, col =
"black")
```


Random
random
0
10
30
0
10
20
30
40
50
Sites


Collector
collector
0
10
30
0
10
20
30
40
50
Sites


Exact
exact
0
10
30
0
10
20
30
40
50
Sites


Coleman
coleman
0
10
30
0
10
20
30
40
50
Sites


Rarefaction
rarefaction
0
10
30
0
10
20
30
40
50
Sites

The species accumulation curves are looking quite nice, but we have a problem, which is that R thinks the whole 6 transects by 9 quadrats is a single collecting event, giving over 50 samples. We can sum the species observations together, and organise them by transects, but if we do this the method **`exact`** will no longer work. That's okay, as we will focus on using the methods **`random`** and **`collector`** from here on.

Let's start off by comparing transects.

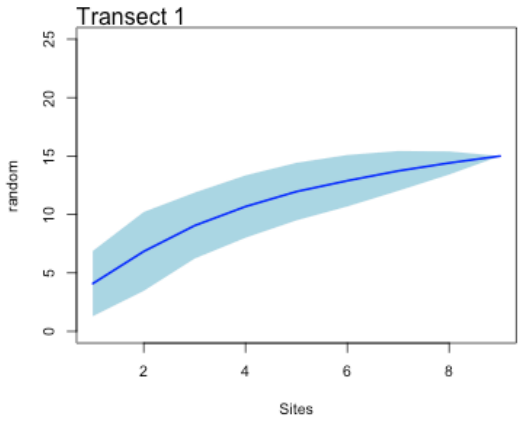
Using the random method (by transect)

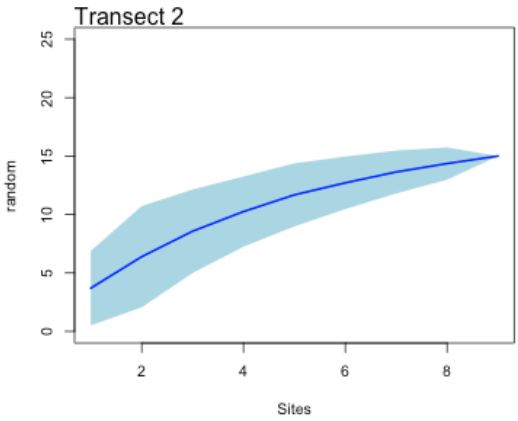
```
# RANDOM

par(mfrow=c(3,2))

sp1 <- specaccum(transect1, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 1", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect2, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 2", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect3, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 3", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect4, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 4", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect5, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 5", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect6, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 6", side = 3, adj = 0, cex = 1, col = "black")
```

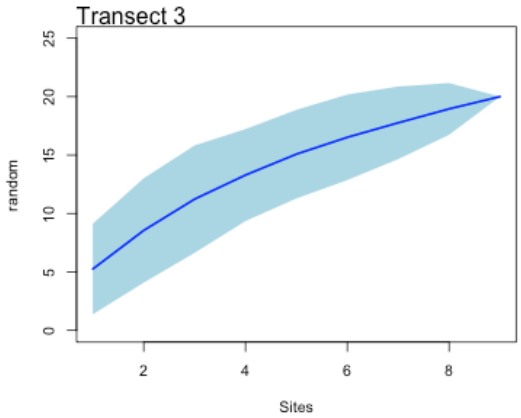
Using the collector method (by transect)

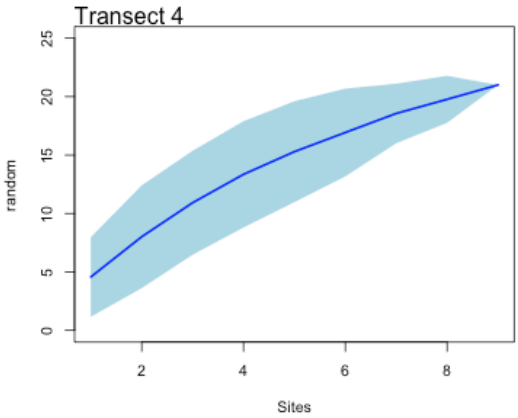
```
# COLLECTOR
par(mfrow=c(3,2))

sp1 <- specaccum(transect1, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 1", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect2, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 2", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect3, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 3", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect4, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 4", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect5, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 5", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect6, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 6", side = 3, adj = 0, cex = 1, col = "black")
```

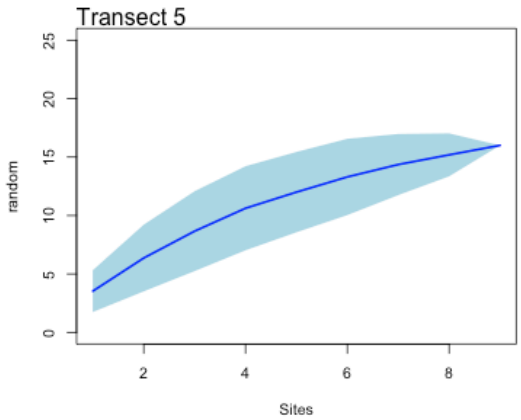
Using the random method:

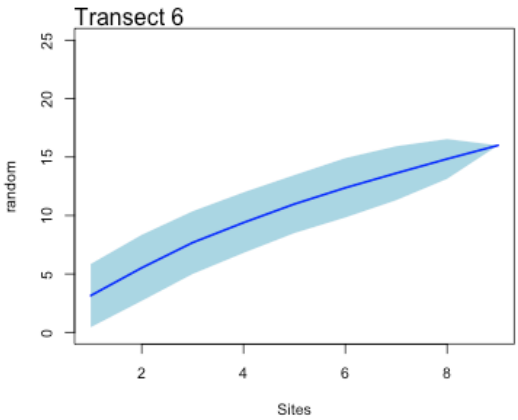




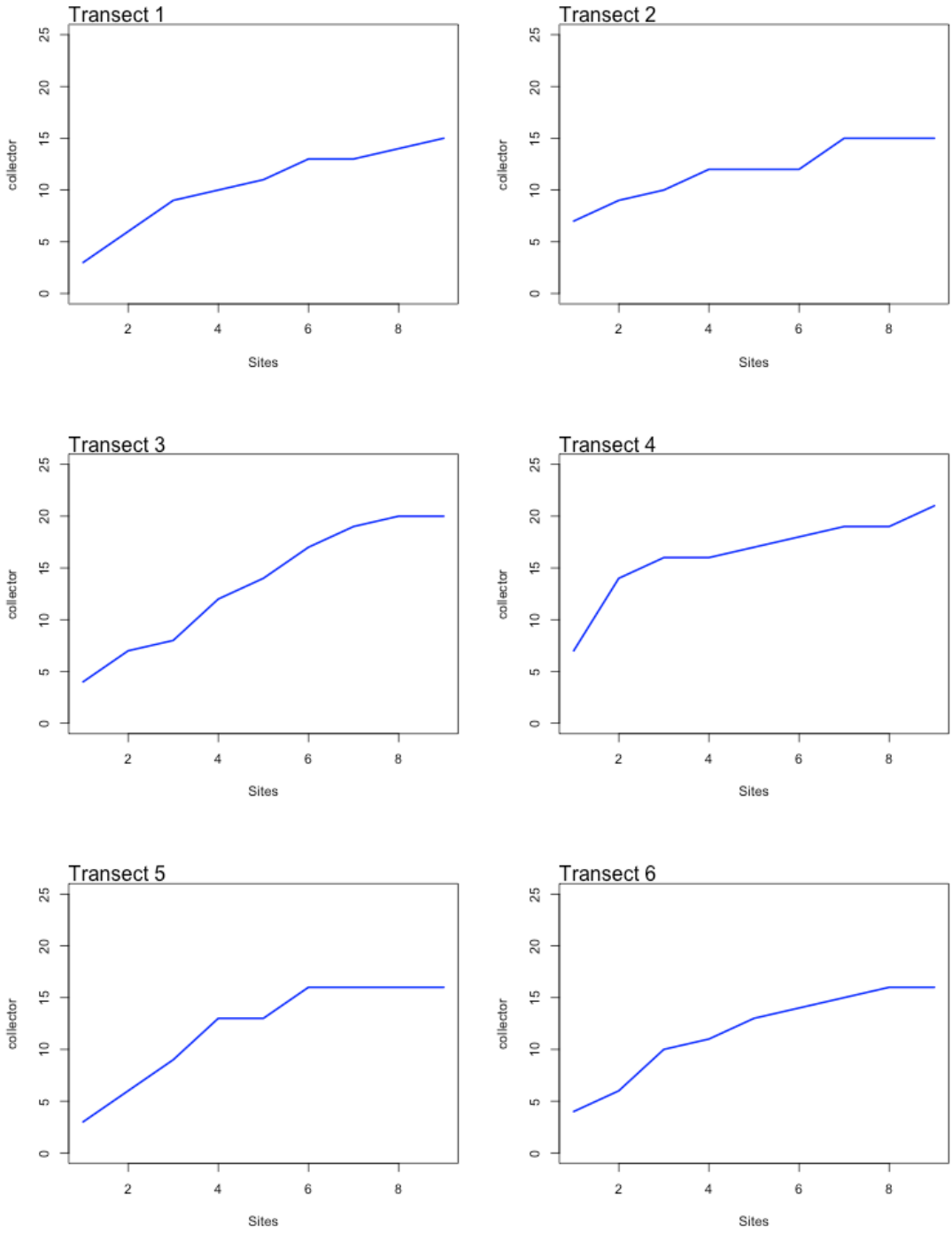








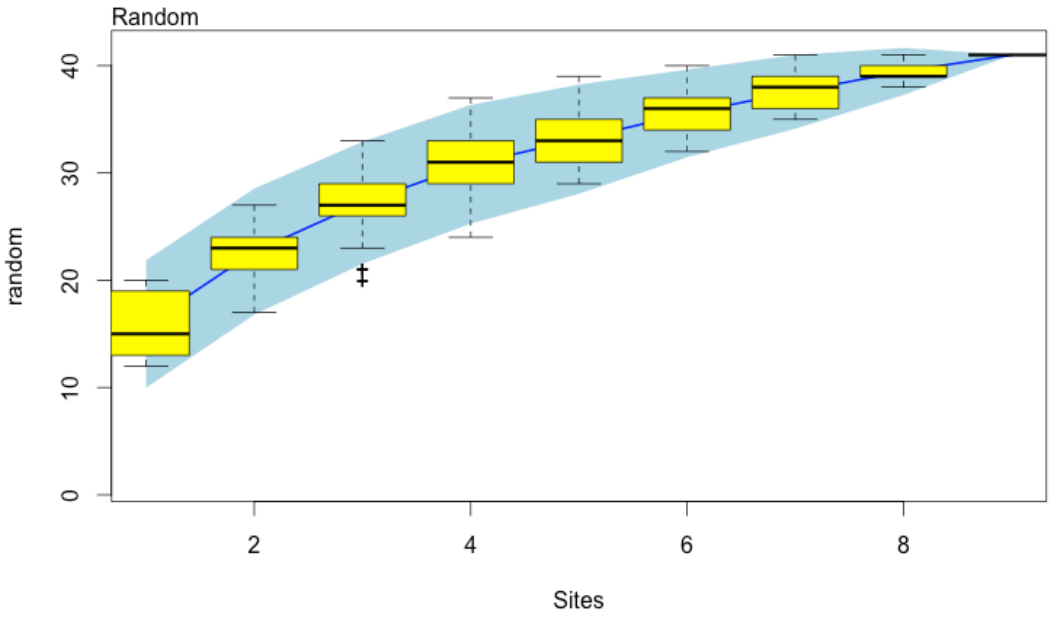
Using the collector method:

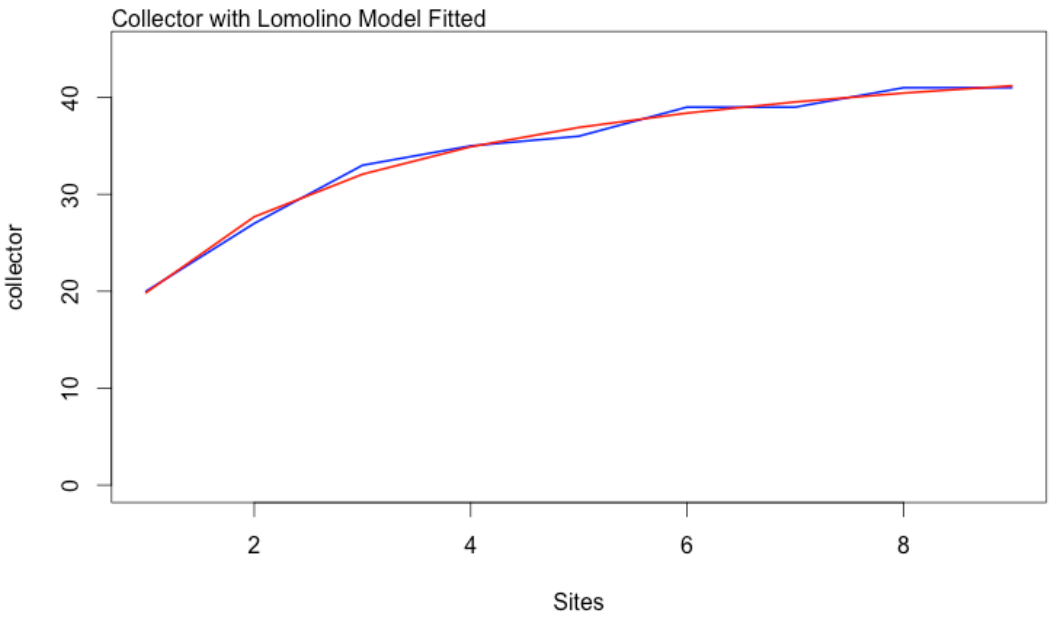


What we are primarily interested in here is whether there are any large jumps or kinks in the data. There are a couple places where there seem to be jumps, but there is nothing consistent, and given the coarse granularity of the transect (just nine quadrats) some jumpiness is to be expected.

Let's now look at all transects combined. We'll add some additional estimates to the graphs, focusing on the data as it was collected, moving from the boardwalk into the rainforest.

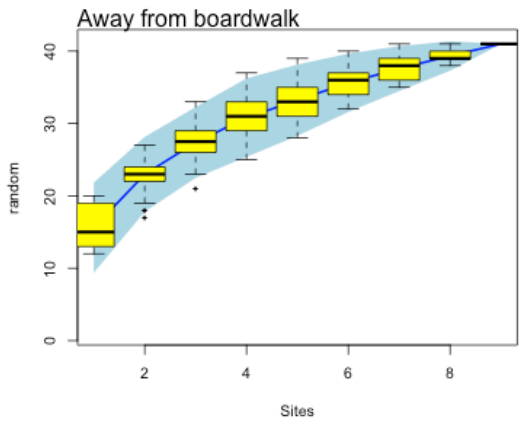
```
par(mfrow=c(2,1))

sp1 <- specaccum(borneo.out, "random")

plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")

boxplot(sp1, col="yellow", add=TRUE, pch="+")

mtext("Random", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(borneo.out, "collector")

plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,45))

mod1 <- fitspecaccum(sp1, "lomolino") # collector

plot(mod1, add = TRUE, col=2, lwd=2)

mtext("Collector with Lomolino Model Fitted", side = 3, adj = 0, cex
= 1, col = "black")
```

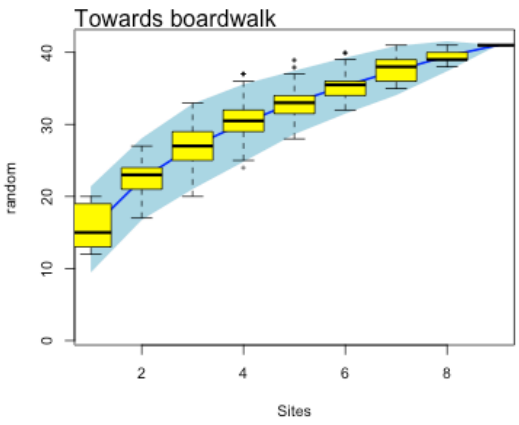


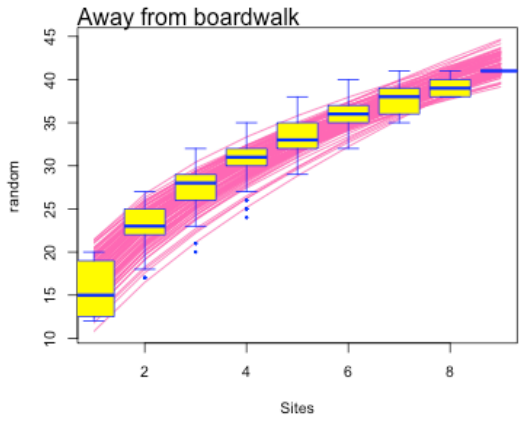


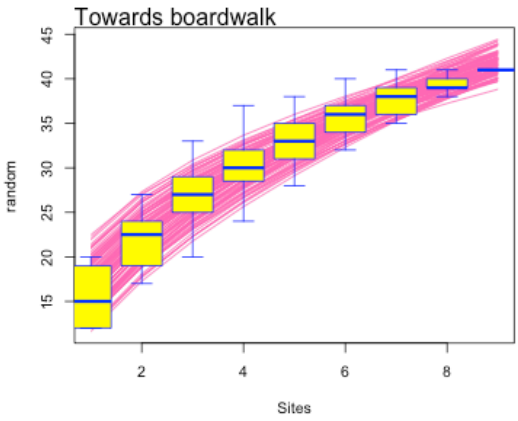
The original question was one of examining whether there is a difference in curves when the data is reversed (i.e. would moving towards the boardwalk produce a different curve to that generated by moving away from the boardwalk).

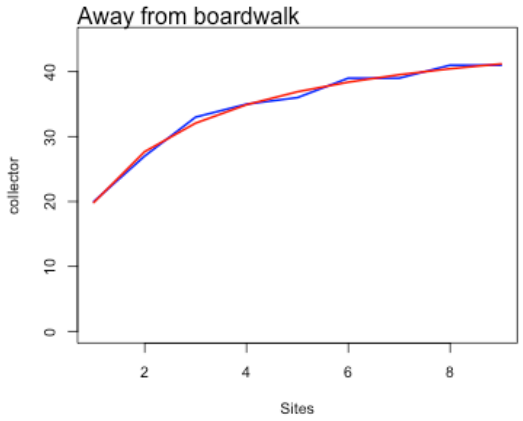
We'll produce use the random method to generate graphs showing the error around the curve as well as all the curves generated by bootstrapping. We'll append the collector curve to the end along with an estimated species curve based on a Lomolino model.

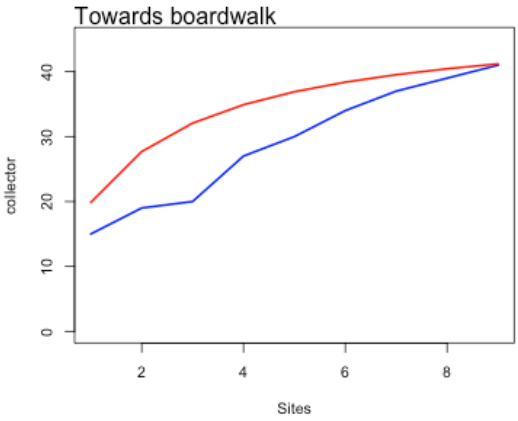












Examining the plots, what we can immediately see is the direction of collection makes no difference to the **random** models, and the Lomolino model fitted to the collector method is the same also. This shouldn't be surprising as the methods we are using are intended to overcome some of the problems of local heterogeneity in a generally uniform environment (no biological systems are perfectly uniform, so small scale or local heterogeneity has to be something that can be managed by these methods).

But, the actual collector curves are quite different. It looks very much like there might be something going on in terms of differences in species richness near and far away from the boardwalks.

The species accumulation curves won't allow us to easily examine this. Instead we'll need to take some different approaches.

Using the rarefy method

We can use rarefaction to identify how many species we would expect to obtain for a given number of plants counted. I'm going to use a basis of 20 plants counted (half of the 40 species), although you can play with values of 5, 10, 30 or more. Once you reach 40+ plants counted you should start to see the numbers of predicted species flattening out.

```
rar <- rarefy(borneo.out, 20)
rar
```

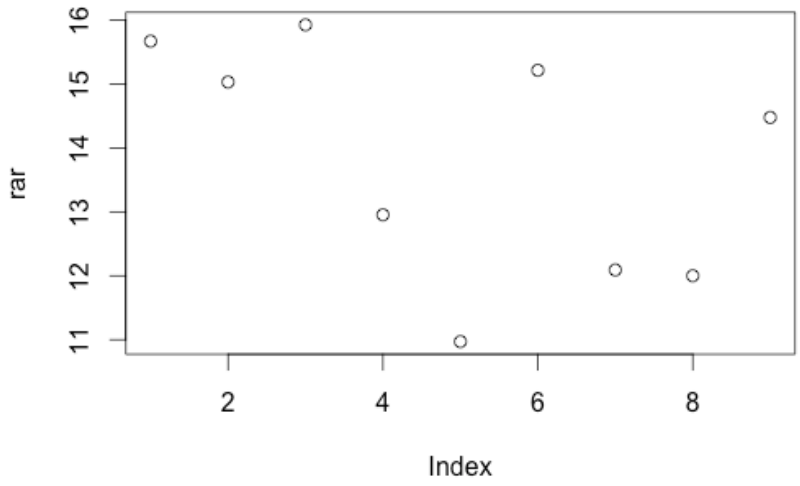
RESULT

```
[1] 15.66952 15.03077 15.92464 12.95323 10.97026 15.21481
12.09091 12.00000 14.47619
attr(,"Subsample")
```

You can interpret this to mean, that for all of the first quadrats combined (closest to boardwalk) the number of species you should expect to obtain for 20 plants counted is 15.7. For quadrat 2 you should expect 15.0 species per 20 plants etc.

You can plot this too:

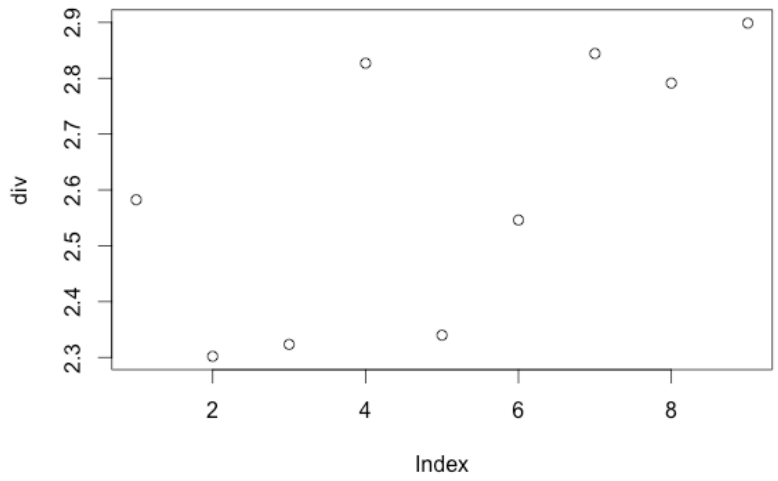
```
par(mfrow=c(1,1))
plot(rar)
```



Shannon's Diversity Index

There are some other indicators of interest too. We can use package vegan to obtain Shannon's Diversity index (a measure of entropy or degree of 'surprise' at finding a new species). Species richness is easily done in Excel just by summing rows, and I've already done this in the **`borneo.richness.csv`**.

```
div <- diversity(borneo.out)
div
plot(div)
```



Species Richness

Species richness is easily done in Excel just by summing rows, and I've already done this in the **borneo.richness.csv**.

```
borneo.summary <- read.table('borneo_richness.csv',header=T, sep = ',')
borneo.summary
```

RESULT

```
> borneo.summary
  QUADRAT AWAY.BOARDWALK TOWARDS.BOARDWALK
1       1             28                21
2       2             26                19
3       3             25                22
4       4             25                27
5       5             24                24
6       6             27                25
7       7             22                25
8       8             19                26
9       9             21                28
```

Let's now generate indices for each quadrat on the transects and add these back into the richness dataset.

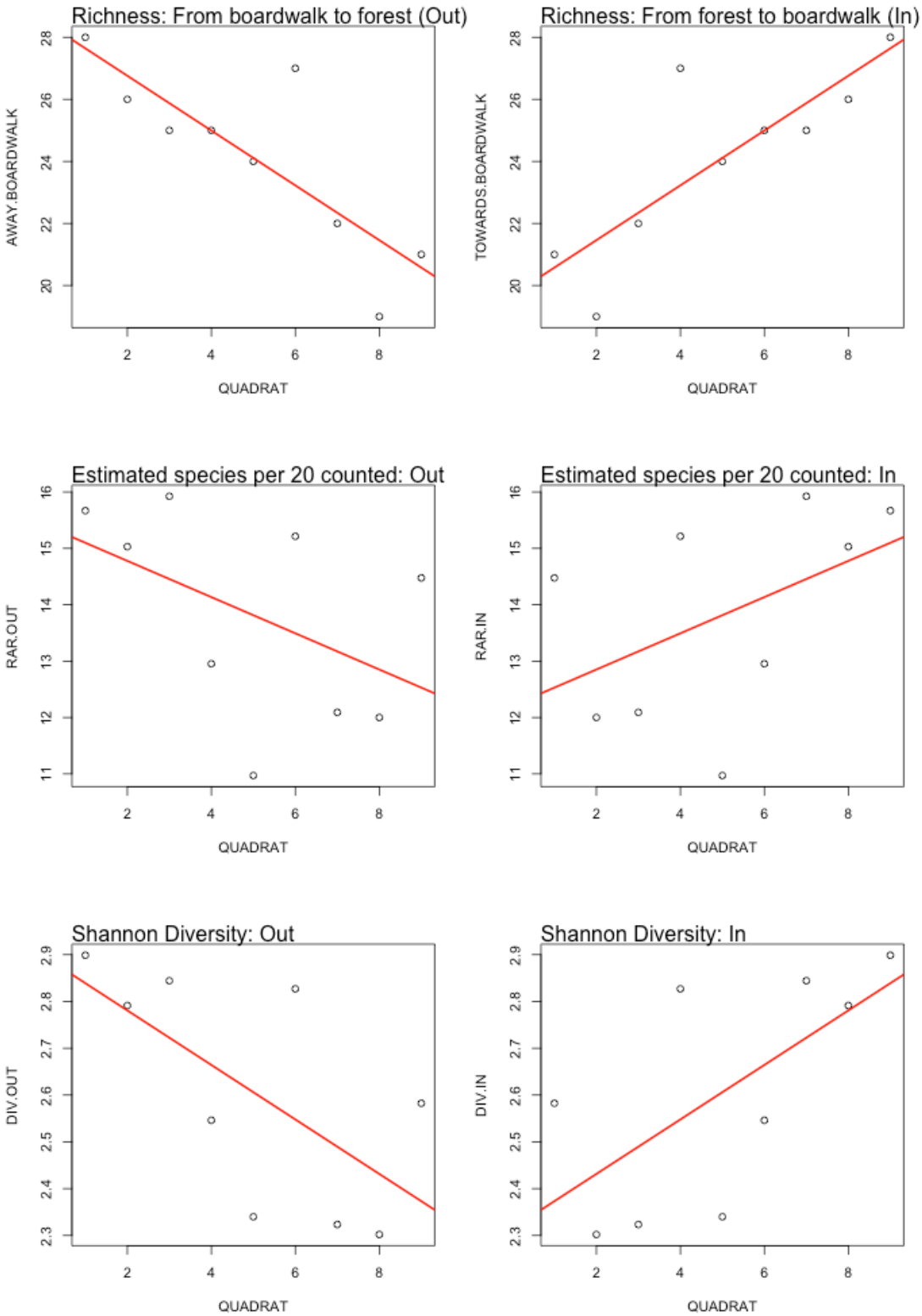
```
library(vegan)

rar <- rarefy(borneo.out, 20)
borneo.summary$RAR.OUT <- rar

rar <- rarefy(borneo.in, 20)
borneo.summary$RAR.IN <- rar

div <- diversity(borneo.out)
borneo.summary$DIV.OUT <- div

div <- diversity(borneo.in)
borneo.summary$DIV.IN <- div

borneo.summary
```

RESULT

```
> borneo.summary
  QUADRAT AWAY.BOARDWALK TOWARDS.BOARDWALK
1       1              28                 21
2       2              26                 19
3       3              25                 22
4       4              25                 27
5       5              24                 24
6       6              27                 25
7       7              22                 25
8       8              19                 26
9       9              21                 28

  RAR.OUT   RAR.IN  DIV.OUT   DIV.IN
1 15.66952 14.47619 2.898746 2.582424
2 15.03077 12.00000 2.791295 2.302014
3 15.92464 12.09091 2.844305 2.323397
4 12.95323 15.21481 2.546117 2.826944
5 10.97026 10.97026 2.339936 2.339936
6 15.21481 12.95323 2.826944 2.546117
7 12.09091 15.92464 2.323397 2.844305
8 12.00000 15.03077 2.302014 2.791295
9 14.47619 15.66952 2.582424 2.898746
```

And we can generate some figures to examine the trends. We should see an mirror opposite of figures here, because all we have done is flip the data.

```
par(mfrow=c(3,2))

plot(AWAY.BOARDWALK~QUADRAT,data=borneo.summary)
abline(lm(AWAY.BOARDWALK~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Richness: From boardwalk to forest (Out)", side = 3,
adj = 0, cex = 1, col = "black")

plot(TOWARDS.BOARDWALK~QUADRAT,data=borneo.summary)
abline(lm(TOWARDS.BOARDWALK~QUADRAT,data=borneo.summary),
lwd=2, col="red")
mtext("Richness: From forest to boardwalk (In)", side = 3, adj
= 0, cex = 1, col = "black")

plot(RAR.OUT~QUADRAT,data=borneo.summary)
abline(lm(RAR.OUT~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Estimated species per 20 counted: Out", side = 3, adj =
0, cex = 1, col = "black")

plot(RAR.IN~QUADRAT,data=borneo.summary)
abline(lm(RAR.IN~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Estimated species per 20 counted: In", side = 3, adj =
0, cex = 1, col = "black")

plot(DIV.OUT~QUADRAT,data=borneo.summary)
abline(lm(DIV.OUT~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Shannon Diversity: Out", side = 3, adj = 0, cex = 1,
col = "black")

plot(DIV.IN~QUADRAT,data=borneo.summary)
abline(lm(DIV.IN~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Shannon Diversity: In", side = 3, adj = 0, cex = 1, col
= "black")
```

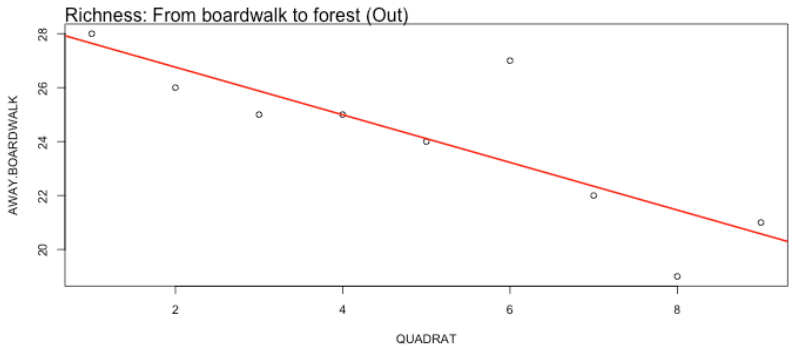


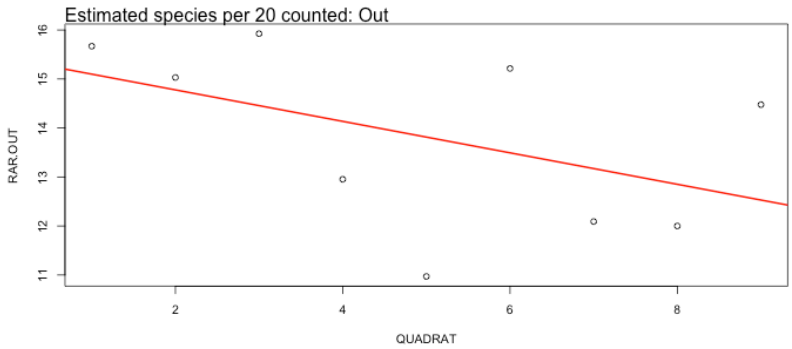
In a sense, the mirror reflection is a bit redundant. All we are really interested in is whether there is a trend across the transects. We can use straightforward regression analysis to obtain P-values also.

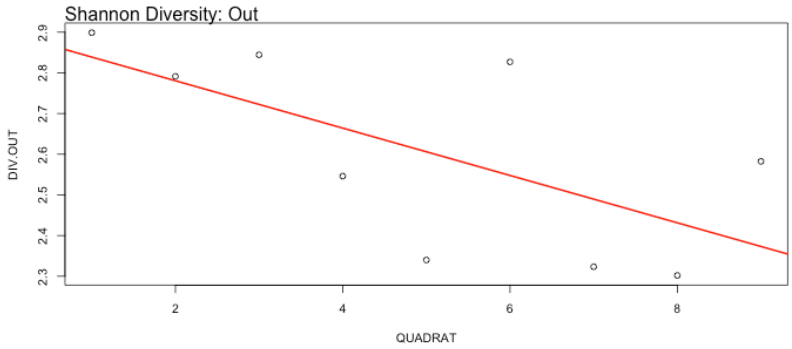
```
# FINAL GRAPHS
par(mfrow=c(3,1))

plot(AWAY.BOARDWALK~QUADRAT,data=borneo.summary)
abline(lm(AWAY.BOARDWALK~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Richness: From boardwalk to forest (Out)", side = 3,
adj = 0, cex = 1, col = "black")

plot(RAR.OUT~QUADRAT,data=borneo.summary)
abline(lm(RAR.OUT~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Estimated species per 20 counted: Out", side = 3, adj =
0, cex = 1, col = "black")

plot(DIV.OUT~QUADRAT,data=borneo.summary)
abline(lm(DIV.OUT~QUADRAT,data=borneo.summary), lwd=2,
col="red")
mtext("Shannon Diversity: Out", side = 3, adj = 0, cex = 1,
col = "black")
```

The following code can be used to run some straightforward regression analyses.

```
summary(lm(AWAY.BOARDWALK~QUADRAT,data=borneo.summary))
summary(lm(RAR.OUT~QUADRAT,data=borneo.summary))
summary(lm(DIV.OUT~QUADRAT,data=borneo.summary))
```


Richness: From boardwalk to forest (Out)
AWAY.BOARDWALK
QUADRAT


Estimated species per 20 counted: Out
RAR.OUT
QUADRAT


Shannon Diversity: Out
DIV.OUT
QUADRAT

Conditional inference trees

Conditional inference trees (CIFs) are a useful way to identify complex relationships in data, especially where there may be several interactions at work. CIFs have some advantages over other methods:

- Any type of data can be used for the response variable (no assumptions)
    - Although transformed response data will sometimes be easier to read
- Any type of data can be used for the explanatory variables (no assumptions)
- Any number of explanatory variables can be used (impossible to over-parameterise)

Conditional inference trees examine all possible splits in a response variable by all possible combinations of predictor variables and then nominates a split in the data that best maximises within group homogeneity and between group variance. The split is tested against a significance level, and if the split is non-significant, the CIF returns a non-significant result (a single boxplot). If the split satisfies the significance level (usually either $P < 0.1$ or $P < 0.05$) then the data is re-examined for further splitting and the process is repeated. This process is called binary recursive partitioning.

Here is the basic formula for a CIF:

```
library(party)

agilis.tree <- ctree(HABITAT~ PRED1 + PRED2 + PRED3,
data=agilis,control = ctree_control(mincriterion = 0.9))
plot(agilis.tree)
```

You can look at the model by writing:
`agilis.tree`

You can adjust the significance level by changing `mincriterion = 0.9`. What happens if you change the `mincriterion = 0.9` to `0.8`, **0.5** or `0.1`?

There is no limit to the number of predictors you can include (although not all of them will make it into the model as a significant effect). Using the table on the next page as a guide, have a go at running CIFs looking at various response variables (small mammal abundance) using various predictor variables.

If you find that one predictor variable (like **HABITAT**) is dominating the trees you are constructing, try removing it and re-running the trees to see what happens.

Also, what happens when you nominate **HABITAT** as the response variable?

VARIABLES YOU MAY WISH TO EXAMINE

sqrtAGILIS	Abundance of all agile antechinus
sqrtF	Abundance of all female antechinus
sqrtM	Abundance of all male antechinus
sqrtFUSCIPES	Abundance of bush rats
sqrtLUTREOLUS	Abundance of swamp rats
sqrtSWAINSONII	Abundance of dusky antechinus
sqrtMUSCULUS	Abundance of feral domestic mice
sqrtRATTUS	Abundance of feral black rats

VARIABLES YOU MAY WISH TO USE AS PREDICTORS

YEAR	Year in which the data was collected
HABITAT	Was the forest fragmented (F) or continuous (C)?
MONTH	What month was the data collected in (1 = Jan, 12 = Dec)
sqrtFOX	Index of fox activity at site
total.ABH	Total area at breast height of trees
ABH.m2	Mean area at breast height of trees
median.ABH	Median area at breast height of trees
TREE.SPECIES	Species richness of trees
STUMPS	Number of stumps in 20x20 quadrates (400m^2)
DOMINANCE	Dominance index applied to shrub species
EVENNESS.W	Wilson's evenness index applied to shrub species
median.DBH	Median diameter at breast height of trees
SHRUB.COUNT	Number of shrubs counted in 20x20 quadrates (400m^2)
SHRUB.SPECIES	Species richness of shrubs
BROWSE	Index of browsing pressure at site (higher = more browsing)
WOODY.DEBRIS	Number of logs counted in 20x20 quadrates (400m^2)
pcDEAD	Percentage of trees that were dead and standing
pcNONEUC	Percentage of trees that were not *Eucalyptus*
CANOPY	Index of canopy. 0 = none. 5 = heavy.
MIDSTOREY	Index of midstorey. 0 = none. 5 = heavy.
UNDERSTOREY	Index of understorey. 0 = none. 5 = heavy.
GROUNDCOVER	Index of groundcover. 0 = none. 5 = heavy.
LEAF.LITTER	Index of leaf litter. 0 = none. 5 = heavy.
BRACKEN	Was bracken 0 absent, 1, present, 2, dominant?
SLOPE	Was there a slope at the site? 1=Y 0 =N
RIDGE	Was there a ridge at the site? 1=Y 0 =N
GULLY	Was there a gully at the site? 1=Y 0 =N
ALT.m	Altitude of site above sea level
LAT.DEC	Decimal latitude of site
LONG.DEC	Decimal longitude of site

Example of a conditional inference tree (with interpretation)

```
agilis.tree <- ctree(sqrtAGILIS~ BRACKEN + LEAF.LITTER,
data=agilis,control = ctree_control(mincriterion = 0.9))

agilis.tree
```

RESULT

```
       Conditional inference tree with 3 terminal nodes

Response:  sqrtAGILIS
Inputs:  BRACKEN, LEAF.LITTER
Number of observations:  120

1) BRACKEN <= 0.5; criterion = 1, statistic = 18.693
  2)*  weights = 24
1) BRACKEN > 0.5
  3) LEAF.LITTER <= 3.5; criterion = 0.998, statistic = 10.736
    4)*  weights = 72
  3) LEAF.LITTER > 3.5
    5)*  weights = 24
```

```
plot(agilis.tree)
```

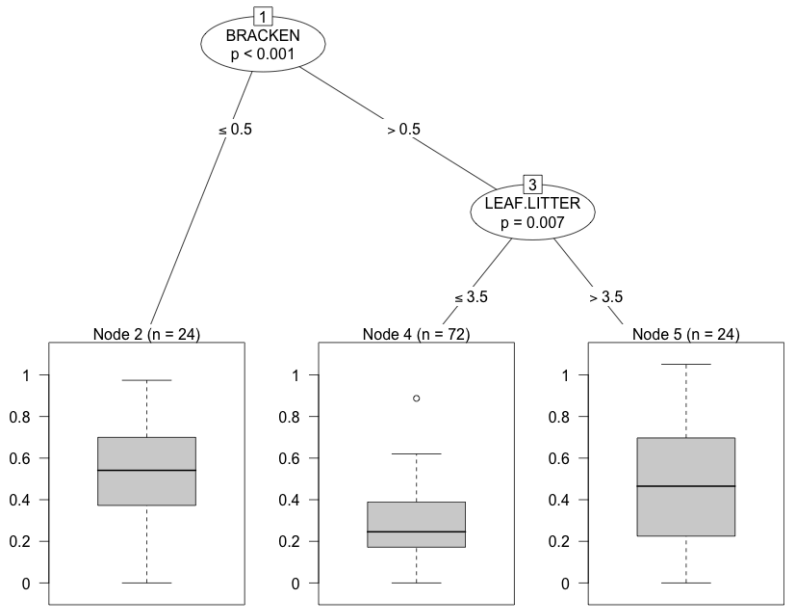


Figure X. Conditional inference tree of agile antechinus total abundance as a function of Bracken and Leaf Litter. In sites with little bracken (index < 0.5) agile antechinus abundances were higher (Node 2, n = 24). In sites where bracken was more dominant (index > 0.5) leaf litter influences agile antechinus abundances. Sites with higher bracken indices and less leaf litter had the lowest agile antechinus abundances (Node 4, n = 72). In sites with high bracken indices but also high leaf litter indices, agile antechinus abundance was higher (Node 5, n = 24).

Random Forests

Random Forests is a model averaging technique based on Conditional Inference trees (above). A Random Forests analysis bootstaps a random set of variables from your set of predictors and tries to build a Conditional Inference Tree using this subset. It then saves the tree, bootstaps another set of variables, and tries to build a second tree, a third and so on. As a default, we usually aim for about 10,000 trees (which can take a while to run depending on the data and your computer). The actual test output is straightforward and elegant. Once all of the trees have been generated, the test counts up the number of times any given variable was used to construct a tree. If a variable has been used more frequently it is probably more important for predicting the response variable. This allows the test to work out a set of relative importances for the predictors involved.

Some random forests basics:

- Random forests analysis requires one response and any number of predictors
- You can't over-fit or over-parametize the model-selection process
- Correlation of predictors is not a concern
- Will accept any type(s) of data including categorical, binomial, continuous etc

Load libraries

```
library(party)
```

Load data

```
agilis <- read.table('agilis-abundance.csv', header=T,sep=',')

agilis <-na.omit(agilis)

str(agilis)
```

Decide on your response and predictors.

Response: **sqrtAGILIS**

Predictors **ABH.m2**
SHRUB.COUNT
WOODY.DEBRIS
LEAF.LITTER
pcNONEUC
ALT.m

Set some controls for the Random Forests process:

```
data.controls <- cforest_control(ntree=10000, mtry=3, replace = FALSE)
```

The above controls are instructing R how many trees to build (**10000**), how many variables to select for each tree (**3**) and whether to replace variables back into the 'bag' after they have been picked (i.e. can a variable be picked twice so that the random assortment is smaller than it would otherwise be?) (**FALSE** = No replacement).

Standard advice is to set the **mtry** to the square root of the number of predictors. We have six predictors, which gives a square root of 2.49, which we will round up to 3. The default is 5.

Set a seed. Any random number will do:

```
set.seed(42)
```

Now, build a model using the following code:

```
fit.cforest <- cforest(sqrtAGILIS ~ ABH.m2 + SHRUB.COUNT + WOODY.DEBRIS + LEAF.LITTER + pcNONEUC + ALT.m, controls = data.controls, data=agilis)
```

The order of predictor variables doesn't matter.

Now generate the relative importances for the variables. Using **`conditional = TRUE`** is less biased than the default (leaving out this command) but is also computationally intensive. Have a go at running the following code, but if it takes more than about 5min cancel the operation by clicking on the red stop sign in R Studio, and re-run the code without the **`conditional = TRUE`** operator.

With **`conditional = TRUE`**

```
fit.varimp <- varimp(fit.cforest, conditional = TRUE)
```

Without **`conditional = TRUE`**

```
fit.varimp <- varimp(fit.cforest)
```

If you were preparing data for a scientific paper or a presentation it would be appropriate to run the data using the **`conditional = TRUE`** operator. This may require you to run the test overnight or get access to a powerful computer.

On the assumption that you probably don't want to wait 15 to 30 min to test this code, I've continued using the test **without** the **`conditional = TRUE`** operator.

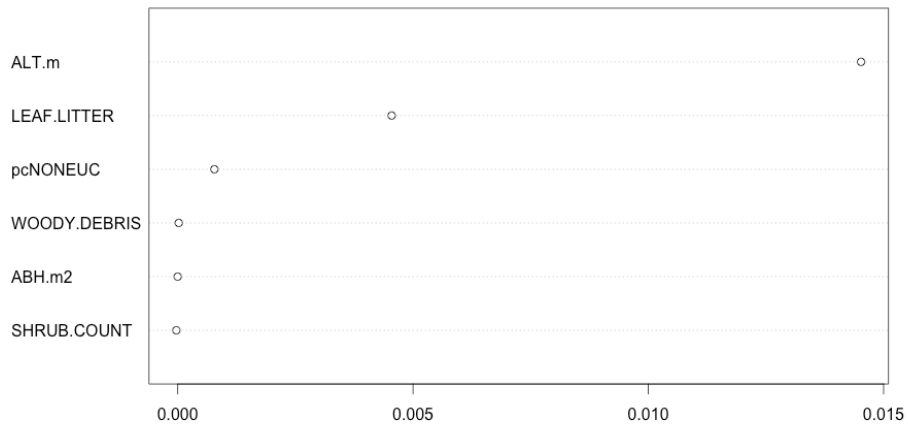
You can look at the relative importances as a set of numbers, but these numbers have no units and on their own they are not very meaningful. It is more sensible to present them graphically:

```
fit.varimp
```

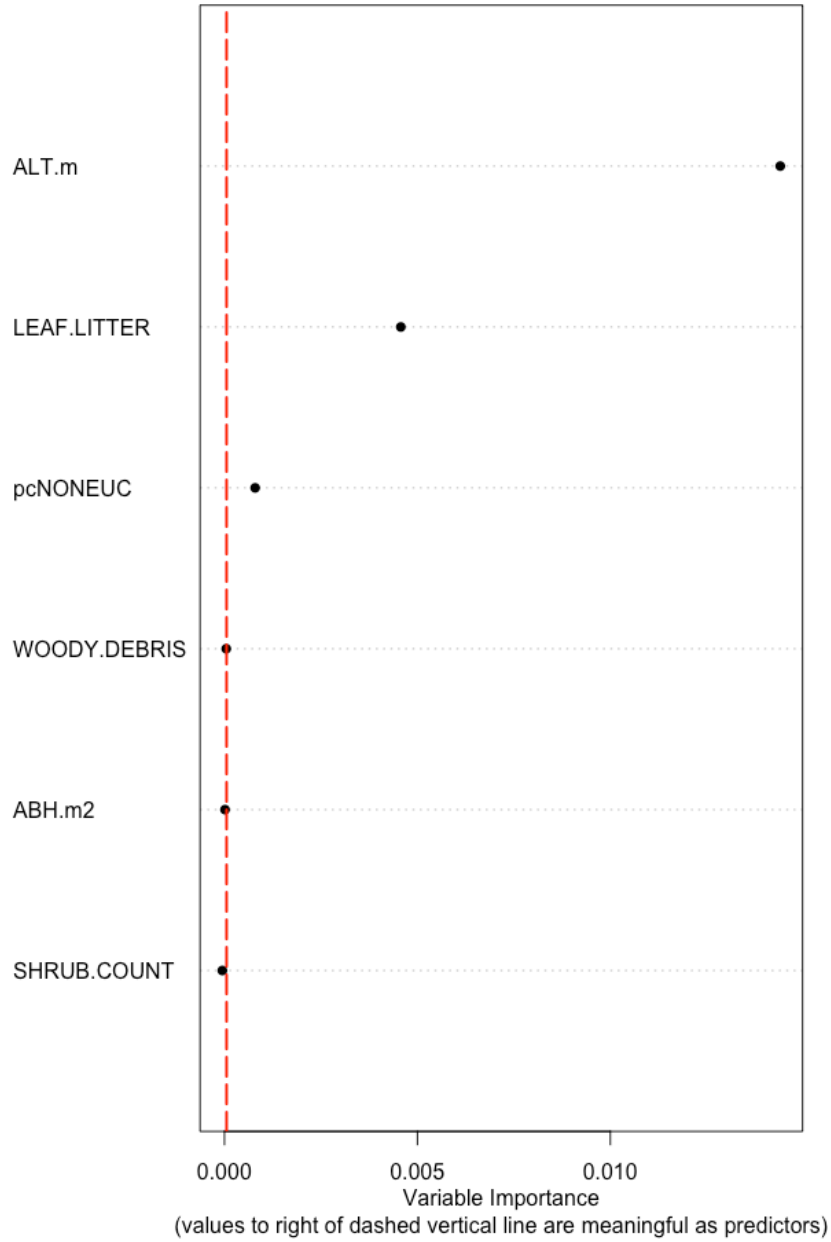
RESULT

```
ABH.m2           SHRUB.COUNT      WOODY.DEBRIS
-1.952247e-06    -2.959465e-05    2.072595e-05
LEAF.LITTER      pcNONEUC         ALT.m
4.546654e-03     7.789202e-04     1.452086e-02
```

```
dotchart(sort(fit.varimp)
```

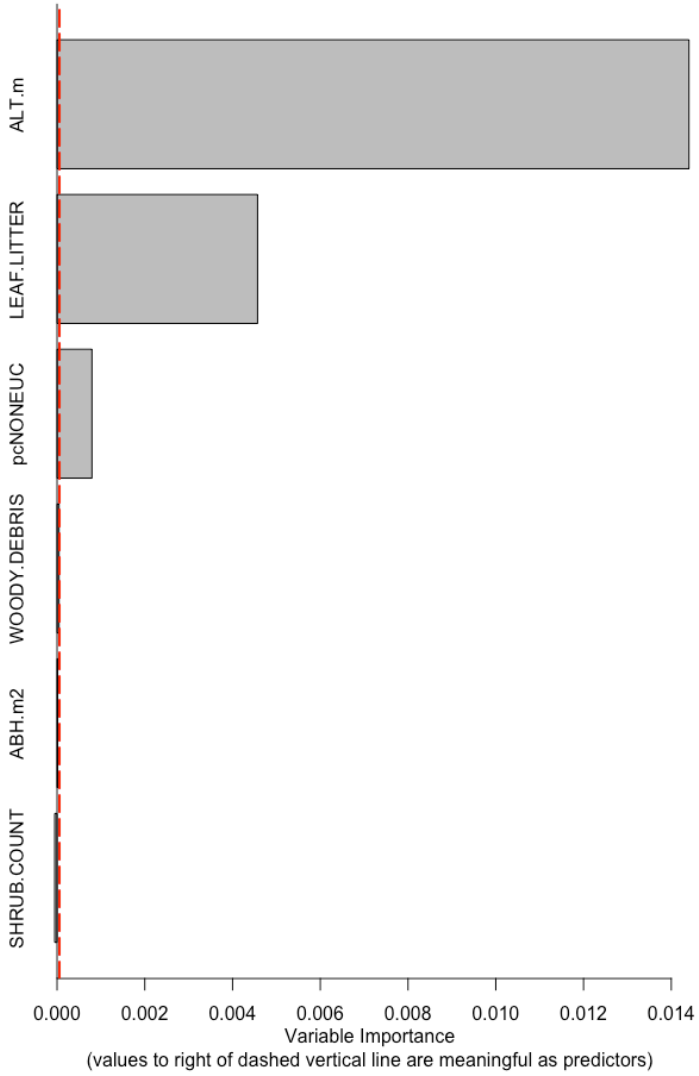


```
dotchart(sort(fit.varimp), pch = 16,
xlab="Variable Importance\n(values to right of dashed vertical
line are meaningful as predictors)")
abline(v=abs(min(fit.varimp)), col='red', lty='longdash',
lwd=2)
```



The same data can be presented as a barplot (overpage).

```
barplot(sort(fit.varimp), horiz=TRUE,
xlab="Variable Importance\n(values to right of dashed vertical
line are meaningful as predictors)")
abline(v=abs(min(fit.varimp)), col='red', lty='longdash',
lwd=2)
abline(v=0, col="black") # to add a line at zero
```



The above results suggest that Altitude of sites, Leaf Litter and percentage of non-Eucalyptus trees at sites are all important for explaining the abundance of agile antechinus. This is telling us the importance of these variables but we don't know anything about the direction of effect. Often when using Random Forests it is useful to include a graph showing the non-parametric correlation coefficients or some boxplots as well.

Structural Equation Modelling (Pathway Analysis)

Structural Equation Modelling is a way to test complex webs of positive and negative relationships for parsimony and significance. It allows for you to compare one theoretical 'structure' or 'web' of relationships with a second or third theoretical web of relationships and identify which pattern of relationships is best supported by the data. If you are interested in Structural Equation Modelling you should consider obtaining and reading Bill Shipley's *Cause and Correlation in Biology*. It is an easy to read, persuasive and interesting book focusing on how causal relationships might be identified in a tangle of correlational data.

Useful things about SEM

Allows for comparing complex relationships: biology is a complicated world full of interactions and variables feeding back on each other. SEM allows you to examine a whole set of interacting variables in one go, instead of piecemeal.

Arguably allows for inferences about cause and effect: this is more contentious, but at least some writers argue that SEM can allow correlational data to be investigated in a way that allows cause and effect to be unravelled. More typically, the view is that cause and effect cannot be shown without a controlled experiment, but the view that SEM provides some insight into cause and effect is becoming more widely accepted.

Considerations when using SEM

- **Latent to observed ratio:** A model needs to have at least 3 observed variables per 1 latent variable. Solutions include:
    - Add observed variable(s) or remove latent variable(s) (preferable)
    - Constrain two latent variables so that they have the same loading (less preferable)
- **Endogenous latent variables:** An endogenous latent variable will probably need three or more predictors (exogenous observed) variables contributing to explaining it. Otherwise, there may not be enough information to allow the model to predict the latent variabe.
- **Degrees of Freedom:** It is easy to saturate a SEM by including too many estimated parameters (latent variables and correlational lines) and too few observed variables (the actual observed data series). You will be warned by R if your degrees of freedom are negative and the model will refuse to build. In this situation, consider removing unobserved variables and/or lines of causal influence from the model.
- **Negative variance:** Problems that can occur is that if you force two highly correlated variables to be non-correlated. This can generate negative variance which can cause the model to collapse.
- **Covariance of predictors:** As with all linear models, covarying predictors can cause issues when building pathway models.
- **Normality of response variables:** Problems with normality of endogenous variables can cause issues with the model building.

Terminology: *One minor issue (but a confusing one) is that latent variables (unmeasured variables) are called 'factors' in the pathway analysis and SEM literature. In the following we use 'latent variable' only and avoid using 'factor' to avoid confusion, but if you are reading reviews or help articles on SEM keep in mind that the word 'factor' may not mean a categorical variable with levels, but rather may mean 'an unmeasured variable'.*

The main package used for SEM in R is **sem** and it contains most of the functions we will use here.

Load libraries
```
install.packages("sem")
library(sem)
```

Load data
```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')
```

We are going to have a go at taking the example conceptual flow diagram proposed by Johnstone et al. (2011) (shown earlier in this document) and turning it into a SEM for analysis. The original diagram is shown on the following page.

We need to take the diagram and turn it into a form that can be used as a template to construct a SEM in R. There are a number of variables, such as covariance values, that are not considered in the diagram as it was published, but which need to be taken into account.

If you are doing this step with your own data you can either write the diagram down on a piece of paper or create it inside a simple graphics program like Powerpoint or Keynote.

Also, a note to the wise. Structural Equation Models are sometimes viewed as a magic wand by biologists who hope these models can demonstrate causal relationships among complex variables including some that were not measured. As we are going to see, building Structure Equation Models is not straightforward. Often it is better to start small and apply a SEM process to just a part of an overall theory that makes good biological sense. In the end, only experience building SEMs will allow you to construct these models without (as many) problems.

The original form of the conceptual flow diagram we'll work with:

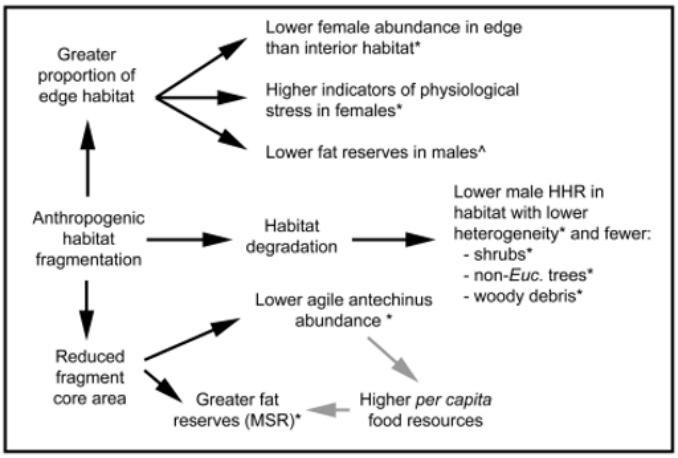


Figure X. Conceptual flow diagram of the main results. There are well established associations between anthropogenic habitat fragmentation and the creation of novel edge habitat, habitat change and habitat area reduction [3]. Associations supported by significant findings are indicated by *. Findings that are significant, but may be confounded by an interaction, are indicated by ^. Grey arrows indicate a theoretical mechanism by which an association could be operating.

Johnstone CP, Lill A, Reina RD - PLoS ONE (2011)

Pictorial representations used for SEM have a standard set of symbols and are called Path Diagrams. For this reason, SEM is also sometimes called pathway analysis. Before looking at how to create a path diagram, it is useful to define some terms (based on David A. Kenny, 2011):

- **Observed Variable:** a variable that has been measured
- **Latent Variable:** a variable that has not been measured
- **Exogenous Variable:** a variable that is not caused by another variable. Exogenous variables can cause one, two or more endogenous variables.
- **Endogenous Variable:** a variable that is caused by another variable. Endogenous variables can also cause other endogenous variables. e.g. *in a very simple example where habitat degradation causes increased predator activity causes decrease in small mammal abundance, habitat degradation is exogenous, predator activity is endogenous and small mammal abundance is also endogenous.*
- **Structural Coefficient:** a measure of change in the effected variable given one unit of change in the causal variable and no change in any other variable. This can be thought of as similar to a regression coefficient, although it is not always estimable using multiple regression.
- **Disturbance:** Equivalent to error in linear equations. This is the amount of change in an effected variable that is not attributable to any variables in the equation. Usually each endogenous variable will have a Disturbance.
- **Structural Model:** The total set of structural equations merged into a model of cause and effect.

Creating a path diagram: step 1

The first step is to work out what variables are **Observed** and what variables are **Latent** in the conceptual flow diagram. Although in the original 2011 published study the perimeter and area of forest fragments were measured, we will treat this as if they were not measured so that we can illustrate how to include **Latent** variables.

- Anthropogenic habitat fragmentation — Observed
- Proportion of Edge Habitat — Latent
- Habitat Degradation — Latent
- Area of Fragment — Observed
- Core area of fragment — Latent
- Female antechinus abundance — Observed
- Female antechinus stress metric — Observed (N:L ratio)
- Simpson's diversity index for shrub species — Observed
- Shrub Count — Observed
- Percentage of non-*Eucalyptus* trees — Observed
- Woody debris (logs) — Observed
- Total agile antechinus abundance — Observed
- Fat reserves — Observed (mass)
- Regenerative anaemia — Observed (RBC)
- *per capita* Food resources — Latent

Where N:L ratio is the ratio of neutrophils to lymphocytes in circulating peripheral blood and MSR is mass-size residuals in grams.

We're going to change the model slightly. Instead of predicting lower female abundance in edge than interior habitat in fragments, we will predict that there is lower female abundance in fragments due to an area effect. All continuous forest will be set to having an 'area' of 1000 ha, although this is really just functioning as an arbitrary dummy value because the areas of continuous forest were 10,000 ha or more. Additionally, although the researchers originally used Shannon's diversity index for shrubs, we will use Simpson's diversity index instead. These two indices should roughly agree, so we shouldn't expect this to be a serious deviation from the original theory outlined in the conceptual flow diagram.

- Anthropogenic habitat fragmentation — **FRAGMENTATION**
- Proportion of Edge Habitat — Latent
- Habitat Degradation — Latent
- Area of Fragment — **AREA.ha**
- Core area of fragment — Latent
- Sex of antechinus — **SEX**
- Total, F & M Antechinus abundance at site — **AGILIS, AGILIS.F, AGILIS.M**
- Female antechinus stress metric — **NL**
- Simpson's diveristy index for shrub species — **SIMSPONS.DIVERSITY**
- Shrub Count — **SHRUBS.COUNT**
- Percentage of non-*Eucalyptus* trees — **pcNONEUC**
- Woody debris (logs) — **WOODY.DEBRIS**
- Fat reserves — **MASS**
- Regenerative anaemia — **RBC**
- *per capita* Food resources — Latent

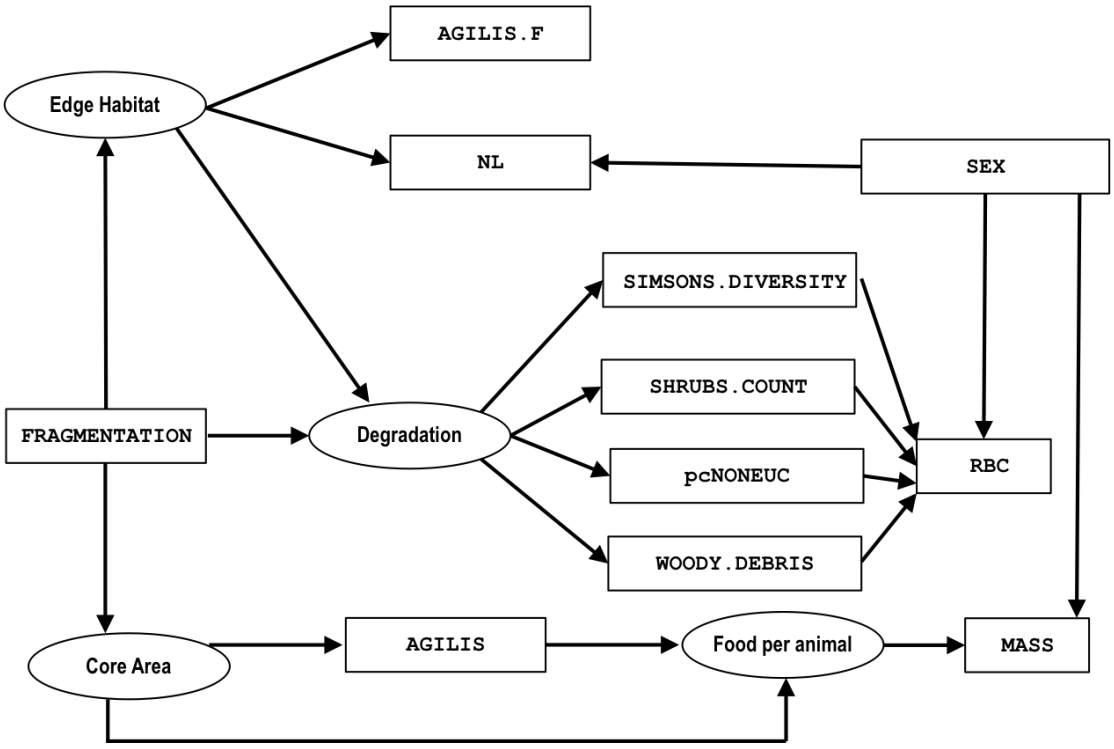
Creating a path diagram: step 2

We can now create the basic path diagram. Here are some basic rules to help keep the path diagram interpretable:

- Causal arrows start at the cause and end at the effect
- Double arrows are used to show feedback or mutual cause
- Observed variables are placed in boxes
- Latent variables are placed in ovals or circles

The following diagram uses the title names for variables in the file agilis-morphometrics.csv. In the original study Mass-size residuals were used to estimate fat reserves of antechinus and haemoglobin hematocrit residuals were used to estimate a condition call regenerative anaemia. We'll use much rougher but also simpler estimates. For an estimate of fat reserves we'll just use the mass (g) of the animals. For an estimate of regenerative anaemia we'll just use red blood cell count per litre.

Note also that we think there might be a causal effect of Edge Habitat on Degradation, both of which are unobserved, or latent variables.



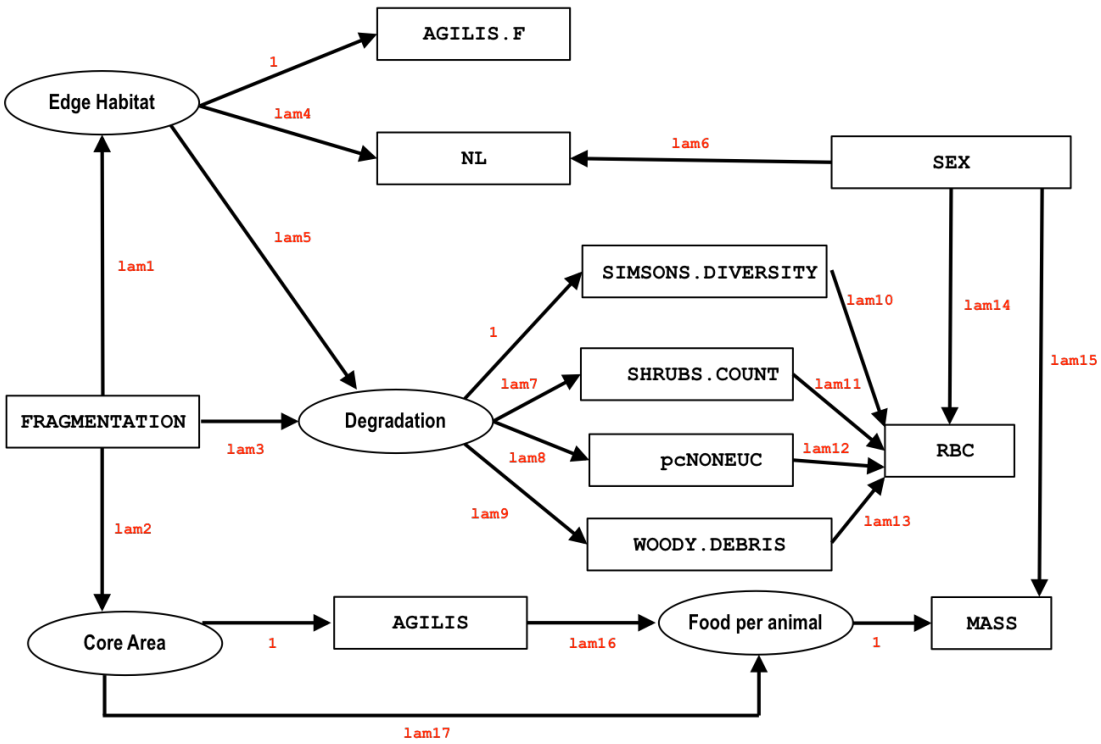
Creating a path diagram: step 3

However, the model is not complete yet. To be able to write out the model in equation form we need to create (arbitrary) names for the causal arrows (red) and a **Disturbance** variables (blue).

Rules for labelling causal arrows and disturbances

- Each endogenous variable receives disturbance
- Exogeneous variables do not receive disturbance
- Names for arrows are arbitrary (but a notation of lam1, lam2, lam3 is typical)
- At least one arrow leading away from a latent variable should be set to 1 to help us identify the scale of the corresponding latent variable

First, we'll add the names for the pathways. Note that for each latent variable one of the causal arrows leading away from the latent variable is not given a name. Instead it is set to a value of 1.



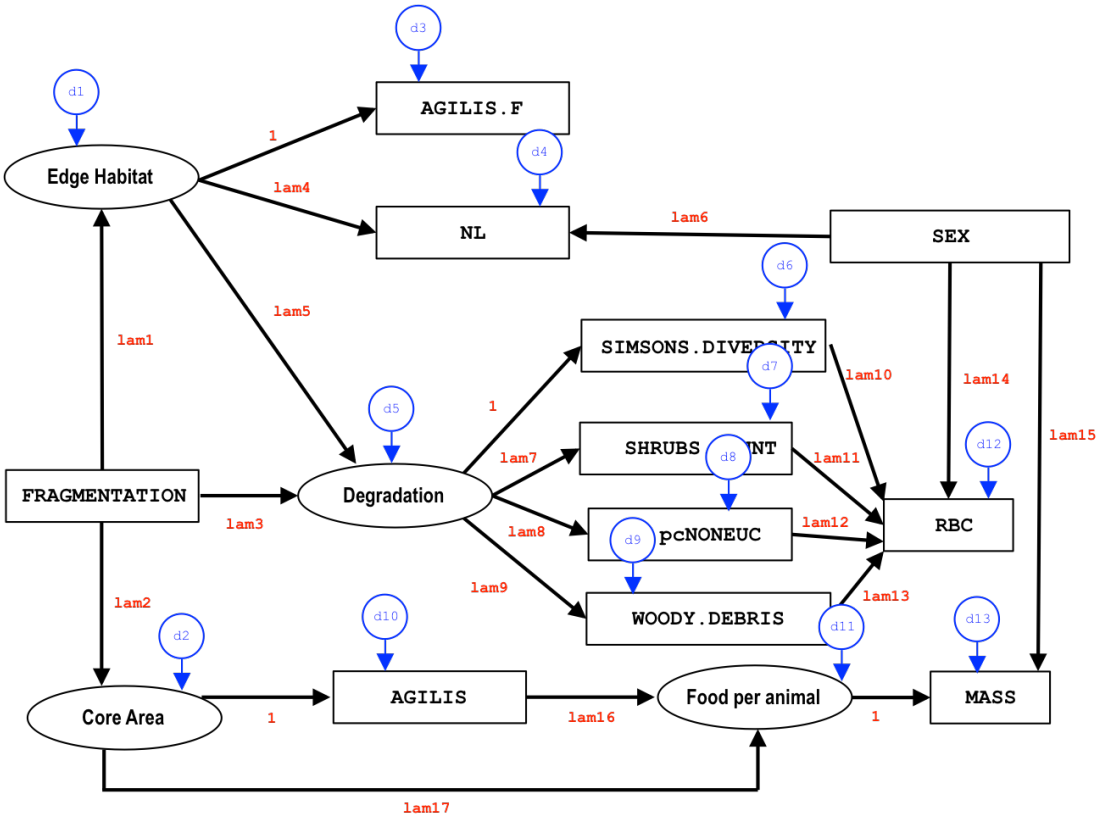
The next step is not strictly necessary for R, but it is useful to do simply to help us remember that all endogenous variables have a disturbance. Disturbance can be thought of as similar to error in a standard variance model: that is, the amount of unexplained variation left over once the variance due to the predictors of interest has been established.

On the next page the same diagram is shown, but with disturbances shown in blue. The disturbances are numbered so that we can remember that the disturbance affecting one variable is not affecting another variable.

Rules for labelling causal arrows and disturbances

- A double-headed arrow (reciprocal causality) cannot be applied to an endogenous variable
- Instead, the arrow is drawn between the disturbances of the two variables

We don't have an example of two endogenous variable mutually causing each other, but if we thought that **pcNONEUC** and **SHRUBS.COUNT** (for example) were casually influencing each other (that is the percentage of non-eucaluptus trees influnced shrubs numbers, and shrub numbers influenced the number of non-eucalyptus trees) then we would draw a double-headed arrow between the disturbances of **pcNONEUC** and **SHRUBS.COUNT**.



The Reticular Action Model (RAM)

The above step can (arguably) be skipped if you are comfortable with structural equations in R because R uses the Reticular Action Model (RAM) notation, which simplifies the process. You don't need to distinguish between disturbance error and measurement error. These are the notations used:

`A -> B`	Causal effect of A on B
`A <-> B`	Covariance A and B
`A <-> A`	**If A is endogenous:** the disturbance (error) associated with A
`A <-> A`	**If A is exogenous:** the measurement error associated with A

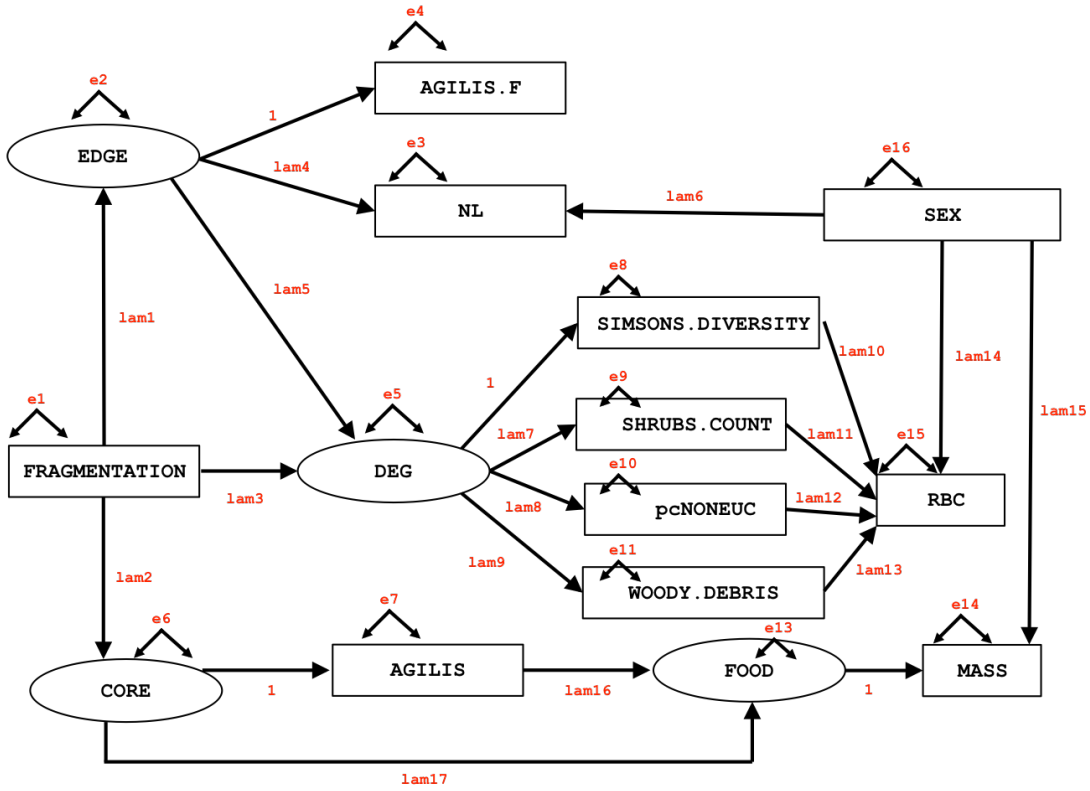
These four notations make up all the possible relationships in the pathway structure.

Creating a structural model

We now have enough information to start building a model. Specifying a model uses the **`specify.model()`** code in package **`sem`**. You need to type out each relationship, including the covarying relationships. WARNING: This is the step that usually creates the most problems. Forgetting a double arrow or getting a name wrong will cause substantial issues. We also need to define some arbitrary names for the latent variables at this stage. It's best to keep the names short but clear. I've decided to use the following:

- Proportion of Edge Habitat — Latent — **`EDGE`**
- Habitat Degradation — Latent — **`DEG`**
- Core area of fragment — Latent — **`CORE`**
- *per capita* Food resources — Latent — **`FOOD`**

And we need to assign pathway labels to the error effects. Remember that there is no need to distinguish between measurement error (of indictor / exogenous variables) and disturbance error (of response / endogenous variables), so all variables receive a similar error pathway.


e4
AGILIS.F
e2
EDGE
1
e3
lam4
NL
lam6
e16
SEX
lam5
lam1
e8
SIMSONS.DIVERSITY
lam10
lam14
lam15
e1
e5
1
e9
lam7
SHRUBS.COUNT
lam11
e15
FRAGMENTATION
DEG
lam3
e10
lam8
pcNONEUC
lam12
RBC
lam9
e11
lam13
WOODY.DEBRIS
lam2
e6
e7
e13
e14
CORE
1
AGILIS
lam16
FOOD
1
MASS
lam17

Before creating a model, we want to remove rows where there is missing data. Remember, this is a drastic way to handle missing values but sometimes it is the only feasible way to clean up a dataset. Load data (if you haven't already)

```
agilis <- read.table('agilis-morphometrics.csv',
header=T,sep=',')
```

Remove missing values

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

The model specification code is as follows:

```
model.fit <- specifyModel()
1:  FRAGMENTATION        ->    EDGE,                lam1,      NA
2:  FRAGMENTATION        ->    DEG,                 lam3,      NA
3:  FRAGMENTATION        ->    CORE,                lam2,      NA
4:  EDGE                 ->    AGILIS.F,            NA,        1
5:  EDGE                 ->    NL,                  lam4,      NA
6:  EDGE                 ->    DEG                  lam5,      NA
7:  DEG                  ->    SIMPSONS.DIVERSITY,  NA,        1
8:  DEG                  ->    SHRUBS.COUNT,        lam7,      NA
9:  DEG                  ->    pcNONEUC,            lam8,      NA
10: DEG                  ->    WOODY.DEBRIS,        lam9,      NA
11: SIMPSONS.DIVERSITY   ->    RBC,                 lam10,     NA
12: SHRUBS.COUNT         ->    RBC,                 lam11,     NA
13: pcNONEUC             ->    RBC,                 lam12,     NA
14: WOODY.DEBRIS         ->    RBC,                 lam13,     NA
14: CORE                 ->    AGILIS,              NA,        1
15: AGILIS               ->    FOOD,                lam16,     NA
16: CORE                 ->    FOOD,                lam17,     NA
17: FOOD                 ->    MASS,                NA,        1
18: SEX                  ->    NL,                  lam6,      NA
19: SEX                  ->    RBC,                 lam14,     NA
20: SEX                  ->    MASS,                lam15,     NA
21: FRAGMENTATION        <->   FRAGMENTATION,       e1,        NA
22: EDGE                 <->   EDGE,                e2,        NA
23: NL                   <->   NL,                  e3,        NA
24: AGILIS.F             <->   AGILIS.F,            e4,        NA
25: DEG                  <->   DEG,                 e5,        NA
26: CORE                 <->   CORE,                e6,        NA
27: AGILIS               <->   AGILIS,              e7,        NA
28: SIMPSONS.DIVERSITY   <->   SIMPSONS.DIVERSITY,  e8,        NA
29: SHRUBS.COUNT         <->   SHRUBS.COUNT,        e9,        NA
30: pcNONEUC             <->   pcNONEUC,            e10,       NA
31: WOODY.DEBRIS         <->   WOODY.DEBRIS,        e11,       NA
32: FOOD                 <->   FOOD,                e13,       NA
33: MASS                 <->   MASS,                e14,       NA
34: RBC                  <->   RBC,                 e15,       NA
35: SEX                  <->   SEX,                 e16,       NA
```

IMPORTANT: The next step is creating a covariance matrix. The covariance matrix should include all measured variables. If a variable is named in the model specification (above) but is not in the covariance matrix R will assume it is a latent (unmeasured) variable. This will be true for misspelling or case errors. If you call a variable **`WOODYDEBRIS`** in the model and **`WOODY.DEBRIS`** in the covariance matrix R will not be able to identify these as the same variable. Take care!

Creating a covariance matrix: step 1

We need to create a covariance matrix cor all observed variables. The latent variances and the disturbance variables do not need to be included, and as we do not have data for them anyway they could not be included in this matrix.

This is a lazy workaround to set up a dataset that entirely consists of your variables of interest and nothing else. Make sure at this point that all your variables are in a numeric form. If you want to generate covariance matrices using factors, packages for this purpose do exist but they are outside the scope of this document. The **`hetcor`** function in the **`polycor`** package is one place to start if you are interested in generating correlation matrices using factors. However, for now we will work with the basic R **`cov`** function.

Create a dataset using one of the variables…

```
cov.data <- data.frame(agilis$FRAGMENTATION)
cov.data$FRAGMENTATION <- cov.data$agilis.FRAGMENTATION
head(cov.data) # look at the output to check it is correct
```

Now we can add all of the other variables of interest.

```
# Now add the other observed variables
cov.data$SIMPSONS.DIVERSITY <- agilis$SIMPSONS.DIVERSITY
cov.data$SHRUBS.COUNT <- agilis$SHRUBS.COUNT
cov.data$pcNONEUC <- agilis$pcNONEUC
cov.data$WOODY.DEBRIS <- agilis$WOODY.DEBRIS
cov.data$AGILIS <- agilis$AGILIS
cov.data$SEX <- agilis$SEX
cov.data$NL <- agilis$NL
cov.data$RBC <- agilis$RBC
cov.data$MASS <- agilis$MASS
```

Creating a covariance matrix: step 2

Now create a covariance matrix using the **`cov`** function. Check everything has worked. Factors or missing values will create problems.

```
cov.fit <- cov(cov.data)
cov.fit
```

COUNTING ROW NUMBERS

Generating a SEM requires entering the number of observations. The easiest way to do this is either embed **`nrow(your.data)`** in the appropriate place in the dataset or use the following code to count the rows and assign them to a variable called **`n`**.

```
n <- nrow(cov.data)
n
```

Fitting the model

You can now fit the final model. Use the following code:

```
model.sem <- sem(model.fit, cov.fit, n)
```

You will receive the following error message. What went wrong?

```
> model.sem <- sem(model.fit, cov.fit, n)
Error in 1:m : NA/NaN argument
In addition: Warning messages:
1: In sem.semmod(model.fit, cov.fit, n) :
  The following observed variables are in the input covariance or raw-moment matrix
but do not appear in the model:
agilis.FRAGMENTATION

2: In sem.default(ram, S = S, N = N, raw = raw, data = data, pattern.number =
pattern.number,  :
  S is numerically singular: expect problems
3: In sem.default(ram, S = S, N = N, raw = raw, data = data, pattern.number =
pattern.number,  :
  S is not positive-definite: expect problems
```

The first warning about **agilis.FRAGMENTATION** can be ignored. That is just a consequence of the way we built a correlation structure.

The second two error messages are more serious and actually the model has not been built as all as a consequence. **S** is the covariance matrix, so we appear to have problems with the fundamental model.

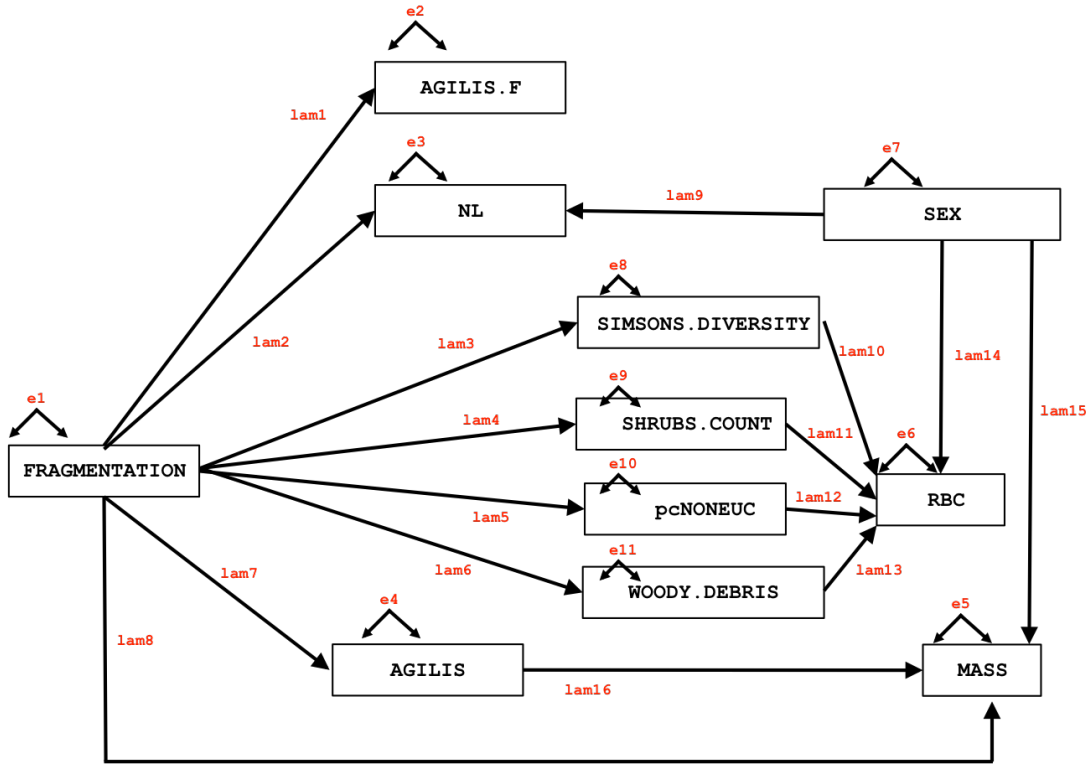
The largest problem with SEM is inexperience with modelling leading to construction of an inappropriate model. Looking at the model more carefully, we might have done a good job of mimicking our original conceptual flow diagram, but we have done a poor job of creating a model that can be analysed. There are three obvious problems:

- Too many latent variables
- Too few measured predictors feeding into latent variables
- Maybe a problem with normality of endogenous variables?

We want a ratio of no less than 3:1 measured to latent (unmeasured variables). We have four latent variables and 12 measured variables. This is exactly a 3:1 ratio which is borderline in terms of model creation.

An even bigger problem is that we have only one or two arrows from measured variables feeding into each latent variable. A minimum of three arrows per latent variable is needed in most models. This also raises some questions around why we even need the latent variables. If **FRAGMENTATION** is the only variable predicting the unmeasured **EDGE** variable, then why include **EDGE** at all? Surely **FRAGMENTATION** would be just as good a predictor for downstream effects (and in this case better because we measured **FRAGMENTATION**).

Regarding the final point, if you try to fit a model and it collapses one thing to remember is that this is a form of linear modelling. The response (endogenous) variables may need to be transformed for normality.
Here's a new path diagram with the (in this case rather meaningless) latent variables removed:



To be on the safe side we might as well apply a transformation to all of the response variables we are using as well. We'll use the (rather drastic) Rank Normal transformation from package **GenABEL**.

```
library(GenABEL)

agilis$rnAGILIS.F <- rntransform(agilis$AGILIS.F)
agilis$rnAGILIS <- rntransform(agilis$AGILIS)
agilis$rnRBC <- rntransform(agilis$RBC)
agilis$rnMASS <- rntransform(agilis$MASS)
agilis$rnNL <- rntransform(agilis$NL)
head(agilis)
```

This will vastly simplify the model but to make it even easier let's start by focusing down on just a part of the model. We'll start with the variables that predict **MASS** (without **SEX**).

```
model.fit <- specifyModel()
FRAGMENTATION            ->    rnAGILIS,            lam7,    NA
FRAGMENTATION            ->    rnMASS,              lam8,    NA
FRAGMENTATION            ->    rnNL,                lam2,    NA
rnAGILIS                 ->    rnMASS,              lam16,   NA
FRAGMENTATION            <->   FRAGMENTATION,       e1,      NA
rnAGILIS                 <->   rnAGILIS,            e4,      NA
rnMASS                   <->   rnMASS,              e5,      NA
rnNL                     <->   rnNL,                e3,      NA
```

Load library **semPlot** so we can draw an attractive plot:

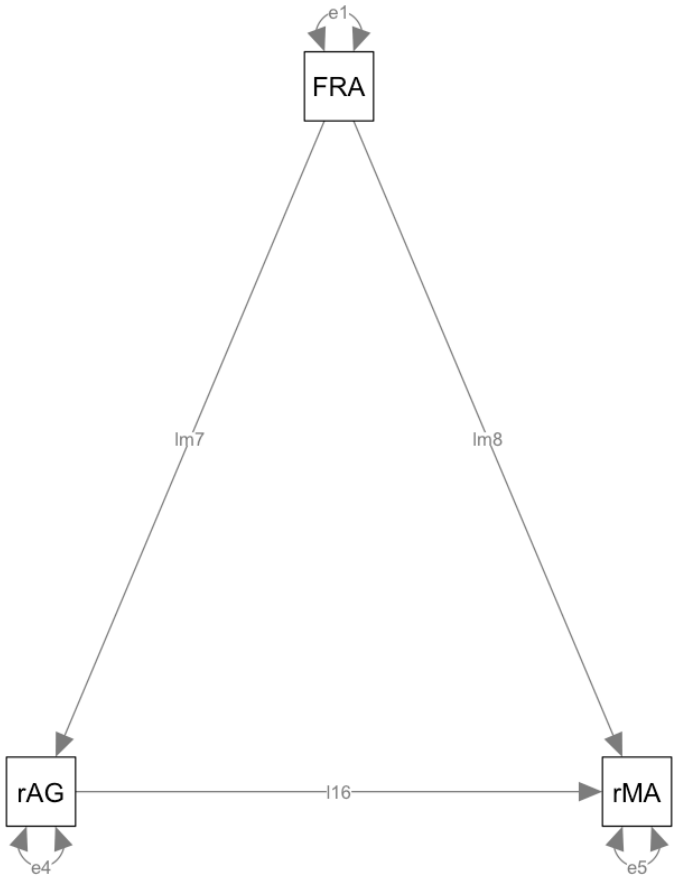
```
install.packages("semPlot")
library(semPlot)
```

Fit the model, check the paths and the summary. Instead of counting the **n** and working out a covariance matrix we will pass the data directly to the **sem** function.

```
model.sem<-sem(model.fit, data=agilis)
semPaths(model.sem)
summary(model.sem)
```

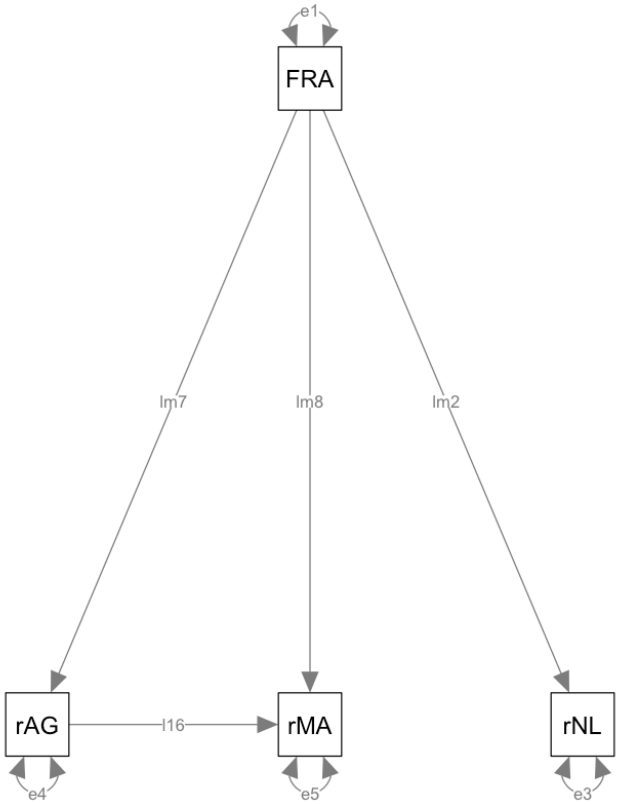
RESULT

```
> summary(model.sem)

 Model Chisquare =  0   Df =  0 Pr(>Chisq) = NA
 AIC =  12
 BIC =  0

 Normalized Residuals
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-1.313e-15 -1.114e-15 -3.744e-16 -4.118e-16  0.000e+00  3.916e-16

 R-square for Endogenous Variables
rnAGILIS   rnMASS
  0.3443   0.0457

 Parameter Estimates
      Estimate   Std Error  z value     Pr(>|z|)
lam7  -1.2053537 0.11762323 -10.2475816 1.213416e-24 rnAGILIS <--- FRAGMENTATION
lam8  -0.1617666 0.17000527  -0.9515385 3.413311e-01 rnMASS <--- FRAGMENTATION
lam16 -0.2435273 0.08275803  -2.9426422 3.254243e-03 rnMASS <--- rnAGILIS
e1     0.2509453 0.02509453  10.0000000 1.523971e-23 FRAGMENTATION <--> FRAGMENTATION
e4     0.6943769 0.06943769  10.0000000 1.523971e-23 rnAGILIS <--> rnAGILIS
e5     0.9511424 0.09511424  10.0000000 1.523971e-23 rnMASS <--> rnMASS

 Iterations =  0
```



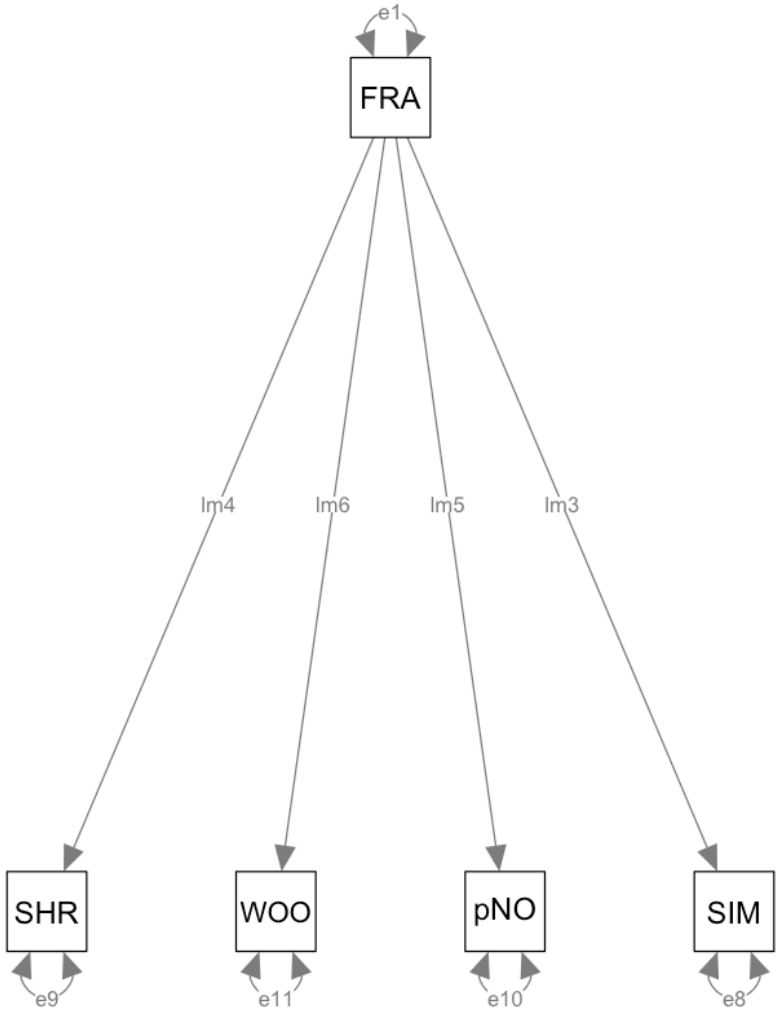
This is a fairly simple model but it does capture one part of our overall theory. Now, we'll try adding NL to the model using the transformed rnNL variable.

```
model.fit <- specifyModel()
FRAGMENTATION      ->     rnAGILIS,      lam7,     NA
FRAGMENTATION      ->     rnMASS,        lam8,     NA
FRAGMENTATION      ->     rnNL,          lam2,     NA
rnAGILIS           ->     rnMASS,        lam16,    NA
FRAGMENTATION      <->    FRAGMENTATION, e1,       NA
rnAGILIS           <->    rnAGILIS,      e4,       NA
rnMASS             <->    rnMASS,        e5,       NA
rnNL               <->    rnNL,          e3,       NA

model.sem<-sem(model.fit, data=agilis)
semPaths(model.sem)
```



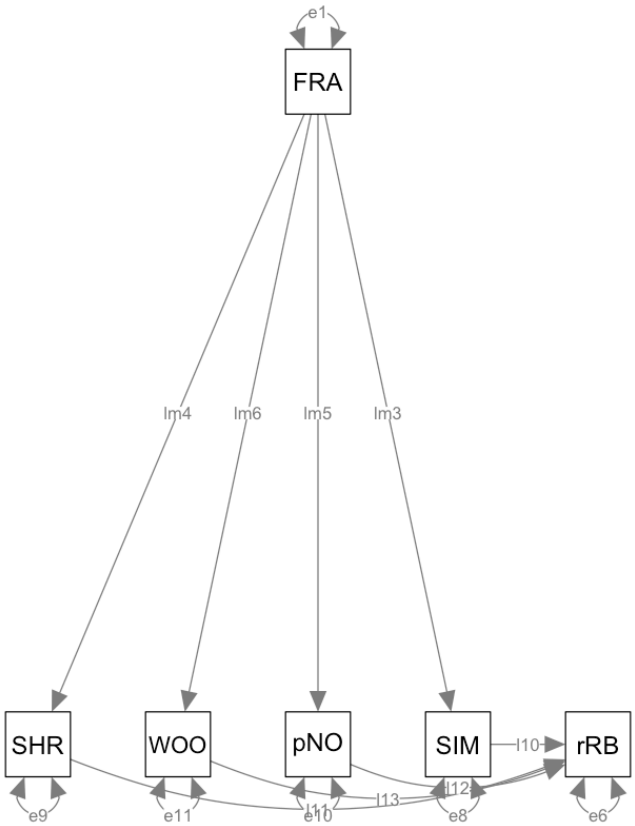
So far so good but if we try adding any more variables the model collapses. Let's have a look at constructing another part of the model.

```
model.fit <- specifyModel()
FRAGMENTATION      ->   SIMPSONS.DIVERSITY,    lam3,  NA
FRAGMENTATION      ->   SHRUBS.COUNT,          lam4,  NA
FRAGMENTATION      ->   pcNONEUC,              lam5,  NA
FRAGMENTATION      ->   WOODY.DEBRIS,          lam6,  NA
FRAGMENTATION      <->  FRAGMENTATION,         e1,    NA
SIMPSONS.DIVERSITY <->  SIMPSONS.DIVERSITY,    e8,    NA
SHRUBS.COUNT       <->  SHRUBS.COUNT,          e9,    NA
pcNONEUC           <->  pcNONEUC,              e10,   NA
WOODY.DEBRIS       <->  WOODY.DEBRIS,          e11,   NA

model.sem<-sem(model.fit, data=agilis)
semPaths(model.sem)
```



Let's have a go at adding the next set of associations through to RBC.

```
model.fit <- specifyModel()
FRAGMENTATION      ->   SIMPSONS.DIVERSITY,  lam3,      NA
FRAGMENTATION      ->   SHRUBS.COUNT,        lam4,      NA
FRAGMENTATION      ->   pcNONEUC,            lam5,      NA
FRAGMENTATION      ->   WOODY.DEBRIS,        lam6,      NA
SIMPSONS.DIVERSITY ->   rnRBC,               lam10,     NA
SHRUBS.COUNT       ->   rnRBC,               lam11,     NA
pcNONEUC           ->   rnRBC,               lam12,     NA
WOODY.DEBRIS       ->   rnRBC,               lam13,     NA
FRAGMENTATION      <->  FRAGMENTATION,       e1,        NA
SIMPSONS.DIVERSITY <->  SIMPSONS.DIVERSITY,  e8,        NA
SHRUBS.COUNT       <->  SHRUBS.COUNT,        e9,        NA
pcNONEUC           <->  pcNONEUC,            e10,       NA
WOODY.DEBRIS       <->  WOODY.DEBRIS,        e11,       NA
rnRBC              <->  rnRBC,               e6,        NA

model.sem<-sem(model.fit, data=agilis)
semPaths(model.sem)
```



You can use the following commands to look at the hypothesis tests, standardized coefficients (loadings) and residuals of the model.

Hypothesis tests

`summary(model.sem)`

RESULT

```
Model Chisquare =  45.53198   Df =  7 Pr(>Chisq) = 1.078023e-07
 AIC =  73.53198
 BIC =  8.40885

 Normalized Residuals
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.52100 -0.06109  0.00000  0.13090  0.53980  3.18300

 R-square for Endogenous Variables
SIMPSONS.DIVERSITY       SHRUBS.COUNT           pcNONEUC       WOODY.DEBRIS              rnRBC
            0.0028             0.0173             0.3306             0.0612             0.1723

 Parameter Estimates
      Estimate      Std Error    z value     Pr(>|z|)
lam3  -1.707283e-02 2.296528e-02 -0.74341927 4.572279e-01 SIMPSONS.DIVERSITY <--- FRAGMENTATION
lam4  -2.097423e+01 1.119060e+01 -1.87427198 6.089295e-02 SHRUBS.COUNT <--- FRAGMENTATION
lam5  -3.102159e-01 3.121163e-02 -9.93911169 2.813246e-23 pcNONEUC <--- FRAGMENTATION
lam6   4.485230e+00 1.242693e+00  3.60928380 3.070436e-04 WOODY.DEBRIS <--- FRAGMENTATION
lam10 -5.657973e-03 3.996601e-01 -0.01415696 9.887048e-01 rnRBC <--- SIMPSONS.DIVERSITY
lam11 -1.380908e-03 8.163211e-04 -1.69162380 9.071773e-02 rnRBC <--- SHRUBS.COUNT
lam12 -1.098146e+00 2.439994e-01 -4.50061045 6.775858e-06 rnRBC <--- pcNONEUC
lam13 -3.480869e-02 7.238257e-03 -4.80898803 1.516963e-06 rnRBC <--- WOODY.DEBRIS
e1     2.509453e-01 2.509453e-02 10.00000000 1.523971e-23 FRAGMENTATION <--> FRAGMENTATION
e8     2.646991e-02 2.646991e-03 10.00000000 1.523971e-23 SIMPSONS.DIVERSITY <-->
                                                          SIMPSONS.DIVERSITY
e9     6.285151e+03 6.285151e+02 10.00000000 1.523971e-23 SHRUBS.COUNT <--> SHRUBS.COUNT
e10    4.889246e-02 4.889246e-03 10.00000000 1.523971e-23 pcNONEUC <--> pcNONEUC
e11    7.750619e+01 7.750619e+00 10.00000000 1.523971e-23 WOODY.DEBRIS <--> WOODY.DEBRIS
e6     8.470803e-01 8.470803e-02 10.00000000 1.523971e-23 rnRBC <--> rnRBC

 Iterations =  0
```

Standard coefficients

```
stdCoef(model.sem)
```

RESULT

```
            Std. Estimate
lam3   lam3 -0.0524951989      SIMPSONS.DIVERSITY <--- FRAGMENTATION
lam4   lam4 -0.1313822351            SHRUBS.COUNT <--- FRAGMENTATION
lam5   lam5 -0.5749995324               pcNONEUC <--- FRAGMENTATION
lam6   lam6  0.2472884193            WOODY.DEBRIS <--- FRAGMENTATION
lam10 lam10 -0.0009112061              rnRBC <--- SIMPSONS.DIVERSITY
lam11 lam11 -0.1091651121                    rnRBC <--- SHRUBS.COUNT
lam12 lam12 -0.2933772703                       rnRBC <--- pcNONEUC
lam13 lam13 -0.3126355591                    rnRBC <--- WOODY.DEBRIS
e1       e1  1.0000000000          FRAGMENTATION <--> FRAGMENTATION
e8       e8  0.9972442541 SIMPSONS.DIVERSITY <--> SIMPSONS.DIVERSITY
e9       e9  0.9827387083            SHRUBS.COUNT <--> SHRUBS.COUNT
e10     e10  0.6693755377               pcNONEUC <--> pcNONEUC
e11     e11  0.9388484377            WOODY.DEBRIS <--> WOODY.DEBRIS
e6       e6  0.8277231049                         rnRBC <--> rnRBC
```

Residuals

```
residuals(model.sem)
```

RESULT

```
                  FRAGMENTATION SHRUBS.COUNT  WOODY.DEBRIS      pcNONEUC SIMPSONS.DIVERSITY        rnRBC
FRAGMENTATION      0.000000e+00    0.0000000  4.440892e-16  0.000000e+00      -8.673617e-19  0.099834579
SHRUBS.COUNT       0.000000e+00    0.0000000  1.636113e+02 -3.863186e+00       9.510381e-01 -1.458133590
WOODY.DEBRIS       4.440892e-16  163.6113144  1.421085e-14 -3.793072e-01       5.650461e-02  0.190282885
pcNONEUC           0.000000e+00   -3.8631856 -3.793072e-01  1.387779e-17      -4.595932e-03  0.018563895
SIMPSONS.DIVERSITY -8.673617e-19    0.9510381  5.650461e-02 -4.595932e-03      -3.469447e-18  0.001766858
rnRBC              9.983458e-02   -1.4581336  1.902829e-01  1.856389e-02       1.766858e-03 -0.025005818
```

From the above models we can see that Fragmentation did have a significant effect on the percentage of non-Eucalyptus trees, the shrubs count and the woody debris but had no effect on diversity of shrubs (measured by Simpson's diversity). Of these environmental variables, only percentage of non-Eucalyptus trees and woody debris had an effect on RBC.

It is also possible to look at the total, direct and indirect effects in a **sem**:

Direct and indirect effects

```
effects(model.sem)
```

RESULT

```
Total Effects (column on row)
                  FRAGMENTATION  SHRUBS.COUNT  WOODY.DEBRIS  pcNONEUC SIMPSONS.DIVERSITY
SIMPSONS.DIVERSITY   -0.01707283 -1.004635e-20  0.000000e+00  0.000000        0.000000000
SHRUBS.COUNT        -20.97422680 -1.110223e-16  0.000000e+00  0.000000        0.000000000
pcNONEUC             -0.31021587 -2.172013e-18 -1.895620e-17  0.000000        0.000000000
WOODY.DEBRIS          4.48522998  3.112253e-17  0.000000e+00  0.000000        0.000000000
rnRBC                 0.21359749 -1.380908e-03 -3.480869e-02 -1.098146       -0.005657973
```

For all of the above outputs a lot of cleaning up will be needed before the results are fit for a results table. However there is a lot of interesting information here and there is a possibility that genuine complex relations and indirect causal chains might be illuminated using this method.

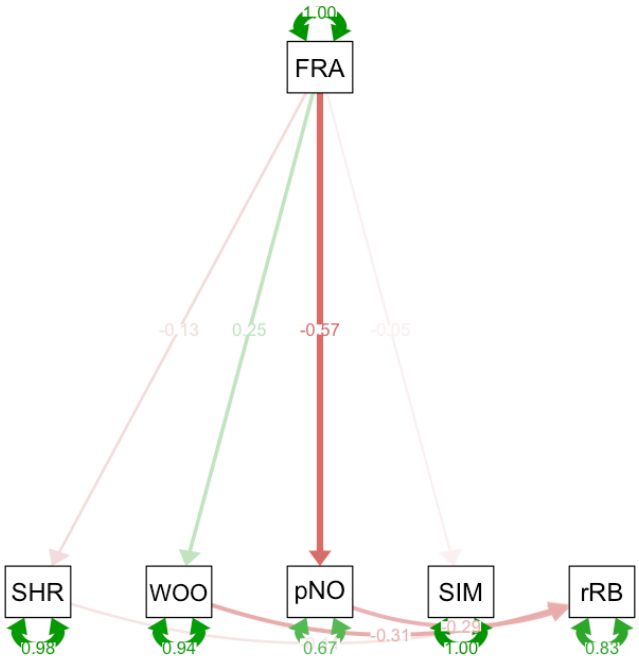
You can also compare two models to check which is more parsimonious. The standard **anova** code for model comparison works for **sems**. Here is an example:

```
model1.fit <- specifyModel()
FRAGMENTATION         ->  SIMPSONS.DIVERSITY,    lam3,     NA
FRAGMENTATION         ->  SHRUBS.COUNT,          lam4,     NA
FRAGMENTATION         ->  pcNONEUC,              lam5,     NA
FRAGMENTATION         ->  WOODY.DEBRIS,          lam6,     NA
SIMPSONS.DIVERSITY    ->  rnRBC,                 lam10,    NA
SHRUBS.COUNT          ->  rnRBC,                 lam11,    NA
pcNONEUC              ->  rnRBC,                 lam12,    NA
WOODY.DEBRIS          ->  rnRBC,                 lam13,    NA
FRAGMENTATION         <-> FRAGMENTATION,         e1,       NA
SIMPSONS.DIVERSITY    <-> SIMPSONS.DIVERSITY,    e8,       NA
SHRUBS.COUNT          <-> SHRUBS.COUNT,          e9,       NA
pcNONEUC              <-> pcNONEUC,              e10,      NA
WOODY.DEBRIS          <-> WOODY.DEBRIS,          e11,      NA
rnRBC                 <-> rnRBC,                 e6,       NA

model2.fit <- specifyModel()
FRAGMENTATION         ->  SHRUBS.COUNT,          lam4,     NA
FRAGMENTATION         ->  pcNONEUC,              lam5,     NA
FRAGMENTATION         ->  WOODY.DEBRIS,          lam6,     NA
SHRUBS.COUNT          ->  rnRBC,                 lam11,    NA
pcNONEUC              ->  rnRBC,                 lam12,    NA
WOODY.DEBRIS          ->  rnRBC,                 lam13,    NA
FRAGMENTATION         <-> FRAGMENTATION,         e1,       NA
SHRUBS.COUNT          <-> SHRUBS.COUNT,          e9,       NA
pcNONEUC              <-> pcNONEUC,              e10,      NA
WOODY.DEBRIS          <-> WOODY.DEBRIS,          e11,      NA
rnRBC                 <-> rnRBC,                 e6,       NA
SIMPSONS.DIVERSITY    <-> SIMPSONS.DIVERSITY,    e8,       NA

model1.sem<-sem(model1.fit, data=agilis)
model2.sem<-sem(model2.fit, data=agilis)

anova(model.1.sem, model.2.sem)

anova(model1.sem,model2.sem)
LR Test for Difference Between Models

           Model Df Model Chisq Df LR Chisq Pr(>Chisq)
model1.sem         7      45.532
model2.sem         9      46.084  2  0.55211     0.7588
```

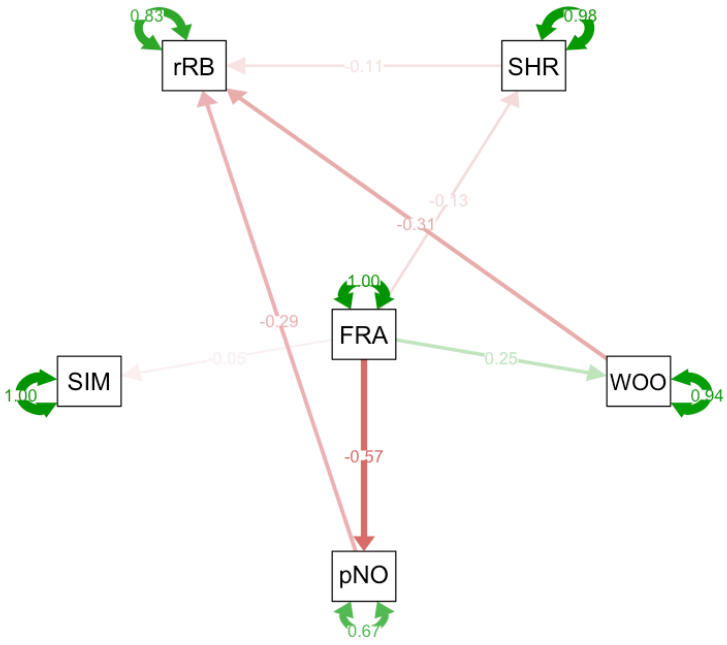
Important: to be able to compare models both models must have the same set of variables so that the covariance matrices can be compared.

Finally, it is worth noting that the semPaths plotting package can do some nice things in terms of layout. Here are some useful options:

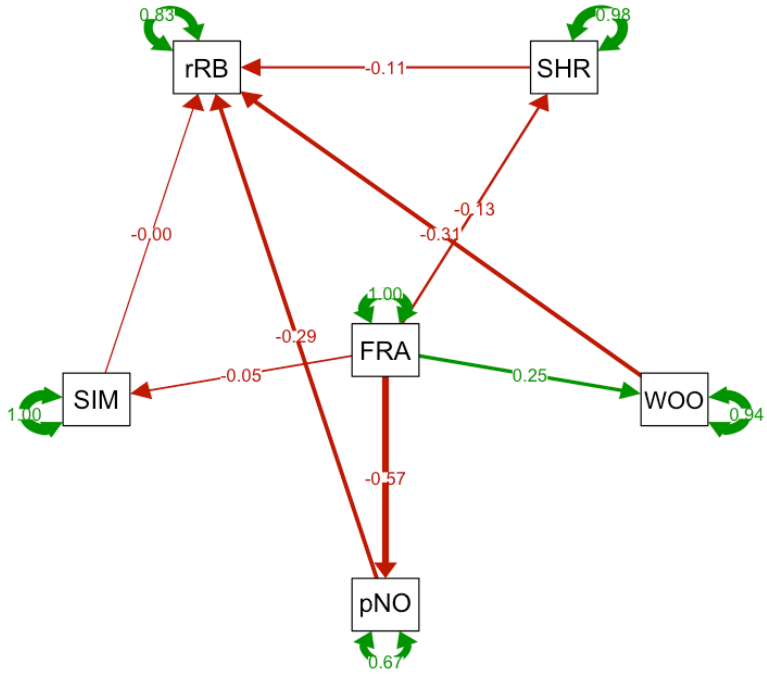
```
semPaths(model.sem, what = "std", layout = "tree")
```



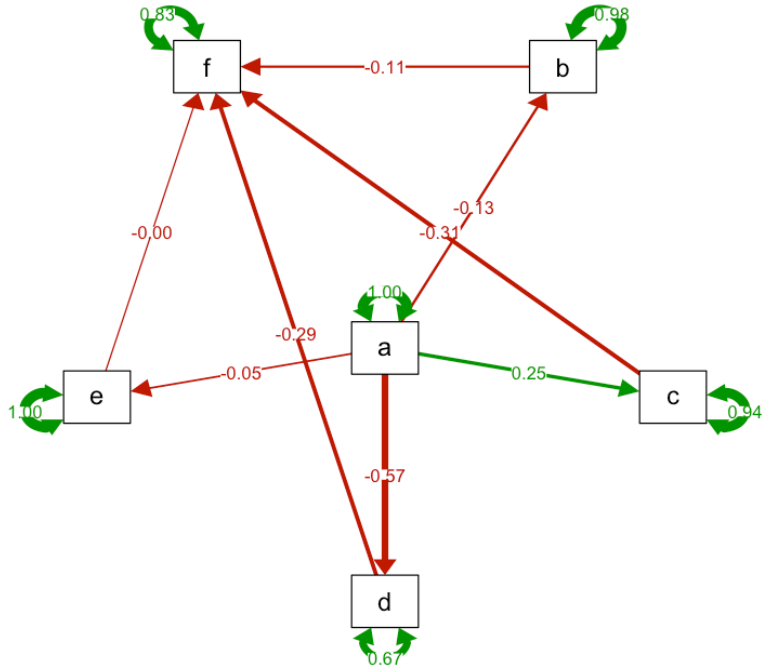

```
semPaths(model.sem, what = "std", layout = "circle")
```



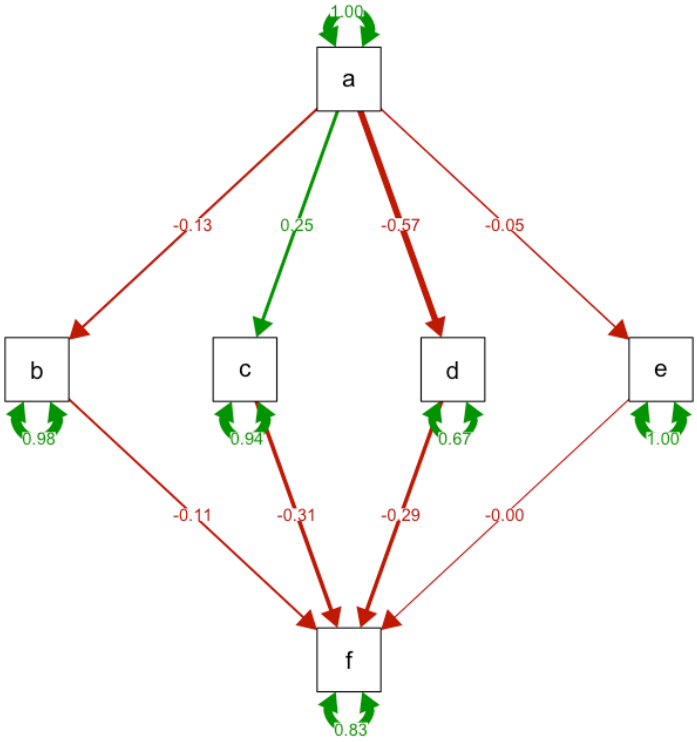
```
semPaths(model.sem, what = "std", layout = "circle", fade=FALSE)
```



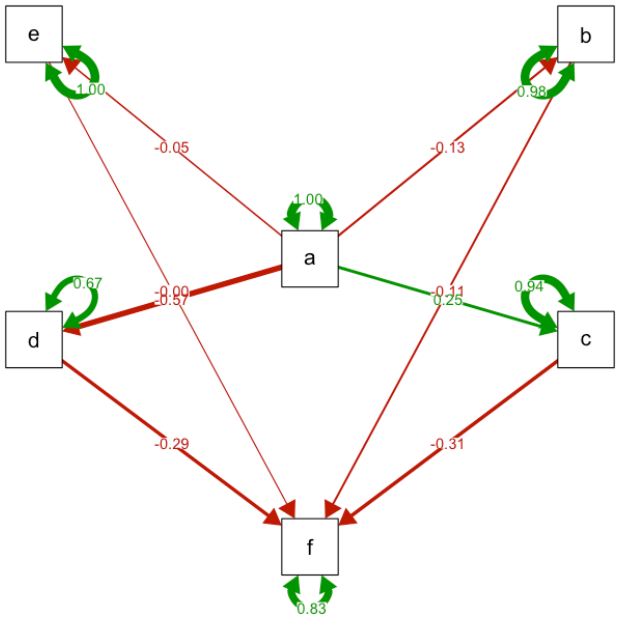

```
semPaths(model.sem, what = "std", layout = "circle",
fade=FALSE,nodeLabels=letters[1:6])
```



```
semPaths(model.sem, what = "std", layout = "tree2",
fade=FALSE,nodeLabels=letters[1:6])
```




```
semPaths(model.sem, what = "std", layout = "circle2",
fade=FALSE,nodeLabels=letters[1:6])
```



What are the take-home messages for pathway analyses?

- If you are going to use structural equation models you'll need to develop a good understanding of what you are doing.
- Start with smaller models first and build onto them.
- Don't try to throw everything into a model all at once.
- Remember to be careful with how many latent (unmeasured) variables you include (if any).
- Ensure that your models are biologically meaningful.

Likelihoods, log likelihoods & AICs

First off, let's start getting into some definitions:

Q. What is the difference between a probability and a likelihood?

Most people who are learning statistics are surprised to learn that statistics is not monolithic. There are competing theories and competing ways to applying statistical analysis to a set of data to obtain some sort of idea about what the data means.

Science is primarily about looking for patterns or regularities in nature and the universe and constructing theories about these patterns. We want our theories to be predicative and we test them to check if they are. For some reason, this seems to work. We don't know why the universe should be regular or have repeatable laws, but it seems to.

Deep time evolution can be taken as evidence supporting the idea that the universe, and importantly for us, biological systems, function in repeatable ways that have predictable patterns. If there were no predictable patterns, evolution would not function as a process. In particular, convergent evolution implies that there are biological pressures that select for a trait, and because the selection pressure is consistent, the trait confers an advantage in the future.

Not all selection pressures are consistent though. Some selection pressures seem to have come and gone. Large, piercing teeth was seemingly an advantage for mammal predators from the Miocene to the Pleistocene, but no modern predators sport these huge teeth and we don't know why they were selected for convergently in both sparassodonts (close relatives of marsupials) and eutherians (felids in particular).

To discuss different theories about how patterns is nature are best understood, we need to return to some terminology we learnt at the beginning of the semester.

Parameters

The actual descriptive values for the whole population (mean, variance etc).

Statistics

The descriptive values we have obtained by sampling the population (mean, variance etc).

Outcomes

The actual discrete data values obtained by sampling.

Thylacosmilus atrox (A South American close relative of marsupials)

Smilodon fatalis (A North American eutherian felid)

We'll use the word 'odds' in its conventional sense to explain this. So far through most of this semester we have been discussing probabilities. The probability gives the odds of an event given the parameters. This is written in an equation form like this:

$$P(x \mid \theta)$$

P = Probability
x = The outcomes or 'statistics', which is the data obtained
$\mid$ = given
θ = The parameters of the population

Because we can obtain probabilities, and by sampling from a population repeatedly, eventually arrive at a probability that is fairly 'stable' we consider a probability to be something that is knowable.

We cannot sample the entire population in most instances, and so for this reason the parameters are never definitely knowable, but, assuming the sampling from a population is **random** and **unbiased** we can **estimate** the parameters.

When attempting to estimate parameters we start to talk about **likelihoods** instead of probabilities. A **likelihood** gives the odds of the parameters given the outcomes (statistics). In a sense, we're trying to work backwards, and are thinking about: what is the likelihood of parameters being X is the outcomes (statistics) are x? Likelihood is written like this in an equation form:

$$L(\theta \mid x)$$

L = **Likelihood**
x = **The outcomes or 'statistics', which is the data obtained**
$\mid$ = **given**
θ = **The parameters of the population**

If sampling is random and unbiased then we can estimate the likelihood (unknown) from the probability (known). This is called the **likelihood function**.

$$L(\theta \mid x) = P(x \mid \theta)$$

There are some statisticians who argue that this estimation is invalid because it is making assumptions about the relationship between the statistics and parameters that can never actually be known. Increasingly, however, most scientists now accept that as long as we acknowledge that the likelihood is only an estimate, this approach is acceptable.

Maximum likelihood estimation (ML)

Maximum likelihood estimation is a method that can be used to estimate the parameters of a population given the outcomes (statistics) assuming a given hypothesis is true. The mathematics of how ML estimation works are beyond the scope of this course. In a practical sense the 'hypothesis' will be the set of predictors you have included in a model and their relationships with each other (+ or *).

In essence, maximum likelihood estimation picks values for the parameters that maximise the likelihood function. An intuitive way to think about this is that ML estimation picks values for the parameters that maximise the agreement between the observed data (the outcomes or statistics) and the hypothesized model.

A maximum likelihood function will range from 0 to 1, where 1 is a perfect model fit.

If you want to compare models to decide which hypothesized model best fits the observed data outcomes, you need to use ML

Restricted maximum likelihood estimation (REML)

Restricted maximum likelihood estimation is similar to maximum likelihood estimation, except that it uses a likelihood function generated from transformed values so that 'nuisance' parameters have no effect on the estimated parameters values.

An typical example of a nuisance parameter is the variance, which we are not usually interested in. More often we are interested in the means of groups, because we are interested in whether groups may have different or the same means.

Note though that variance in not always a nuisance parameter. For example, in evolutionary biology and examination of fitness we are often interested in the variance rather than the means of seed production among species or populations. If you want to test variances you often need to use very specific statistical approaches, because most approaches assume that variance is of little or no interest.

The mathematics underlying REML is beyond the scope of this course. The important thing to know about REML is that:

You cannot use REML to compare models

REML is better than ML if you want to look at the *P* values for predictors in a model

Can we select the best model and then look at its P values?

Really, really, no. The problem is that this would be like having your cake and eating it too. Through a process of comparing models we have already picked the model that represents a hypothesis that best fits the observed data. Interpreting *P* values is not really sensible because 1) it would be a surprise if the *P* values **were not** significant, and 2) we interpret *P* values when we are trying to falsify a bold prediction about data that has not been collected yet. By constructing the best possible model from a set of data and ***then*** testing P values we would be strongly biasing significance.

Remember that falsification is intended to get around the problem of induction. If we collect data, then use induction (which model selection falls broadly within) to build the best model, and then take *P* values from the best model, what we have done is build a hypothesis (model) from the data we have, and then test the hypothesis (model) using the data that was used to construct the model.

That would be highly circular. So how is model selection used and reported then? We use Information Criteria to pick the best model and the best model (and those that are nearly as good) are presented as the best explanations for the data.

Before looking at Information Criterion, however, let's look at Log Likelihood.

Q. What is a log likelihood? Why are log likelihoods used?

To explain what a log likelihood is, we need to start by explaining what the likelihood ratio is. We start with the following Law of Likelihood:

> *Within the framework of a statistical model, a particular set of data supports one statistical hypothesis better than another if the likelihood of the first hypothesis given the data exceeds the likelihood of the second hypothesis given the data.*

This can be written as so:

$$Likelihood\ ratio = \frac{P(x \mid H_1)}{P(x \mid H_2)}$$

The likelihood of the hypothesis given the outcomes in the data is assumed to be same as the probability of the outcomes given the hypothesis (assuming random and unbiased sampling). If we were interested in testing a null hypothesis this could be rewritten:

$$Likelihood\ ratio = \frac{Likelihood\ given\ the\ model\ of\ interest}{Likelihood\ given\ the\ alternative\ model}$$

A log likelihood test is used for model comparison, that is, deciding which model is better of two (or more) competing models. The test statistic is sometimes called the **Support** but is more often termed the **Deviance** (often denoted as ***D***), and it is calculated:

$$D = -2 * \ln\left(\frac{Likelihood\ given\ the\ model\ of\ interest}{Likelihood\ given\ the\ alternative\ model}\right)$$

It is called a **Deviance** value because high values indicate that the observed data are strongly deviating from the hypothesis that is being tested. Low values indicate that the observed data is not strongly deviating from the data being tested. Why do we transform it by -2? And is this related to the 'tests of deviance' we have looked at already? You know, those generalised linear model things?

Why do we transform by natural logs and multiply by -2?

Likelihoods are ratios. They range from 0 to 1, where 1 would be a perfect model fit where the model perfectly predicts all values of the response. However, working with ratios is mathematically tricky, and to make ratios easier to work with we log transform them.

A natural log-transformed (ln) likelihood will range from infinitely negative numbers for a very poor fit to 0 for a model that perfectly fits all values of the observed response. This is called the **log likelihood (LL)**.

One further step is applied. Negative numbers are inconvenient to work with, and to make the log of the likelihood into a positive linear relationship we multiple the log by -2. After this step a model that perfectly fits the data and predicts all values of the response perfectly will have a **-2*LL** of 0, and poor models will have increasingly high values of **-2*LL**.

Is this related to a deviance test?

Yes. This is the 'clever maths' that a deviance test uses to take a set of binary or count data and turn it into a form that can be examined using linear regression, from which *P* values can be obtained.

In a **generalised linear model**, we take advantage of the mathematical property that the log of a ratio is equal to the log of the first number minus the log of the second number.

$$D = -2 * \ln\left(\frac{Likelihood\ given\ the\ model\ of\ interest}{Likelihood\ given\ the\ alternative\ model}\right)$$

Only the parameters of interest

The Deviance is the distance from model of interest to everything fitted

$$D = -2 * \ln\left(\frac{P(x \mid \theta_{null})}{P(x \mid \theta_{full})}\right)$$

A parameter for every observation

$$D = -2 * \left(\ln\left(P(x \mid \theta_{interest}) \right) - \ln\left(P(x \mid \theta_{full}) \right) \right)$$

The higher the deviance the lower the goodness of fit of the full model (the model with all predictors included). The Deviance is then used in a way similar to how the *F* ratio is used in an ANOVA.

An ANOVA compares residuals in a way that is similar to how a regression model works except that residuals are taken from the mean of each group rather than across a range of values. An *F*-ratio is derived from the variance of the residuals.

***F* = signal / noise**
***F* = variance between treatments / variance within treatments**
***F* = mean squares of treatment / means squares of the error**
***F* = [sum of squares of treatment / [(T-1)] / [sum of squares of error / (n-1)]**

Remember hat the *F*-ratio is conceptually very similar to the *t*-value. It is a measure of the signal to noise in the data. If we view the Deviance as similar to the mean squares of the error, we can generate a signal to noise ratio and that can be used to generate *P* values for predictors in a model.

Q. What is an information criterion? How are information criteria used in model selection? What does 'most parsimonious' mean?

Information Criterion are used to judge the goodness of fit of a given model. They are in particular used for comparison among models with differing predictor values.

The Deviance (-2*LL) is a form of Information Criterion where the lower the value the lower the Deviance of the data from the hypothesis and therefore the better the fit. But usually, we modify this value to try and judge whether a model is not only a good fit but a parsimonious fit.

What is meant by parsimony?

Remember that simple models are always better. Remember also that eventually, just by adding more and more and more predictors to a model, we will be able to explain all values of the outcomes because there will be a predictor for every observed data point. Imagine you have three observations:

x	*Predictor 1*	*Predictor 2*	*Predictor 3*
31	HIGH	1	20
5	HIGH	10	19
17	LOW	3	14

If we used all three predictors, then we could state that *x* is equal to 17 when Predictor 1 has the factorial value LOW, and *x* is equal to 5 when Predictor 2 has a high value, and *x* is equal to 31 when Predictor 3 has a high value.

But this model isn't actually useful. It is 'over-parameterised' or 'over-fitted'. We have added too many predictors, and because we have one predictor per data point we are explaining everything, and therefore nothing.

A **parsimonious** model (and by extension a **parsimonious** hypothesis) is one where the fewest number of predictors explain the most variation in the observed outcomes. Information Criterion that attempt to pick the most parsimonious model are trading-off between perfect explanation of all observed outcomes on one hand and model simplicity on the other hand. They are attempting to pick an intermediate between explanatory power and too much complexity.

There are several different commonly used Information Criterion. **Akaike Information Criterion (AIC)**, **Bayesian Information Criterion (BIC)** and **Deviance Information Criterion (DIC)** are all commonly used and are superficially similar, in that they attempt to estimate how parsimonious a model is. AICs are the most commonly used of these Information Criterion, and this statistic works by taking the Deviance and penalising it for additional predictors.

AIC (Akaike information criterion)

Q. What is an Akaike information criterion? How is it used?

The AIC is usually written as follows:

$$AIC = 2k - 2ln(L)$$

But if we want to look at the logic of how it works, this is easier to understand.

$$AIC = -2ln(L) + 2k$$

-2ln(L) is **Deviance**. It is the same as **-2*LL** written differently. The value ***k*** is the number of predictors in the model. To understand why we add the predictors to the **Deviance** to penalise the **Deviance** remember that low values of Deviance imply a high goodness of fit and high values of **Deviance** imply a poor goodness of fit.

The decision to add ***k*** multiplied by **2** instead of just add ***k*** isn't arbitrary. It is based on information entropy, but getting into an explanation of how exactly we arrive at **2*k*** instead of understand ***k*** is beyond what we can reasonably tackle here.

In principal what this means, is that the Deviance will be calculated for a model and then it will be penalised for additional predictors. The Deviance will be penalised by two for one predictor, four for two predictors, six for three predictors and so on. That means that all else being equal, if two models give the same Deviance, but one model includes more predictors it will be more heavily penalised.

We take a lower **AIC** to be evidence of a more parsimonious model, and we usually take a difference of 2 as evidence that there is a difference between the models. We don't say 'significant' difference because we don't want to confuse AICs with *P* values. A Bayes Information Criterion (BIC) is similar but takes sample size into account too.

Bayes Information Criterion (BIC)

$$BIC = -2ln(L) + k * ln(n)$$

AIC & BIC :: R Code

Load libraries

```
install.packages("nlme")
library(nlme)
```

Load data

```
agilis <- read.table('agilis-abundance.csv',
header=T,sep=',')

str(agilis)
```

The package **nlme** provides another way to generate linear mixed effect models. The advice is that **lme4** is more reliable for model comparison, but if we are interested in *P* value **nlme** is easier to use. For most purposes, **nlme** will be fine for model comparison too, and it is a bit easier to learn with, so we'll have a go at using it today.

```
lme1 <- lme(sqrtAGILIS ~ HABITAT * SEX * ABH.m2 * LEAF.LITTER,
random = ~1 | YEAR / MONTH, data=agilis, method="REML")
```

Remember: use REML if interested in *P* values and ML if interested in model selection

To use a broadly falsificationist approach, we'd check *P* values for the fixed effects (**HABITAT**, **SEX**, **ABH.m2** and **LEAF.LITTER**) and calculate the percentage of variance explain for the random effects (**YEAR** and **SITE**) as well as provide an R^2 value for the model as a whole (i.e. how much variation in agile antechinus abundances does the model explain in total).

P values for the linear mixed effects model:

```
anova(lme1)
```

The R^2 value of the whole model:

```
(cor(fitted(lme1),getResponse(lme1))^2)
```

Percentage of variation explained by the random effects:

```
VarCorr(lme1)
```

Results for a falsificationist approach to a linear mixed effects model

RESULT

```
> anova(lme1)
                                  numDF denDF   F-value p-value
(Intercept)                           1    94 125.24971  <.0001
HABITAT                               1    94  30.71766  <.0001
SEX                                   1    94   7.00872  0.0095
ABH.m2                                1    94   0.97497  0.3260
LEAF.LITTER                           1    94   1.24246  0.2678
HABITAT:SEX                           1    94   2.21283  0.1402
HABITAT:ABH.m2                        1    94   0.44223  0.5077
SEX:ABH.m2                            1    94   0.42621  0.5154
HABITAT:LEAF.LITTER                   1    94   0.03283  0.8566
SEX:LEAF.LITTER                       1    94   2.43790  0.1218
ABH.m2:LEAF.LITTER                    1    94   4.31845  0.0404
HABITAT:SEX:ABH.m2                    1    94   1.03839  0.3108
HABITAT:SEX:LEAF.LITTER               1    94   0.34609  0.5577
HABITAT:ABH.m2:LEAF.LITTER            1    94   0.90513  0.3438
SEX:ABH.m2:LEAF.LITTER                1    94   1.70031  0.1954
HABITAT:SEX:ABH.m2:LEAF.LITTER        1    94   0.73939  0.3920

> (cor(fitted(lme1),getResponse(lme1))^2)
[1] 0.3735033

> VarCorr(lme1)
            Variance     StdDev
YEAR =      pdLogChol(1)
(Intercept) 0.001125295  0.03354541
MONTH =     pdLogChol(1)
(Intercept) 0.001131311  0.03363496
Residual    0.045092472  0.21234988
>
```

A model comparison approach to a linear mixed effects model instead

If instead we were going to use a model selection approach, we would construct more than one model and compare them. Let's make a reduced and full model. The reduce model will just be additive and won't have any interactions in it.

```
full.lme <- lme(sqrtAGILIS ~ HABITAT * SEX * ABH.m2 *
LEAF.LITTER, random = ~1 | YEAR / MONTH, data=agilis,
method="ML")

reduced.lme <- lme(sqrtAGILIS ~ HABITAT + SEX + ABH.m2 +
LEAF.LITTER, random = ~1 | YEAR / MONTH, data=agilis,
method="ML")
```

We can check the log likelihoods and AICs by using the commands:

```
logLik(full.lme)
logLik(reduced.lme)

AIC(full.lme)
AIC(reduced.lme)
```

AICs can be negative. This happens if the Deviance (-2*LL) is greater than the number of predictors multiplied by 2. The interpretation is the same. The lower the AIC, the more parsimonious the model. Which model is more parsimonious according to the AICs?

Which model is more parsimonious?

RESULT

```
> logLik(full.lme)
'log Lik.' 22.48653 (df=19)

> logLik(reduced.lme)
'log Lik.' 14.63009 (df=8)

> AIC(full.lme)
[1] -6.973065

> AIC(reduced.lme)
[1] -13.26019
```

The model with the lower AIC has the better parsimony. In this case, the reduced model. Incidentally, AIC values can be negative. You still take the lowest (most negative) value if your AICs are negative.

We can also compare models using the anova function:

```
anova(full.lme, reduced.lme)
```

RESULT

```
            Model df    AIC        BIC      logLik   Test   L.Ratio   p-value
full.lme     1    19  -6.973065   45.98928  22.48653
reduced.lme  2    8   -13.260189  9.03975   14.63009 1 vs 2 15.71288  0.1521
```

Because the Likelihood Ratio Test is the same as the log of likelihood 1 minus the log of likelihood 2, the L.Ratio is 22.49 - 14.63 = 15.72. The *P*-value indicates whether there is a significant difference between the models. Although there is no significant difference, we would still accept that Model 2 (reduced) is more parsimonious than Model 1 (full) because the difference in AICs is > 2 (i.e. 13.26 - 6.97 is > 2).

We can compare more than two models in this way. Let's create a another reduced model where **YEAR** is removed and we only control for **SITE** as a random factor.

```
noyear.lme <- lme(sqrtAGILIS ~ HABITAT + SEX + ABH.m2 +
LEAF.LITTER, random = ~1 | MONTH, data=agilis, method="ML")

anova(full.lme, reduced.lme, noyear.lme)
```

Which model is more parsimonious?

RESULT

```
> anova(full.lme, reduced.lme, noyear.lme)
            Model df        AIC      BIC   logLik   Test   L.Ratio p-value
full.lme        1 19  -6.973065 45.98928 22.48653
reduced.lme     2  8 -13.260189  9.03975 14.63009 1 vs 2 15.712876  0.1521
noyear.lme      3  7 -15.179955  4.33249 14.58998 2 vs 3  0.080234  0.7770
```

It is looking as if the reduced model that is also missing Year as a random effect may be preferable.

But what if we want to compare these models to the null model (with no predictors) and we want to check all possible combinations of predictors? There is a function called **dredge** in the library **MuMIn** that will do this:

Load libraries

```
install.packages("MuMIn")
library(MuMIn)
```

Apply the **dredge** function to the reduced model (this could take a while…):

```
dredge(reduced.lme)
```

RESULT

```
Global model call: lme.formula(fixed = sqrtAGILIS ~ HABITAT + SEX + ABH.m2 +
LEAF.LITTER,
    data = agilis, random = ~1 | YEAR/MONTH, method = "ML")
---
Model selection table
   (Int)     ABH.m2    HAB LEA.LIT SEX df logLik AICc  delta weight
11  0.422000           +           +   6  13.535 -14.3  0.00 0.355
15  0.268600           +   0.03757 +   7  14.283 -13.6  0.76 0.243
12  0.393000 3.115e-05 +           +   7  13.880 -12.8  1.57 0.162
16  0.236900 3.227e-05 +   0.03761 +   8  14.630 -12.0  2.36 0.109
3   0.473300           +               5  10.223  -9.9  4.41 0.039
7   0.318300           +   0.03797     6  10.953  -9.2  5.16 0.027
13 -0.005099               0.09590 +   6  10.744  -8.7  5.58 0.022
4   0.445000 3.057e-05 +               6  10.545  -8.3  5.98 0.018
8   0.289900 3.063e-05 +   0.03777     7  11.253  -7.5  6.82 0.012
14 -0.035160 3.085e-05     0.09603 +   7  11.046  -7.1  7.23 0.010
5   0.046220               0.09590     5   7.599  -4.7  9.65 0.003
6   0.019470 2.891e-05     0.09590     6   7.875  -3.0 11.32 0.001
9   0.314600                       +   5   0.225  10.1 24.40 0.000
10  0.286700 3.016e-05             +   6   0.490  11.8 26.09 0.000
1   0.365900                           4  -2.432  13.2 27.54 0.000
2   0.338100 2.999e-05                 5  -2.183  14.9 29.22 0.000
Random terms (all models):
'1 | YEAR', '1 | MONTH %in% YEAR'
```

The 'global model' is the model that has all variables included (based on what you used in your model construction). Model numbers are given on the left-hand side and the variables that are included or not included in each model are shown next. If you were to present this as a table in a paper you would (usually) only include the models that have AICs that are within a difference of 2 from the lowest AIC and the global model for comparison. The table below shows the output reformatted for inclusion in a report.

Table X. Demonstration of how to present model selection results in a table. The table caption will need to explain that *df* = degrees of freedom; LL = log likelihood; AICc = corrected AIC; ΔAICc = delta corrected AIC; *w* = model weights.

MODEL	PREDICTORS				*df*	LL	AICc	ΔAICc	*w*
1		HABITAT		SEX	6	13.5	-14.3	0.0	0.36
2		HABITAT	LEAF LITTER	SEX	7	14.3	-13.6	0.8	0.24
3	ABH	HABITAT		SEX	7	13.9	-12.8	1.6	0.16
GLOBAL	ABH	HABITAT	LEAF LITTER	SEX	8	14.6	-12.0	2.4	0.11

Note that we've renamed the models with numbers that make sense given that we've discarded a lot of the other models. These are models 11, 15 and 12 in the R output. The most parsimonious mode in the above example is the model that includes the predictors

HABITAT and **SEX** only, but models 2 and 3 are not different enough to be considered unreasonable explanations for the observed data as well. The above example is ignoring potential interactions in the model.

What happens if you **dredge** the full model we created earlier?

dredge(full.lme)

What happens when you dredge the full model? You should get output that looks something like this where all possible interaction terms have also been include as possible variables to include in the model. This is why you would typically only report the global model, the null model and the top 5-10 models (based on parsimony). Reporting all models would make for a very large (and not very meaningful) table.

```
Global model call: lme.formula(fixed = sqrtAGILIS ~ HABITAT * SEX * ABH.m2 * LEAF.LITTER,
    data = agilis, random = ~1 | YEAR/MONTH, method = "ML")
---
Model selection table
        (Int)     ABH.m2 HAB   LEA.LIT SEX ABH.m2:HAB ABH.m2:LEA.LIT ABH.m2:SEX HAB:LEA.LIT HAB:SEX LEA.LIT:SEX ABH.m2:HAB:LEA.LIT ABH.m2:HAB:SEX ABH.m2:LEA.LIT:SEX HAB:LEA.LIT:SEX ABH.m2:HAB:LEA.LIT:SEX df loglik  AICc delta weight
11    0.422000           +           +                                                                                                                                  6 13.535 -14.3  0.00  0.059
267   0.393200           +           +                                                      +                                                                           7 14.620 -14.2  0.09  0.056
783   0.056590           +  0.082280 +                                                      +         +                                                                 9 16.606 -13.6  0.75  0.040
15    0.268600           +  0.037570 +                                                                                                                                7 14.283 -13.6  0.76  0.040
271   0.239300           +  0.037600 +                                                      +                                                                         8 15.376 -13.5  0.87  0.038
12    0.393000  3.115e-05 +           +                                                                                                                                7 13.880 -12.8  1.57  0.027
816   0.211900 -1.332e-04 +  0.024080 +               5.230e-05                          +         +                                                               11 18.560 -12.7  1.65  0.026
268   0.363400  3.168e-05 +           +                                                      +                                                                         8 14.975 -12.7  1.67  0.025
48    0.424600 -1.343e-04 + -0.020590 +               5.230e-05                                                                                                    9 16.140 -12.6  1.68  0.025
304   0.395000 -1.338e-04 + -0.020580 +               5.230e-05                          +                                                                        10 17.278 -12.5  1.79  0.024
832   0.474700 -5.096e-04 + -0.034680 +         +     1.354e-04                          +         +                                                              12 19.717 -12.5  1.81  0.024
64    0.685300 -5.072e-04 + -0.078900 +         +     1.346e-04                                                                                                   10 17.229 -12.4  1.89  0.023
320   0.656700 -5.084e-04 + -0.079100 +         +     1.350e-04                          +                                                                        11 18.399 -12.4  1.97  0.022
784   0.024560  3.338e-05 +  0.082170 +                                                      +         +                                                       10 16.992 -12.0  2.36  0.018
16    0.236900  3.227e-05 +  0.037610 +                                                                                                                           8 14.630 -12.0  2.36  0.018
143   0.170300           +  0.061650 +                                          +                                                                                 8 14.607 -11.9  2.41  0.018
911  -0.040810           +  0.106300 +                                          +         +         +                                                       10 16.936 -11.9  2.47  0.017
272   0.207500  3.280e-05 +  0.037570 +                                                    +                                                                         9 15.741 -11.8  2.48  0.017
28    0.352100  7.431e-05 +           +         +                                                                                                                    8 14.555 -11.8  2.51  0.017
399   0.141500           +  0.061650 +                                          +         +                                                                       9 15.704 -11.8  2.56  0.016
284   0.322700  7.463e-05 +           +         +                                          +                                                                       9 15.660 -11.7  2.64  0.016
527   0.272200           +  0.036490 +                                                                    +                                                       8 14.285 -11.3  3.05  0.013
76    0.378000  5.499e-05 +           +                                    +                                                                                     8 14.088 -10.9  3.45  0.010
332   0.341200  5.556e-05 +           +                                    +                                                                                     9 15.189 -10.7  3.59  0.010
112   0.402500 -1.105e-04 + -0.020590 +               5.230e-05           +                                                                                    10 16.357 -10.7  3.63  0.010
4976  0.354200 -2.546e-04 + -0.027270 +               9.820e-05           +                    +         +                                                 +  13 20.059 -10.7  3.64  0.010
880   0.189700 -1.094e-04 +  0.024090 +               5.230e-05           +                    +         +                                                       12 18.787 -10.7  3.67  0.009
368   0.372800 -1.099e-04 + -0.020570 +               5.230e-05           +                    +                                                                 11 17.500 -10.6  3.77  0.009
4992  0.618300 -6.331e-04 + -0.086310 +         +     1.818e-04           +                    +         +                                                 +  14 21.262 -10.5  3.80  0.009
176   0.368000 -1.422e-04 + -0.004490 +               5.204e-05                      +                                                                             10 16.257 -10.5  3.83  0.009
896   0.452700 -4.861e-04 + -0.034720 +         +     1.355e-04           +                    +         +                                                       13 19.951 -10.5  3.86  0.009
128   0.663300 -4.837e-04 + -0.078930 +         +     1.347e-04           +                                                                                    11 17.452 -10.5  3.87  0.008
944   0.158600 -1.409e-04 +  0.039220 +               5.212e-05                      +         +         +                                                       12 18.667 -10.4  3.91  0.008
384   0.634700 -4.848e-04 + -0.079140 +         +     1.351e-04           +                    +                                                                       12 18.627 -10.3  3.99  0.008
432   0.340000 -1.416e-04 + -0.004933 +               5.208e-05                      +         +                                                                 11 17.390 -10.3  3.99  0.008
32    0.246300  6.240e-05 +  0.028590 +         +                                                                                                                  9 14.985 -10.3  3.99  0.008
2432  0.587400 -4.346e-04 + -0.079340 +         +     1.354e-04           +                    +                                                 +                13 19.854 -10.3  4.05  0.008
560   0.428200 -1.343e-04 + -0.021670 +               5.230e-05                                          +                                                       10 16.142 -10.3  4.06  0.008
800   0.033820  6.314e-05 +  0.073270 +         +                                                      +         +                                             11 17.353 -10.3  4.06  0.008
288   0.216800  6.275e-05 +  0.028610 +         +                                                      +                                                       10 16.099 -10.2  4.15  0.007
192   0.645200 -5.020e-04 + -0.067900 +         +     1.323e-04                      +                                                                             11 17.267 -10.1  4.24  0.007
960   0.439800 -5.051e-04 + -0.025120 +         +     1.334e-04                      +         +         +                                                       13 19.747 -10.1  4.26  0.007
80    0.214800  5.611e-05 +  0.037600 +                                    +                                                                                     9 14.842 -10.0  4.28  0.007
9103 -0.131500           +  0.128500 +                                                0         +         +                                       +          11 17.233 -10.0  4.30  0.007
576   0.688900 -5.072e-04 + -0.079980 +         +     1.346e-04                                            +                                                       11 17.231 -10.0  4.31  0.007
848   0.002392  5.726e-05 +  0.082160 +                                    +                     +         +                                                       11 17.214 -10.0  4.34  0.007
448   0.619100 -5.035e-04 + -0.068780 +         +     1.328e-04           +                    +                                                                       12 18.433 -10.0  4.38  0.007
 [ reached getOption("max.print") -- omitted 120 rows ]
Models ranked by AICc(x)
Random terms (all models):
'1 | YEAR', '1 | MONTH %in% YEAR'
```

Corrected AIC

Q. What is a corrected Akaike information criterion? How is it used?

$$AICc = AIC + \frac{2k\,(k+1)}{n - k - 1}$$

The corrected AIC takes into account sample size and is preferable to use when sample sizes are small, especially less than 10. The dredge function defaults to using **cAIC**, which is why the output reads **cAIC** instead of **AIC**. The equation uses the number of observations (n) and number of predictors (k).

A corrected AIC is usually a good choice, but it can under very particular situations generate a result that is uninterpretable. Work out the corrected AIC for the following situation.

AIC = 205.1
k = 9
n = 10

What is the corrected AIC for the above?

RESULT

```
> k=9
> n=10
> 205.1 + (2*k*(k+1))/(n-k-1)
[1] Inf
```

What happened in the above example? Well, **n-k-1** is zero, and you can't divide a number by zero. How would you avoid getting an undefined number for a corrected AIC? The best option is to be sure that your number of predictors is at least n-2 or lower. i.e. in the above example, K needed to be 8 predictors or less.

Code to generate 'rntransform'

The function 'runtransform' is used a lot in this PDF, but the library GenABEL isn't being regularly maintained at the point when I'm writing this. If you want to generate the function, you can run the following code in R (which was simply taken from a working version of GenABEL, so this should still be cited back to GenABEL).

```
# create ztransform first

ztransform <- function (formula, data, family = gaussian)
{
  if (missing(data)) {
    if (is(formula, "formula"))
      data <- environment(formula)
    else data <- environment()
  }
  else {
    if (is(data, "gwaa.data")) {
      data <- data@phdata
    }
    else if (!is(data, "data.frame")) {
      stop("data argument should be of gwaa.data or data.frame class")
    }
  }
  if (is.character(family))
    family <- get(family, mode = "function", envir = parent.frame())
  if (is.function(family))
    family <- family()
  if (is.null(family$family)) {
    print(family)
    stop("'family' not recognized")
  }
  if (is(try(formula, silent = TRUE), "try-error")) {
    formula <- data[[as(match.call()[["formula"]], "character")]]
  }
  if (is(formula, "formula")) {
    mf <- model.frame(formula, data, na.action = na.pass,
                      drop.unused.levels = TRUE)
    mids <- complete.cases(mf)
    mf <- mf[mids, ]
    y <- model.response(mf)
    desmat <- model.matrix(formula, mf)
    lmf <- glm.fit(desmat, y, family = family)
    resid <- lmf$resid
  }
  else if (is(formula, "numeric") || is(formula, "integer") ||
           is(formula, "double")) {
    y <- formula
    mids <- (!is.na(y))
    y <- y[mids]
    resid <- y
    if (length(unique(resid)) == 1)
      stop("trait is monomorphic")
    if (length(unique(resid)) == 2)
      stop("trait is binary")
  }
  else {
    stop("formula argument must be a formula or one of (numeric, integer, double)")
  }
  y <- (resid - mean(resid))/sd(resid)
  tmeas <- as.logical(mids)
  out <- rep(NA, length(mids))
  out[tmeas] <- y
  out
}
```

```
### create rntransform

rntransform <- function (formula, data, family = gaussian)
{
  if (is(try(formula, silent = TRUE), "try-error")) {
    if (is(data, "gwaa.data"))
      data1 <- phdata(data)
    else if (is(data, "data.frame"))
      data1 <- data
    else stop("'data' must have 'gwaa.data' or 'data.frame' class")
    formula <- data1[[as(match.call()[["formula"]], "character")]]
  }
  var <- ztransform(formula, data, family)
  out <- rank(var) - 0.5
  out[is.na(var)] <- NA
  mP <- 0.5/max(out, na.rm = T)
  out <- out/(max(out, na.rm = T) + 0.5)
  out <- qnorm(out)
  out
}

### test it worked on some numbers
rntransform(c(2,3,4,5,10,4,5,3,2,19))

ztransform(c(2,3,4,5,10,4,5,3,2,19))
```